Pemrograman DirectX dengan Delphi

Zamrony P Juhara

Bab 1
Pendahuluan

1.1 Sejarah DirectX.

Sebelum diluncurkannya Windows 95, program game umumnya dirilis untuk sistem operasi DOS dengan bantuan DOS extender seperti DOS4GW untuk mengatasi keterbatasan memori yang disediakan DOS.

Keluarnya Windows 95 merupakan tanda dari awal berakhirnya era DOS. Banyak pengembang game mulai menulis ulang (memporting) game-game mereka agar bisa berjalan optimal di Windows 95.

Graphic Device Interface (GDI) Windows tidak didesain untuk game sehingga game-game yang menggunakan GDI untuk menangani grafis game tampil dengan grafis tidak begitu bagus karena GDI relatif lambat.

Microsoft kemudian meluncurkan WinG (DIBSection) untuk mengatasi masalah para pengembang game. WinG jauh lebih baik dibandingkan GDI karena lebih cepat, namun problem yang sering dihadapi pengembang game untuk memporting game mereka ke Windows 95 adalah menampilkan grafis game secara full screen. Dengan WinG aplikasi game ditampilkan dalam window seperti aplikasi-aplikasi lainnya, sedangkan game umumnya berjalan pada mode full screen.

Selain itu program game umumnya dibuat agar berjalan secepat mungkin, sehingga banyak fungsi-fungsi yang dimiliki dirancang untuk memanipulasi perangkat keras secara langsung. Untuk program game yang berjalan pada sistem operasi DOS hal ini tidak menjadi masalah karena DOS mengijinkan program mengakses perangkat keras secara langsung, namun untuk game yang berjalan pada sistem operasi multi-tasking seperti Windows mengakses perangkat keras secara langsung tidak disarankan mengingat tiap aplikasi harus berbagi sumber daya dengan aplikasi lain.

Untuk mengatasi hal ini Microsoft merilis DirectX. Awalnya DirectX dikembangkan oleh RenderMorphics, Microsoft kemudian ikut bergabung mengembangkan DirectX. Versi awal DirectX banyak dikritik oleh pengembang game karena desainnya yang buruk dan tidak terdokumentasi dengan baik.

Mulai versi 3 Microsoft lebih serius mengembangkan DirectX guna mendorong lebih banyak pengembang game untuk membuat game yang berjalan pada sistem operasi Windows. DirectX pun akhirnya dirilis sebagai public domain sehingga DirectX dapat diperoleh dengan gratis.

Langkah Microsoft ini diikuti oleh para produsen perangkat keras grafik dengan menyertakan dukungan terhadap DirectX.

1.2 Apa Itu DirectX?

DirectX adalah sekumpulan *Application Programming Interface* (API) yang dirancang untuk menciptakan aplikasi game dan aplikasi multimedia yang berkinerja tinggi. Fungsi utama DirectX adalah menyediakan prosedur standar untuk mengakses berbagai perangkat keras yang berbeda. Pada versi 7 dan versi-versi sebelumnya DirectX terdiri atas komponen:

- *DirectDraw*, untuk menangani double buffering dan grafis 2D.
- *Direct3D*, menangani grafik 3D.
- *DirectSound*, menangani efek suara.
- *DirectPlay*, menangani jaringan (untuk game multiplayer).
- *DirectInput*, menangani input dari joystick, keyboard dan mouse.
- *DirectMusic*, untuk menangani proses memainkan musik. (DirectX 6 ke atas).

Untuk saat ini DirectX telah mencapai versi 8.1. Pada versi 8 ke atas, DirectDraw dan Direct3D digabung menjadi *DirectX Graphics* yang digunakan berfungsi untuk menangani grafis 2D dan 3D, sedangkan DirectSound dan DirectMusic digabung menjadi *DirectX Audio*. Namun komponen-komponen yang digabung tersebut masih tetap didukung oleh DirectX. Pada versi 8 terdapat tambahan komponen baru yaitu *DirectShow* yang berfungsi untuk video playback.

Seluruh komponen DirectX disusun sebagai *Component Object Model* (COM). COM adalah komponen software yang dapat di gunakan berulang-ulang (reuseable) dan memenuhi spesifikasi COM. Untuk menggunakan DirectX kita akan menggunakan objek COM yang telah disediakan oleh DirectX. Cara ini lebih mudah dibandingkan harus mengimplementasikan objek COM milik kita sendiri, karena cara ini mirip dengan cara menggunakan kelas pada Delphi atau C++. Walaupun demikian pengetahuan tentang cara kerja COM tetap diperlukan terutama ketika mengerjakan proses inisialisasi dan finalisasi. Pembahasan tentang pemrograman COM dengan Delphi akan diuraikan pada sub.bab 1.6, namun pembahasan ini hanya sebatas pengenalan terhadap pemrograman COM. Jika anda tertarik untuk memperdalam kemampuan anda dalam pemrograman COM, anda dapat mencari referensi lain yang khusus membahas tentang COM.

1.3 Komponen DirectX.

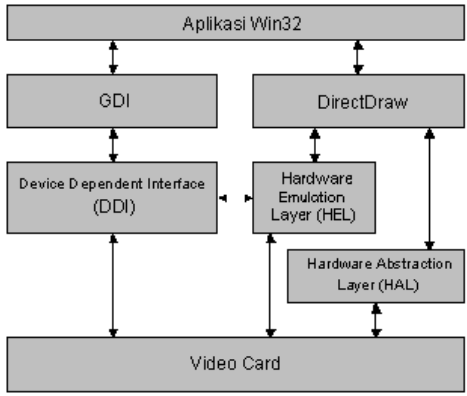
Karena tutorial ini hanya menjelaskan dasar-dasar penggunaan DirectX, maka penjelasan interface komponen DirectX hanyalah interface-interface yang dipakai dalam tutorial ini yakni IDirectDraw, IDirectSound dan IDirectInput.

1.3.1 DirectDraw.

DirectDraw pada dasarnya adalah sebuah video memory manager. Kemampuan terpentingnya adalah kemampuan untuk mengijinkan programmer menyimpan dan memanipulasi bitmap langsung di memori kartu grafis.

Dengan DirectDraw kita bisa memanfaatkan kemampuan perangkat kartu grafis secara optimal. Programmer aplikasi tidak perlu pusing terhadap berbagai perangkat keras kartu grafis yang berbeda-beda yang mungkin dimiliki pengguna. Selama kartu grafis tersebut mendukung DirectX, maka prosedur penggunaannya akan sama.

Berikut ini adalah skema hubungan antara GDI milik Windows dan DirectDraw.



Gambar 1.1 Skema DirectDraw dan GDI

DirectDraw seperti gambar diatas bekerja bersama GDI, keduanya memiliki akses ke perangkat keras melalui suatu *device-dependent abtraction layer*. Berbeda dengan GDI, DirectDraw selalu berusaha menggunakan kelebihan yang ada pada hardware jika fitur tersebut tersedia. Jika tidak maka DirectDraw mencoba mengemulasikan dengan HEL. Untuk bisa bekerja bersama GDI, DirectDraw menyediakan device context surface.

DirectDraw menggunakan flat memory model dan pengalamatan memori 32 bit karena DirectDraw berjalan pada sistem operasi Windows 95 ke atas, dengan demikian memori kartu grafis dan memori sistem dinyatakan sebagai blok penyimpanan besar. Jika anda pernah membuat program untuk DOS yang mengakses memori secara langsung dengan model pengalamatan *segment:offset* anda akan mengetahui flat memory model jauh lebih baik.

Selain itu DirectDraw juga memudahkan kita untuk melakukan *page flipping* dengan beberapa back buffer untuk aplikasi full screen. Pembahasan mengenai page flipping dan back buffer dijelaskan di Sub bab 1.4.

DirectDraw terdiri atas beberapa interface yakni IDirectDraw, IDirectDrawSurface, IDirectDrawPalette dan IDirectDrawClipper.

1.3.1.1 IDirectDraw.

Interface ini adalah interface utama yang harus diciptakan untuk bisa menggunakan DirectDraw. Interface ini pada dasarnya mewakili kartu grafis itu sendiri. Interface ini bisa diciptakan lebih dari satu jika komputer kita memiliki beberapa kartu grafis dan monitor, tetapi karena umumnya komputer hanya terdiri atas satu monitor dan satu kartu grafis maka biasanya aplikasi hanya membutuhkan satu interface IDirectDraw. Fungsi interface ini adalah untuk mengatur video mode, level kooperatif, menciptakan interface lain seperti IDirectDrawSurface, IDirectDrawClipper dan IDirectDrawPalette atau untuk mendapatkan pointer ke interface yang versinya lebih baru.

1.3.1.2 IDirectDrawSurface.

Interface ini adalah interface dimana kita akan menggambar apa yang ingin kita tampilkan di layar dengan menggunakan DirectDraw.

Interface ini mewakili memori video (VRAM) yang ada di kartu grafis. Jika memori kartu grafis cukup maka surface akan diciptakan di VRAM jika tidak maka surface diciptakan di memori sistem (RAM), walaupun demikian aplikasi dapat memerintahkan DirectDraw menciptakan surface di RAM jika diinginkan.

1.3.1.3 IDirectDrawPalette.

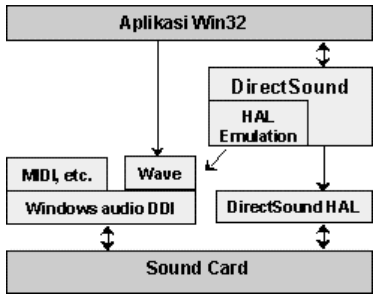
Interface ini hanya diperlukan untuk video mode 8 bit ke bawah, digunakan untuk mengontrol palette. Pada video mode 16 bit ke atas interface ini tidak diperlukan.

1.3.1.4 IDirectDrawClipper.

Interface ini mewakili daerah yang boleh digambar oleh DirectDraw, terdiri atas sekumpulan rectangle yang mengatur proses clipping. Interface ini bekerja mirip filter rendering atau mask pada program pengolah gambar.

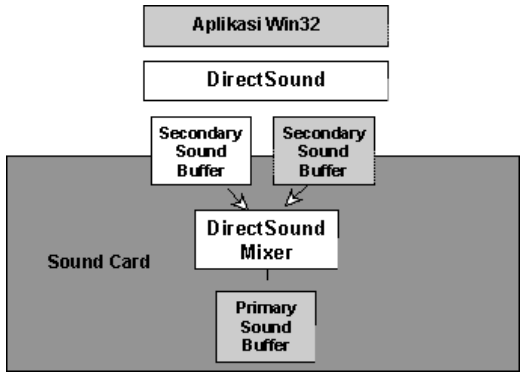
1.3.2 DirectSound.

DirectSound adalah komponen audio DirectX. Dengan DirectSound kita dapat melakukan mixing suara dengan hardware maupun software, merekam suara dan menambahkan efek ke suara yang kita mainkan.



Gambar 1.2a Skema DirectSound

Pada dasarnya DirectSound adalah sebuah sound mixing engine. Aplikasi menyimpan sekumpulan suara di suatu penampung yang disebut secondary buffer. Untuk melakukan proses mixing, DirectSound menggabungkan suara-suara ini dan menuliskannya ke primary buffer sehingga bisa didengar.



Gambar 1.2b Cara kerja DirectSound

Pembahasan lengkap tentang DirectSound akan dijelaskan pada Bab 8.

1.3.2.1 IDirectSound.

Interface utama yang harus diciptakan untuk menggunakan DirectSound. Interface ini mewakili sound card yang ada pada komputer. Interface ini digunakan untuk mengatur level kooperatif, menciptakan sound buffer dan lain-lain.

1.3.2.2 IDirectSoundBuffer.

Interface ini mewakili suara yang hendak dimainkan pada komputer. Sound buffer bisa diciptakan di memori sound card atau di memori sistem.

1.3.2.3 IDirectSound3DBuffer.

Interface ini mewakili suara 3D yang hendak dimainkan. Perbedaan IDirectSoundBuffer dengan IDirectSound3Dbuffer terletak pada adanya tambahan informasi posisi, orientasi dan lingkungan sound buffer di ruang 3D. Pemrograman suara 3D belum dibahas pada tutorial ini.

1.3.2.4 IDirectSound3DListener.

Interface ini digunakan untuk mengatur deskripsi pendengar (listener) seperti posisi pendengar, orientasi pendengar dan lingkungan ruang tempat pendengar. Interface ini berkaitan dengan interface IDirectSound3Dbuffer. Pada tutorial ini interface ini belum dibahas.

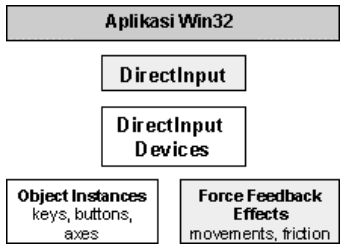
1.3.2.5 IDirectSoundCapture.

Digunakan untuk merekam suara dari mikropon atau peralatan lain. Untuk aplikasi game, interface ini umumnya tidak begitu diperlukan, mengingat aplikasi game hanya peduli pada bagaimana memainkan suara. Interface ini tidak akan dibahas pada tutorial ini.

1.3.2.6 IDirectSoundNotify.

Digunakan untuk komunikasi antara aplikasi dengan DirectSound melaui pesan (message). Interface ini juga belum kita perlukan untuk pengenalan dasar-dasar penggunaan DirectX.

1.3.3 DirectInput.



Gambar 1.3 Skema DirectInput

1.3.3.1 IDirectInput.

Interface ini adalah interface utama DirectInput yang harus diciptakan untuk menggunakan DirectInput.

1.3.3.2 IDirectInputDevice.

Digunakan untuk komunikasi dengan perangkat keras (keyboard, mouse, joystick).

1.4 Beberapa Konsep Dalam Dunia Game.

1.4.1 Pixel.

Istilah pixel berasal dari singkatan *Picture Element*. Pixel merupakan elemen grafis terkecil yang umumnya dipandang sebagai sebuah titik di layar monitor. Definisi ini tidak begitu tepat secara matematis karena titik tidak memiliki lebar dan tinggi, sebaliknya pixel memilikinya, namun definisi ini cukup mudah dipahami sehingga di tutorial ini istilah pixel mengacu pada titik di layar monitor.

1.4.2 RGB.

RGB berasal dari singkatan Red Green Blue, digunakan untuk menyatakan warna. Semua warna dapat dinyatakan sebagai komposisi komponen warna merah, warna hijau dan warna biru. Komponen warna merah, hijau dan biru selanjutnya akan disebut R, G dan B.

Jika intensitas RGB adalah dari 0-255 dengan intensitas maksimum 255, maka untuk menghasilkan warna putih RGB=(255,255,255), untuk warna hitam RGB=(0,0,0), warna merah RGB=(255,0,0), warna hijau RGB=(0,255,0), warna biru RGB=(0,0,255), warna kuning RGB=(0,255,255) dan sebagainya.

1.4.3 Kedalaman Warna (Color Depth).

Istilah kedalaman warna menyatakan berapa banyak bit yang dipakai untuk menyimpan informasi warna sebuah pixel. Semakin banyak bit yang digunakan semakin banyak warna yang dapat ditampilkan pada satu saat, tapi semakin besar pula memori yang dibutuhkan untuk menyimpan informasi warna.

Kedalaman warna yang ada saat ini adalah:

- *1 bit*, pada kedalaman warna ini jumlah warna yang dapat ditampilkan adalah 2^1=2 warna yaitu hitam dan putih. Untuk menyimpan informasi warna 8 pixel hanya dibutuhkan 1 byte.
- *2 bit*, jumlah warna yang dapat ditampilkan 2^2=4 warna. 1 byte dapat menyimpan informasi warna 4 pixel.
- *4 bit*, jumlah warna yang dapat ditampilkan adalah 2^4=16 warna. 1 byte dapat menampung informasi warna 2 pixel.
- *8 bit*, jumlah warna yang dapat ditampilkan 2^8=256 warna. Kedalaman warna ini pada era DOS banyak dipakai oleh game mengingat jumlah warna yang cukup banyak. Kedalaman warna 1,2 ,4 dan 8 bit menggunakan palette (lihat subbab 1.4.4 mengenai palette).

- *15 bit*, informasi warna tiap pixel disimpan masing-masing sebanyak 5 bit untuk R,G dan B. Karena komponen RGB masing-masing 5 bit maka tiap komponen RGB memiliki 32 tingkat intensitas dengan jumlah total warna 2^{15}=32768 warna. Berbeda dengan kedalaman warna 8 bit ke bawah, untuk kedalaman warna 15 bit ke atas tidak mengunakan palette. Informasi warna dinyatakan langsung oleh RGB. Kedalaman warna 15 bit kadang-kadang disebut sebagai kedalaman warna 16 bit berformat 555 karena informasi warna disimpan dalam 1 word (16 bit) dengan 1 bit paling signifikan diabaikan. Untuk mengkodekan nilai RGB menjadi informasi warna dengan kedalaman warna 15 bit dapat dipakai rumus berikut:

  Color=((R and 32) shl 10) or ((G and 32) shl 5) or (B and 32)

- *16 bit*, hampir sama dengan kedalaman warna 15 bit. Perbedaannya terletak pada jumlah bit RGB yaitu 5 bit R, 6 bit G dan 5 bit B (format 565). Jumlah warna yang dapat ditampilkan 2^{16}=65536 warna. Alasan mengapa jumlah bit komponen hijau lebih banyak daripada komponen merah dan biru adalah karena mata manusia paling sensitif terhadap warna hijau. Kedalaman warna 16 bit kadang-kadang disebut sebagai kedalaman warna 16 bit berformat 565. Untuk mengkodekan nilai RGB menjadi informasi warna dengan kedalaman warna 16 dapat digunakan rumus berikut:

  Color=((R and 32) shl 11) or ((G and 64) shl 6) or (B and 32)

- *24 bit*, pada kedalaman warna ini tiap komponen RGB disimpan dalam data 8 bit (1 byte) sehingga terdapat 256 tingkat intensitas untuk tiap komponen RGB. Total warna yang dapat ditampilkan adalah 2^{24}=16777216 warna.
- *32 bit*, mirip dengan kedalaman warna 24 bit karena tiap komponen RGB juga disimpan dalam data 8 bit. Perbedaannya terletak pada tambahan informasi transparansi (alpha) sebanyak 8 bit yang disimpan di 8 bit paling signifikan.

1.4.4 Palette.

Pengertian palette dalam pemrograman grafis mirip palette yang digunakan oleh pelukis. Pada palette disimpan informasi komposisi RGB. Pada kedalaman warna 8 bit, apa yang dialokasikan ke memori layar sebenarnya adalah nomor warna yang merupakan indeks pada tabel palette. Kartu grafis kemudian akan melihat ke tabel palette untuk menentukan RGB warna tersebut, baru kemudian menampilkannya. Oleh karena itu nomor warna yang sama, mungkin saja memiliki komposisi RGB yang berbeda, tergantung pada tabel palette yang sedang aktif saat ini.

Sistem palette ini banyak digunakan pada kedalaman warna 8 bit ke bawah. Dengan palette kita dapat melakukan animasi palette seperti efek meredupkan gambar menjadi gelap (fade out) atau sebaliknya (fade in), efek silau dengan mudah. Sistem palette tidak cocok digunakan, apabila kita ingin membuat aplikasi image processing.

1.4.5 Double Buffering.

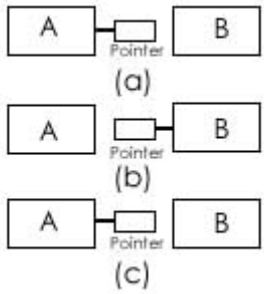
Jika kita menggambar langsung ketika melakukan animasi di layar akan terlihat banyak kerjapan (flicker). Untuk meminimalkan flicker, diciptakan teknik double buffering.

Teknik ini membutuhkan memori relatif besar karena dibutuhkan dua buah buffer untuk melakukan proses animasi yaitu front buffer dan back buffer. Front buffer adalah buffer yang terlihat oleh pengguna sedangkan back buffer adalah buffer yang tidak terlihat.

Dengan teknik ini kita tidak menggambar langsung pada front buffer, melainkan pada back buffer. Semua proses penggambaran dikerjakan pada back buffer. Setelah semua proses penggambaran selesai, back buffer ditransfer ke front buffer.

Ada dua cara proses pemindahan back buffer ke front buffer yaitu:

- *Blitting*, blit berasal dari singkatan *bit block transfer*. Proses blitting pada dasarnya adalah proses pengkopian data dari back buffer ke front buffer.
- *Page Flipping*, page flipping atau sering disebut flipping adalah teknik pemindahan back buffer ke front buffer dengan cara memindahkan penunjuk start scanning kartu grafis ke alamat buffer yang siap ditampilkan. Tidak ada proses pengkopian data, yang terjadi adalah proses pertukaran front buffer dan back buffer. Dengan cara ini back buffer akan menjadi front buffer dan front buffer menjadi back buffer. Cara kerja page flipping dapat dijelaskan melalui Gambar 1.4 Page Flipping. Pada Gambar 1.4.a pointer menunjuk ke buffer A, pengguna melihat buffer A, semua proses penggambaran dikerjakan di buffer B, Gambar 1.4.b buffer B siap ditampilkan, pointer menunjuk buffer B sehingga pengguna melihat buffer B, proses penggambaran dikerjakan di buffer A, Gambar 1.4.c buffer A siap ditampilkan pointer di pindahkan ke buffer A pengguna melihat buffer A, proses penggambaran dikerjakan di buffer B demikian seterusnya.



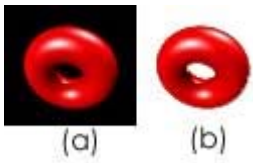
Gambar 1.4 Page Flipping

1.4.6 Bitmap dan Sprite.

Istilah bitmap berasal dari singkatan *bit mapping*. Bitmap adalah kumpulan sejumlah bit yang disusun sedemikian rupa sehingga membentuk gambar. Bitmap berbentuk persegi panjang.

Sprite pada dasarnya adalah bitmap juga, perbedaannya terletak pada adanya daerah yang dianggap sebagai daerah transparan. Daerah transparan ini tidak akan ditampilkan ketika sprite digambar. Dengan sprite kita bisa menampilkan gambar yang tidak berbentuk persegi panjang.

Contoh sprite yang paling mudah dilihat adalah kursor mouse.



Gambar 1.5 Bitmap dan Sprite (a) Bitmap (b) Sprite dengan daerah transparan

1.5 Elemen Game.

Proses pembuatan sebuah game sebenarnya terbagi atas lima bagian besar yakni: desain game, grafis game, musik dan efek suara, pemrograman game dan pengetesan.

1.5.1 Desain Game.

Desain game, konsentrasi utamanya adalah pada proses penyusunan ide cerita (skenario game), desain user interface game termasuk proses interaksi antara pemain dengan game (proses input output game), merancang spesifikasi-spesifikasi untuk grafis, suara dan pemrograman, merancang level dan peraturan dalam game dan lain-lain.

Desain game merupakan elemen yang kompleks, karena melibatkan berbagai aspek. Bagian ini merupakan landasan bagi proses-proses lain, sehingga desain game yang bagus selain menentukan kualitas game yang dihasilkan, juga akan memudahkan pengerjaan proses-proses lainnya. Proses ini ditangani oleh seorang desainer game.

1.5.2 Grafis Game.

Grafis game berurusan dengan proses pembuatan gambar-gambar yang digunakan game, baik gambar animasi tiap karakter dalam game, gambar background atau animasi pembuka (opening movie).

Grafis game ditangani oleh desainer grafis dan artis. Artis biasanya bertanggung jawab terhadap gambar animasi dalam bentuk dasar, dimana gambar

dasar ini dikerjakan dengan tangan. Gambar ini kemudian diubah menjadi gambar digital oleh desainer grafis.

Proses pembuatan gambar-gambar tersebut mengikuti spesifikasi yang telah ditentukan oleh desainer game.

1.5.3 Musik dan Efek Suara.

Memainkan sebuah game tanpa suara akan terasa hampa dan kosong walaupun game tersebut ditunjang oleh grafis yang bagus. Oleh karena itu musik dan efek suara yang bagus dan sesuai sangat diperlukan.

Bagian ini menangani proses pembuatan musik dan efek suara yang digunakan dalam game. Musik dan latar biasanya ditangani oleh seorang komposes musik. Pembuatan musik dan efek suara.

Seperti halnya grafis game, pembuatan musik dan efek suara game juga mengikuti spesifikasi yang ditentukan oleh desainer game.

1.5.4 Pemrograman Game.

Pada bagian ini semua spesifikasi yang ada dalam desain game disatukan menjadi satu aplikasi game. Proses ini jauh lebih mudah dikerjakan bila desain game tersusun dengan baik.

1.5.5 Pengetesan Game.

Pengetesan game dilakukan oleh tester. Tester ini mencoba program game yang telah disusun seperti layaknya pemain biasa, tester harus mencoba berbagai kemungkinan aksi yang mungkin dilakukan pemain dan mencatat apakah program game yang sedang diuji ada cacatnya.

Hasil laporan ini dikembalikan tester kepada desainer game untuk dievaluasi.

1.6 Sekilas Tentang Pemrograman COM dengan Delphi.

Seperti yang sudah disebutkan di atas, seluruh komponen DirectX disusun menggunakan COM. Pada Delphi 3 Borland menambahkan kata tercadang *interface* ke Delphi. Interface merupakan landasan pemrograman COM, interface memiliki kemiripan dengan kelas abstrak. Kelas abstrak seperti yang anda ketahui, memiliki suatu metode yang telah dideklarasikan tetapi belum diimplementasikan. Hal ini terjadi mengingat kita belum tahu cara mengimplementasikan metode tersebut Kelas turunannya yang harus mengimplementasikan metode yang belum lengkap tersebut.

Contoh deklarasi kelas abstrak.

```
Type TBinatang=class
        Public
          Procedure Suara;virtual; abstract;
        End;
        TMonyet=class(Tbinatang)
         Public
           Procedure Suara;
         End;
```

Pada contoh di atas metode *Suara* milik kelas TBinatang dideklarasikan sebagai metode abstrak dengan menambahkan kata tercadang *virtual;abstract;* karena kita belum mengetahui bagaimana bunyi suara binatang, mengingat tiap binatang memiliki suara yang berbeda-beda.

Agar metode ini berguna maka kita harus menyusun kelas yang lebih spesifik lagi yang merupakan turunan dari kelas bnatang tersebut misalnya kelas TMonyet. Karena monyet memiliki suara yang khas maka kita mengetahui cara untuk memgimplementasikan metode Suara tersebut.

Kelas abstrak biasanya tidak kita ciptakan *object instance*-nya karena umumnya kelas abstrak hanyalah landasan bagi kelas-kelas turunannya.

Jika anda mencoba menciptakan object instance suatu kelas abstrak, misal dengan contoh rutin berikut:

```
Procedure Test;
Var binatang:TBinatang;
Begin
   Try
     Binatang:=Tbinatang.Create;
     Binatang.Suara;
   Finally
    Binatang.Free;
   End;
End;
```

Potongan program di atas dapat dikompilasi dengan sukses, tetapi Delphi akan menampilkan pesan peringatan: *Constructing instance of 'TBinatang' containing abstract method 'TBinatang.Suara'*.

Secara konseptual, sebuah interface bisa dianggap sebagai sebuah kontrak antara pembuat interface dengan pengguna interface. Pembuat interface menentukan spesifikasi interface, sedangkan pengguna interface menyusun implementasi lengkap dari interface tersebut dengan mengacu pada spesifikasi yang telah dibuat oleh pembuat interface.

1.6.1 Perbedaan Antara Interface dan Kelas.

Meskipun interface mirip dengan kelas abstrak namun terdapat beberapa perbedaan penting. Berikut ini kita tuliskan lagi deklarasi kelas abstrak TBinatang dan interface IBinatang.

```
Type TBinatang=class
    Public
      Procedure Suara;virtual; abstract;
    End;
  IBinatang=interface
      Procedure Suara;
    End;
```

Karakteristik interface adalah sebagai berikut:

- Interface dideklarasikan sebagai tipe interface menggunakan kata tercadang *interface* dan bukan tipe kelas. Dengan konvensi nama interface didahului oleh huruf I, sedangkan kelas dimulai dengan huruf T.
- Semua interface diturunkan langsung atau tidak langsung dari *IUnknown*. IUnknown adalah basis semua interface seperti halnya *TObject* yang merupakan basis semua kelas dalam Delphi.
- Interface tidak dapat diciptakan instance-nya. Potongan program berikut tidak akan dapat dikompilasi oleh Delphi

```
Var Binatang:IBinatang;
Begin
  Binatang:=IBinatang.Create;
End;
```

  Delphi akan menampilkan pesan kesalahan “Object or class type required”.

- Pada interface tidak diperkenankan adanya metode yang bersifat public, private, protected maupun published. Semua metode adalah public, namun kita tidak boleh menuliskan kata tercadang *public*, *private*, *protected* dan *published*.
- Pada interface tidak diijinkan mendeklarasikan variabel. Pada interface hanya dideklarasikan metode-metode. Bagaimana metode itu akan diimplementasikan tidak ada pembatasan.
- Semua fungsi dan prosedur yang dideklarasikan adalah fungsi dan prosedur abstrak virtual, namun kata tercadang *abstract* dan *virtual* tidak boleh digunakan.

Setelah interface didefinisikan dan dirilis untuk digunakan oleh orang lain, interface tidak dapat dimodifikasi. Jika kita ingin menambah kemampuan sebuah interface, kita dapat menciptakan versi baru interface tersebut, dengan cara mengimplementasikan sebuah interface baru.

Kita tidak harus mengimplementasi ulang interface jika kita hanya ingin menambahkan kemampuan baru ke interface dan kemampuan lain tetap sama. Kita cukup menurunkan interface ini dari interface yang lama. Contoh interface yang lama adalah Iengine, kita ingin menambahkan prosedur baru bernama Hide ke interface, untuk itu kita harus membuat interface baru, misalnya kita beri nama IEngine2 dimana kedua interface ini adalah deklarasinya sebagai berikut:

```
Type IEngine=interface
        Procedure Show;
    End;
   IEngine2=interface
       Procedure Show;
        Procedure Hide;
     End;
```

atau dapat berbentuk seperti berikut

```
Type IEngine2=interface(IEngine)
        Procedure Hide;
     End;
```

1.6.2 Deklarasi Interface.

```
Type IEngine=interface
      ['{4FB6413E-FAD1-4BDE-86C7-3AB360FB5269}']
      Procedure Show;
    End;
```

Di atas ini adalah contoh lain deklarasi interface. Perbedaannya dengan deklarasi sebelumnya adalah adanya string angka di bawah nama interface. String angka ini disebut *Globally Unique Identifier* (GUID). Semua interface membutuhkan GUID yang unik agar bisa berjalan dengan benar.

1.6.2.1 GUID.

Sebuah GUID adalah bilangan 16 byte yang unik. Algoritma yang digunakan untuk menciptakan GUID cukup kompleks. Algoritma ini mengikutsertakan data tanggal, waktu, nomer proses program yang menghasilkan GUID dan ID kartu jaringan jika kartu jaringan ada.

Jika kita memiliki kartu jaringan, maka GUID yang dihasilkan dijamin akan unik, tetapi bila tidak, maka GUID yang dihasilkan unik secara statistik karena besarnya jumlah digit GUID dan kompleksnya algoritma yang digunakan.

Membuat GUID adalah langkah penting dalam membuat interface. Jangan pernah mengkopi GUID sebuah interface ke interface lain. Untuk membuat GUID dengan Delphi caranya adalah kita tempatkan kursor pada posisi dimana kita akan

menyisipkan GUID (pada baris dibawah sesudah nama interface) dan tekan tombol Shift+Ctrl+G.

1.6.2.2 Struktur TGUID.

```
TGUID=record
    D1:LongWord;
    D2:Word;
    D3:Word;
    D4:array[0..7] of Byte;
  End;
```

GUID dapat ditulis sebagai

```
MyGUID:=['{4FB6413E-FAD1-4BDE-86C7-3AB360FB5269}'];
```

Atau

```
MyGUID:TGUID=
(D1:$4FB6413E;D2:$ FAD1;D3:$4BDE;D4:($86,$C7,$3A,$B3,$60,$FB,$52,$69));
```

1.6.3 Menciptakan dan Membebaskan Interface.

Interface, seperti yang telah disebutkan diatas, tidak dapat diciptakan instance-nya. Agar interface tersebut bisa digunakan maka kita harus membuat implementasi interface tersebut.

Penulis tidak akan membahas lebih jauh mengenai proses membuat implementasi interface, karena fokus tutorial ini bukan pemrograman COM, selain itu mengunakan interface-interface yang disediakan oleh DirectX caranya hampir sama dengan menggunakan kelas biasa. Untuk menciptakan interface-interface DirectX kita akan menggunakan fungsi-fungsi yang telah disediakan oleh DirectX.

Yang perlu mendapat perhatian adalah ketika membebaskan memori interface. Interface menggunakan mekanisme *reference count* untuk membebaskan untuk memori. Reference count adalah suatu bilangan yang mencatat berapa banyak referensi yang menunjuk ke interface. Jika reference count sama dengan nol maka interface akan dibebaskan.

Untuk mengurangi reference count digunakan fungsi *Release* sedangkan untuk menambah reference count digunakan *AddRef*. Kedua fungsi ini adalah fungsi anggota interface IUnknown. Fungsi lain yang merupakan fungsi anggota IUnknown adalah *QueryInterface* yang digunakan untuk medapatkan pointer ke suatu interface.

Karena semua interface diturunkan langsung maupun tidak langsung dari IUnknown maka tiap kelas yang mengimplementasikan sebuah interface harus melengkapinya dengan implementasi fungsi Release, AddRef dan QueryInterface.

Jika anda menggunakan Delphi atau C++ Builder maka anda tidak perlu memanggil Release untuk membebaskan interface karena proses ini dikerjakan oleh

Delphi secara otomatis ketika program dikompilasi. Delphi akan menambahkan kode fungsi AddRef dan Release ke dalam kode program ditempat yang sesuai tanpa campur tangan programmer. Untuk kompiler lain mungkin anda harus menuliskan memanggil AddRef sebelum menggunakan interface dan memanggil Release setelah selesai menggunakan interface.

Dengan Delphi, cara lain jika kita ingin memaksa suatu interface dibebaskan adalah dengan mengisikan nilai variabel interface sama dengan nil.

Bab 2
Menggunakan DirectDraw

2.1 Pendahuluan.

Umumnya program game terbagi atas tiga bagian besar yakni inisialisasi, loop utama game dan finalisasi. Pada bagian inisialisasi program game menyiapkan segala sesuatu yang diperlukan bagi game tersebut, seperti membaca file-file data game, menyiapkan memori untuk menampung data-data, mengatur mode grafik dan lain-lain. Pada bagian loop utama dilakukan pengecekan input pengguna, mengecek berbagai kondisi dan kejadian dalam program game seperti tabrakan, tembakan, melakukan rendering di back buffer dan lain-lain.

Aplikasi yang memanfaatkan DirectDraw umumnya melakukan bagian-bagian berikut:

1. Menciptakan window.
2. Menyiapkan dan mengatur variabel-variabel DirectDraw.
3. Menciptakan objek DirectDraw.
4. Mengatur level kooperatif.
5. Mengatur display mode (untuk mode full-screen).
6. Menciptakan surface utama (primary surface) dan back buffer.
7. Menciptakan offscreen surface untuk menampung data bitmap atau sprite (bila diperlukan).
8. Menciptakan clipper dan memasangkan clipper ke window (untuk mode windowed).
9. Melakukan rendering di back buffer.
10. Melakukan flipping/blitting back buffer ke primary surface.
11. Mengulangi langkah 9 sampai pengguna menghentikan aplikasi.
12. Membebaskan memori yang digunakan dan kembalikan display mode ke kondisi semula.

2.2 Menciptakan Objek DirectDraw.

Untuk menciptakan objek DirectDraw digunakan fungsi *DirectDrawCreate*. Deklarasi fungsi ini sebagai berikut:

```
var DirectDrawCreate : function (lpGUID: PGUID;
    out lplpDD: IDirectDraw;
```

pUnkOuter: IUnknown) : HResult; stdcall;

Parameter:

lpGUID

Global unique identifier device driver perangkat keras grafik. Jika lpGUID=nil maka DirectDraw akan menggunakan default device driver.

lplpDD

Objek DirectDraw yang akan dibuat.

pInkOuter

Tidak dipakai dan bisa diisi nil.

Setelah pemanggilan fungsi ini, kita bisa mengecek status keberhasilannya dengan fungsi *Failed*. Jika hasil pemanggilan fungsi ini bernilai true maka fungsi yang dikerjakan sebelumnnya gagal.

Function Failed(hr:Hresult):Boolean;

Jika objek DirectDraw berhasil diciptakan maka lplpDD akan berisi objek DirectDraw yang telah dibuat. Objek ini selanjutnya dapat digunakan untuk menciptakan objek-objek lain seperti: surface, clipper, palette, juga untuk mengatur display mode dan lain-lain.

2.3 Mengatur Level Kooperatif dan Display Mode.

Setelah objek DirectDraw berhasil dibuat, hal berikutnya yang harus dilakukan adalah mengatur level kooperatif. Level kooperatif perlu diset untuk menentukan bagaimana aplikasi kita akan dijalankan. Apakah tampilannya full-screen atau dalam window (windowed mode) seperti aplikasi Windows biasa.

Untuk mengatur level kooperatif digunakan fungsi anggota objek DirectDraw *SetCooperativeLevel*.

function SetCooperativeLevel (hWnd: HWND; dwFlags: DWORD) : HResult; stdcall;

Parameter

hWnd

handle window aplikasi kita.

dwFlags

Flag yang memberitahukan DirectDraw level kooperati apa yang ingin kita gunakan.

Tabel Konstanta Flag untuk Level Kooperatif

DDSCL_FULLSCREEN	Jika kita ingin menggunakan mode full screen.

DDSCL_ALLOWREBOOT	Dengan flag ini maka tombol kombinasi untuk reboot (Ctrl+Alt+Del) bisa digunakan sehingga memungkinkan pengguna melakukan reboot saat menjalankan aplikasi kita.
DDSCL_NOWINDOWCHANGE	Aplikasi tidak mengijinkan pengguna untuk mengubah window aplikasi seperti minimize/maximize window (Alt+Tab). Ketika suatu aplikasi DirectDraw full-screen di-minimize semua surface akan lenyap karena GDI milik Windows mengambil alih kendali perangkat grafik. Oleh karena itu, saat pengguna kembali ke aplikasi DirectDraw yang di-minimize -jika aplikasi tersebut tidak mengembalikan surface yang hilang- yang ditampilkan hanya warna hitam. Dengan level kooperatif ini problem tersebut bisa diatasi, namun aplikasi menjadi tidak fleksibel.
DDSCL_NORMAL	Aplikasi berjalan dalam mode windowed seperti aplikasi window biasa.
DDSCL_EXCLUSIVE	Aplikasi memperoleh prioritas tertinggi. Flag ini biasanya dikombinasikan dengan flag DDSCL_FULLSCREEN

Keluaran fungsi ini bisa dicek status keberhasilannya dengan *Failed.*

Setelah mengatur level kooperatif berikutnya adalah mengatur display mode. Untuk mengatur display mode digunakan fungsi anggota objek DirectDraw *SetDisplayMode*.

function SetDisplayMode(dwWidth: DWORD; dwHeight: DWORD; dwBpp: DWORD) : HResult; stdcall;

Parameter *dwWidth* dan *dwHeight* menyatakan lebar dan tinggi resolusi layar yang akan kita gunakan, misal 640x480, 800x600 dan lain-lain. *dwBpp* (bit per pixel) menyatakan kedalaman warna yang akan kita gunakan, misal 2, 4, 8, 16, 24, 32.

Untuk saat ini kedalaman warna 2 bit dan 4 bit tidak lagi dipergunakan dalam game. Game-game saat ini umumnya menggunakan kedalaman warna 16 bit, namun trend ini akan segera bergeser mengingat kemampuan perangkat keras semakin meningkat.

Sebagai informasi tambahan, fungsi *SetDisplayMode* memiliki parameter yang berbeda bila interface DirectDraw yang kita gunakan bukan interface default., misal untuk IDirectDraw7 *SetDisplayMode* adalah seperti berikut:

```
function SetDisplayMode (dwWidth: DWORD; dwHeight: DWORD; dwBPP: DWORD;dwRefreshRate: DWORD; dwFlags: DWORD) : HResult; stdcall;
```

Namun karena masih tahap pengenalan kita akan menggunakan fungsi anggota interface default DirectDraw (IDirectDraw) sedangkan fungsi IDirectDraw7 dan lain-lain sementara kita lupakan.

2.4 Menciptakan Primary Surface dan Back Buffer.

Untuk menciptakan surface terlebih dahulu kita harus menyiapkan variabel bertipe *TDDSurfaceDesc* yang berguna untuk menampung informasi deskripsi surface yang akan kita buat.

Struktur *TDDSurfaceDesc* adalah sebagai berikut:

```
TDDSurfaceDesc_DX6 = packed record
  dwSize: DWORD;
  dwFlags: DWORD;
  dwHeight: DWORD;
  dwWidth: DWORD;
  case Integer of
  0: (
    dwLinearSize : DWORD;
   );
  1: (
    lPitch: LongInt;
    dwBackBufferCount: DWORD;
    case Integer of
    0: (
      dwMipMapCount: DWORD;
      dwAlphaBitDepth: DWORD;
      dwReserved: DWORD;
      lpSurface: Pointer;
      ddckCKDestOverlay: TDDColorKey;
      ddckCKDestBlt: TDDColorKey;
      ddckCKSrcOverlay: TDDColorKey;
      ddckCKSrcBlt: TDDColorKey;
```

```
      ddpfPixelFormat: TDDPixelFormat_DX6;
      ddsCaps: TDDSCaps;
     );
    1: (
      dwZBufferBitDepth: DWORD;
     );
    2: (
      dwRefreshRate: DWORD;
     );
   );
 end;

 PDDSurfaceDesc = ^TDDSurfaceDesc;
{$IFDEF DIRECTX5}
 TDDSurfaceDesc = TDDSurfaceDesc_DX5;
{$ELSE}
 TDDSurfaceDesc = TDDSurfaceDesc_DX6;
{$ENDIF}
```

Mengingat kita masih dalam tahap perkenalan, penulis hanya menjelaskan kegunaan field-field record yang sering dipakai.

Field *dwSize* harus diisi dengan ukuran struktur *TDDSurfaceDesc*. Field *dwFlags* menyimpan informasi field mana yang berisi data yang valid. Beberapa konstanta *dwFlags* yang valid adalah sebagai berikut:

DDSD_CAPS

DDSD_BACKBUFFERCOUNT

DDSD_WIDTH

DDSD_HEIGHT

DDSD_PIXELFORMAT

DDSD_LPSURFACE

DDSD_REFRESHRATE

DDSD_ZBUFFERDEPTH

DDSD_ALPHABITDEPTH

DDSD_MIPMAPCOUNT dst..

Contoh: *dwFlags*=DDSD_CAPS or DDSD_HEIGHT or DDSD_WIDTH menyatakan bahwa field-field record *ddsCaps*, *dwWidth*, *dwHeight* berisi data valid dan hal ini menyebabkan DirectDraw hanya melihat field-field ini, field-field lainnya akan diabaikan.

Salah satu field yang paling sering dipakai adalah *ddsCaps*. Field ini bertipe *TDDSCaps.* Struktur TDDSCaps adalah seperti berikut:

```
PDDSCaps = ^TDDSCaps;
TDDSCaps = packed record
  dwCaps: DWORD;
end;
```

Beberapa konstanta yang valid untuk *ddsCaps.dwCaps* adalah:

- DDSCAPS_PRIMARYSURFACE menciptakan primary surface.
- DDSCAPS_BACKBUFFER menciptakan back buffer.
- DDSCAPS_OFFSCREENPLAIN menciptakan offscreen surface.
- DDSCAPS_FLIP memberitahukan bahwa surface yang akan dibuat merupakan bagian struktur flipping.
- DDSCAPS_COMPLEX memberitahukan bahwa surface adalah bagian dari struktur flipping kompleks yakni terdiri atas lebih dari satu surface. Jika DDSCAPS_FLIP diset maka DDSCAPS_COMPLEX juga harus diset.

2.4.1 Menciptakan Primary Surface dan Back Buffer pada Mode Full Screen.

Pada mode full screen kita memiliki dua alternatif untuk menampilkan hasil render kita ke layar, yaitu dengan flipping atau blitting.

Jika kita memilih flipping maka kita membutuhkan primary surface dan back buffer yang merupakan suatu struktur flipping.

Untuk menciptakan primary surface dan back buffer yang merupakan bagian struktur flipping pada mode full screen contoh langkahnya sebagai berikut:

```
function CreatePrimarySurfaceBackBuffer(MyDirectDraw:IDirectDraw;
                        out MyPrimary,MyBack:IdirectDrawSurface):boolean;
Var SurfaceDesc:TDDSurfaceDesc;
  Hr:Hresult;
Begin
  Result:=true;
  Fillchar(SurfaceDesc,SizeOf(TsurfaceDesc),0);
  SurfaceDesc.dwSize:= SizeOf(TsurfaceDesc);
  SurfaceDesc.dwFlags:=DDSD_CAPS or DDSD_BACKBUFFERCOUNT;
```

```
  SurfaceDesc.ddsCaps.dwCaps:=DDSCAPS_PRIMARYSURFACE or DDSCAPS_FLIP or
DSCAPS_COMPLEX;
  SurfaceDesc.dwBackBufferCount:=1;
  Hr:=MyDirectDraw.CreateSurface(SurfaceDesc,MyPrimary);
  If Failed(Hr) then
  Begin
    Result:=false;
    Exit;
  End;
  SurfaceDesc.ddsCaps.dwCaps:=DDSCAPS_BACKBUFFER;
  MyPrimary.GetAttachedSurface(SurfaceDesc.ddsCaps,MyBack);
End;
```

Contoh diatas hanyalah salah satu alternatif. Sebenarnya pembaca bisa mengubah-ubah berbagai parameter deskripsi surface, namun sebagai perkenalan dengan DirectDraw, rutin diatas sudah cukup.

Berikut ini adalah penjelasan fungsi *CreatePrimarySurfaceBackBuffer*. Parameter yang dibutuhkan adalah objek DirectDraw yang telah dibuat, primary surface dan back buffer. Keluaran adalah surface yang akan menampung primary surface dan back buffer. Pertama kita kita membutuhkan variabel yang menampung deskripsi surface. Variabel ini dideklarasikan sebagai *SurfaceDesc* serta variable *hr* yang menampung status keberhasilan proses pembuatan surface yang bertipe *Hresult*.

Asumsikan bahwa proses yang akan kita kerjakan berhasil yaitu dengan mengisi

```
Result:=true;
```

Lakukan inisialisasi *SurfaceDesc* dengan mengisi semua field-nya dengan nol,

```
Fillchar(SurfaceDesc,SizeOf(TsurfaceDesc),0);
```

Selanjutnya adalah mengisi *SurfaceDesc* dengan deskripsi yang akan kita buat. Field *dwSize* kita isi dengan ukuran struktur *SurfaceDesc*. Mengisi field *dwSize* penting untuk dilakukan karena DirectDraw selalu mengambil informasi ukuran struktur dekripsi surface sebelum menciptakan surface.

```
SurfaceDesc.dwSize:= SizeOf(TsurfaceDesc);
```

Berikutnya kita tentukan field mana yang akan kita gunakan untuk mendeskripsikan surface kita dengan mengatur field *dwFlags*. Karena kita akan membuat primary surface dan back buffer untuk mode full screen yang merupakan bagian struktur flipping, maka field yang kita butuhkan adalah field *ddsCaps* dan *dwBackBufferCount*, sehingga field *dwFlags* kita isi seperti berikut:

```
SurfaceDesc.dwFlags:=DDSD_CAPS or DDSD_BACKBUFFERCOUNT;
```

Isikan deskripsi surface kita dengan mengisi field *ddsCaps* dan *dwBackBufferCount*.

```
SurfaceDesc.ddsCaps.dwCaps:=DDSCAPS_PRIMARYSURFACE or DDSCAPS_FLIP or DSCAPS_COMPLEX;

SurfaceDesc.dwBackBufferCount:=1;
```

Yang perlu mendapat perhatian adalah, jika kita mengisi *ddsCaps* dengan DDSCAPS_FLIP maka field *dwBackBufferCount* paling tidak berisi 1. Jika karena suatu hal kita lupa mengisi *dwBackBufferCount* maka pemanggilan CreateSurface akan gagal.

Walaupun DirectDraw mengijinkan kita menggunakan jumlah back buffer berapa pun yang kita mau, sebaiknya jumlah back buffer tidak terlalu banyak. Normalnya, kita cukup memakai satu back buffer, tapi jika ingin meningkatkan performa, dua back buffer bisa digunakan. Namun semakin banyak back buffer tidak selalu berarti performa semakin baik. Hal ini terjadi karena semakin banyak surface yang dibuat, jumlah memori yang dibutuhkan juga meningkat. Jika video RAM di kartu grafis tidak cukup, DirectDraw akan menciptakan surface di RAM system. Akibatnya akan terjadi penurunan performa mengingat flipping surface yang berada di RAM system ke surface yang ada di video RAM jauh lebih lambat dibandingkan flipping surface di video RAM ke surface lain yang juga di video RAM.

Setelah deskripsi surface telah disiapkan, kita ciptakan primary surface.

```
Hr:=MyDirectDraw.CreateSurface(SurfaceDesc,MyPrimary);
```

Cek status keberhasilan dengan fungsi *Failed*. Fungsi ini dideklarasikan di unit *Windows*. Jika gagal keluar dan *Result* kita isi false.

```
If Failed(Hr) then
Begin
  Result:=false;
  Exit;
End;
```

Jika berhasil lanjutkan dengan mengambil informasi back buffer. Kita isi field *ddsCaps.dwCaps* dengan DDSCAPS_BACKBUFFER. Dengan demikian kita memberitahukan DirectDraw bahwa surface berikutnya adalah back buffer.

Karena saat menciptakan primary surface, kita menggunakan DDSCAPS_FLIP dan DDSCAPS_COMPLEX maka ketika *CreateSurface* dipanggil, otomatis back buffer juga diciptakan. Oleh karena itu kita tidak perlu memanggil *CreateSurface* lagi,tapi cukup dengan menggunakan fungsi anggota *IDirectDrawSurface GetAttachedSurface* seperti berikut:

```
SurfaceDesc.ddsCaps.dwCaps:=DDSCAPS_BACKBUFFER;

MyPrimary.GetAttachedSurface(SurfaceDesc.ddsCaps,MyBack);
```

Jika kita menggunakan blitting untuk menampilkan hasil render ke layar maka primary surface dan back buffer diciptakan dengan cara sebagai berikut:

```
function CreatePrimarySurfaceBackBufferBlt(MyDirectDraw:IDirectDraw;
                                Mywidth,MyHeight:integer;
                        out MyPrimary,MyBack:IdirectDrawSurface):boolean;
Var SurfaceDesc:TDDSurfaceDesc;
   Hr:Hresult;
Begin
   Result:=true;
   Fillchar(SurfaceDesc,SizeOf(TsurfaceDesc),0);
   SurfaceDesc.dwSize:= SizeOf(TsurfaceDesc);
   SurfaceDesc.dwFlags:=DDSD_CAPS;
   SurfaceDesc.ddsCaps.dwCaps:=DDSCAPS_PRIMARYSURFACE;
   Hr:=MyDirectDraw.CreateSurface(SurfaceDesc,MyPrimary);
   If Failed(Hr) then
   Begin
     Result:=false;
     Exit;
   End;
   SurfaceDesc.dwFlags:=DDSD_CAPS or DDSD_WIDTH or DDSD_HEIGHT;
   SurfaceDesc.ddsCaps.dwCaps:=DDSCAPS_OFFSCREENPLAIN;
   SurfaceDesc.dwWidth:=MyWidth;
   SurfaceDesc.dwHeight:=MyHeight;
   Hr:=MyDirectDraw.CreateSurface(SurfaceDesc,MyBack);
   If Failed(Hr) then
   Begin
     Result:=false;
     Exit;
   End;
End;
```

Langkah-langkah yang harus ditempuh untuk menciptakan primary surface dan back buffer untuk blitting sedikit berbeda dengan langkah untuk flipping. Pada blitting tidak dikenal system struktur flipping, sehingga tiap yang akan digunakan sebagai primary surface dan back buffer harus diciptakan dengan pemanggilan fungsi anggota DirectDraw *CreateSurface*. Parameter deskripsi surface *dwFlags* diisi dengan DDSD_CAPS dan *ddsCaps.dwCaps* diisi dengan konstanta DDSCAPS_PRIMARYSURFACE untuk menciptakan primary surface dan DDSCAPS_OFFSCREENPLAIN untuk menciptakan back buffer. Yang perlu mendapat perhatian adalah perlunya kita menentukan lebar dan tinggi surface yang

akan kita gunakan sebagai back buffer, oleh karena itu parameter *dwFlags* kita isi dengan DDSD_CAPS or DDSD_WIDTH or DDSD_HEIGHT selanjutnya parameter *dwWidth* dan *dwHeight* kita isi dengan lebar dan tinggi yang kita inginkan. Misalnya kita ingin menggunakan mode 640x480 maka *dwWidth* kita isi dengan 640 dan *dwHeight*=480.

2.4.2 Menciptakan Primary Surface dan Back Buffer pada Windowed Mode.

Pada windowed mode tidak dikenal adanya flipping. Untuk merender tampilan ke layar kita harus menggunakan blitting. Hal ini berbeda dengan pada mode full screen yang mengijinkan kita menggunakan flipping atau blitting untuk melakukan proses rendering. Cara menciptakan primary surface dan back buffer untuk blitting telah dijelaskan pada sub.bab sebelumnya.

2.5 Menciptakan Offscreen Surface.

Offscreen surface biasanya digunakan untuk menampung data-data gambar atau bitmap. Cara menciptakan offscreen surface sama dengan cara menciptakan back buffer untuk blitting yakni dengan menggunakan konstanta DDCAPS_OFFSCREENPLAIN dan menentukan lebar dan tinggi surface.

2.6 Menciptakan Offscreen Surface untuk Sprite.

Untuk menciptakan surface yang akan digunakan menampung data sprite, selain membuat offscreen surface seperti langkah di atas, juga perlu di tentukan warna yang akan ditentukan sebagai warna transparan. Hal ini penting untuk dilakukan mengingat sprite sesungguhnya hanyalah bitmap biasa. Jika tidak ditentukan warna transparan maka sprite akan ditampilkan seperti dalam kotak. DirectX menggunakan konsep color key untuk menentukan warna transparan.

Untuk menentukan color key digunakan fungsi anggota DirectDrawSurface *SetColorKey*. Deklarasi fungsi ini adalah sebagai berikut:

```
function SetColorKey (dwFlags: DWORD; lpDDColorKey: PDDColorKey) : HResult; stdcall;
```

Sebelum memanggil fungsi ini terlebih dahulu harus disiapkan variable yang akan menampung informasi struktur color key. Variabel ini bertipe PDDColorKey deklarasi lengkap PDDColorKey adalah sebagai berikut:

```
PDDColorKey = ^TDDColorKey;
TDDColorKey = packed record
  dwColorSpaceLowValue: DWORD;
  dwColorSpaceHighValue: DWORD;
end;
```

Walaupun warna yang akan digunakan sebagai warna transparan bisa warna apa saja, namun biasanya warna yang dijadikan warna transparan untuk sprite adalah warna hitam R=0, G=0, B=0 atau putih R=255, G=255, B=255. Untuk menggunakan warna hitam field *dwColorSpaceLowValue* dan *dwColorSpaceHighValue* kita isi sama dengan nol. Jika menggunakan warna putih field *dwColorSpaceLowValue* dan *dwColorSpaceHighValue* kita isi sama dengan 255.

Konsekuensi yang harus diambil jika kita menggunakan warna hitam R=0, G=0, B=0 adalah kita tidak bisa menampilkan warna hitam ke layar. Untuk bagian sprite yang yang berwarna hitam dan bagian tersebut ingin kita tampilkan maka kita harus menggunakan warna hitam dengan komponen RGB tidak sama dengan nol, misal warna hitam dengan R=1, G=1, B=1. Perbedaannya hampir tidak terlihat tapi bagian sprite tersebut akan ditampilkan. Field *dwFlags* bisa kita isi dengan beberapa konstanta berikut:

- DDCKEY_SRCBLT Untuk menentukan sebuah atau beberapa warna transparan yang berfungsi sebagai warna transparan sumber untuk operasi blitting
- DDCKEY_DESTBLT Untuk menentukan sebeuah atau beberapa warna transparan yang berfungsi sebagai warna transparan tujuan untuk operasi blitting
- DDCKEY_COLORSPACE Jika struktur color key berisi bebrapa warna yang akan digunakan sebagai warna transparan.

Setelah pemanggilan fungsi *SetColorKey*, fungsi ini akan mengembalikan status keberhasilan pemanggilan yang bisa dicek dengan fungsi *Failed*.

2.7 Mengcopy Bitmap ke Surface.

Agar bitmap yang berasal dari file BMP bisa ditampilkan oleh DirectDraw maka data bitmap harus dicopy ke surface terlebih dahulu. Untuk mengcopy bitmap ada beberapa cara, yakni dengan mendapatkan device context surface atau melalui teknik penguncian surface (surface locking). Disini hanya akan dijelaskan cara pertama, karena cara ini paling mudah.

Langkah mengcopy bitmap dengan device context adalah dengan memanggil fungsi anggota DirectDrawSurface *GetDC* untuk mendapatkan device context surface.

```
function GetDC (out lphDC: HDC) : HResult; stdcall;
```

Selanjutnya melakukan proses mengcopy data dengan fungsi-fungsi GDI Windows seperti *BitBlt* atau *StretchBlt*. Setelah data bitmap dicopy device context dibebaskan dengan memanggil fungsi anggota DirectDrawSurface *ReleaseDC*.

```
function ReleaseDC (hDC: Windows.HDC) : HResult; stdcall;
```

Contoh implementasi teknik ini adalah sebagai berikut:

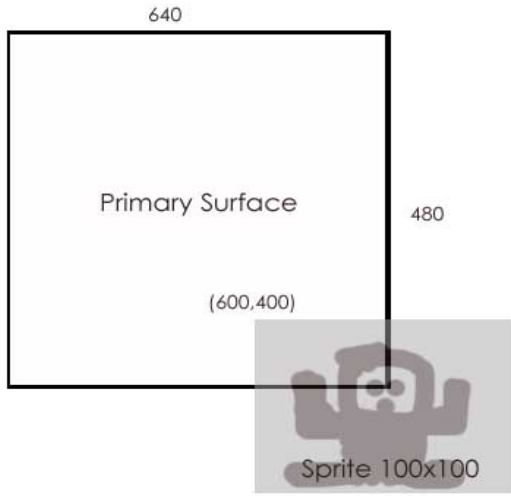
```
Procedure CopyBitmap(const Filename:string; MyDirectDrawSurface:IDirectDrawSurface);
Var bmp:Tbitmap;
    MySurfaceDC:HDC;
Begin
  Try
    Bmp:=TBitmap.Create;
    Bmp.LoadFromFile(Filename);
    MyDirectDrawSurface.GetDC(MySurfaceDC);
    BitBlt(MySurfaceDC,0,0,bmp.Width,bmp.Height,
          bmp.Canvas.Handle,0,0,SRCCOPY);
    MyDirectDrawSurface.ReleaseDC(MySurfaceDC);
  Finally
   Bmp.Free;
  End;
End;
```

Rutin diatas mengasumsikan MyDirectDrawSurface telah diciptakan dan file BMP ada. Jika tidak maka akan timbul kesalahan.

2.8 Menciptakan Clipper.

Clipper diperlukan oleh aplikasi DirectX yang berjalan pada mode Windowed dan untuk melakukan pemotongan sprite. Fungsi clipper adalah untuk membatasi area yang boleh digambar oleh DirectX sehingga DirectX tidak menggambar di luar window milik aplikasi.

Bayangkan situasi dimana kita memiliki sprite berukuran 100x100 di suatu offscreen surface,dan program kita berjalan pada mode 640x480. Kita ingin menampilkan sprite tersebut di koordinat 600,400 seperti pada gambar di bawah.



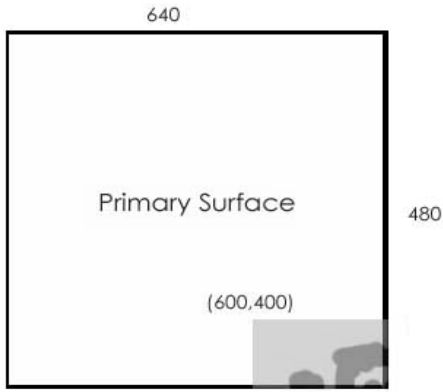
Gambar 2.1 Situasi Sprite Melebihi Batas Layar tanpa Clipping

Pada situasi seperti ini maka data sprite yang berada di luar batas primary surface akan ditulis ke memori diluar memori milik primary surface. Problem yang mungkin timbul pada situasi seperti ini adalah tampilan yang tampak dilayar menjadi kacau, atau yang paling parah adalah program kita meyebabkan komputer menjadi hang dan harus direset.

DirectDraw otomatis akan membatalkan seluruh proses penggambaran jika suatu sprite melebihi batas layar (walaupun hanya satu piksel saja yang melebihi batas layar) untuk mencegah problem-problem seperti diatas.

Untuk mengatasi hal ini kita membutuhkan clipper. Dengan clipper maka DirectDraw akan tahu dengan tepat bagaimana memotong sprite yang melebihi batas layar.

Graphic engine yang akan kita buat nanti akan menggunakan clipper untuk melakukan pemotongan sprite sehingga sprite kita akan ditampilkan dengan bagus.



Gambar 2.2 Situasi Sprite Melebihi Batas Layar Setelah Proses Clipping

Objek clipper bertipe *IDirectDrawClipper*. Untuk menciptakan clipper digunakan fungsi anggota DirectDraw *CreateClipper*.

function CreateClipper (dwFlags: DWORD; out lplpDDClipper: IDirectDrawClipper; pUnkOuter: IUnknown) : HResult; stdcall;

Parameter *dwFlags* untuk saat ini tidak terpakai sehingga bisa diisi dengan nol. *lplpDDClipper* adalah objek clipper yang akan dibuat. *pUnkOuter* tidak terpakai sehingga bisa diisi nil.

2.8.1 Clipper Untuk Aplikasi Mode Windowed.

Untuk mendapatkan informasi ukuran window digunakan fungsi anggota objek clipper *SetHWnd*.

function SetHWnd (dwFlags: DWORD; hWnd: HWND) : HResult; stdcall;

dwFlags tidak terpakai sehingga bisa diisi nol. hWnd adalah handle window aplikasi kita.

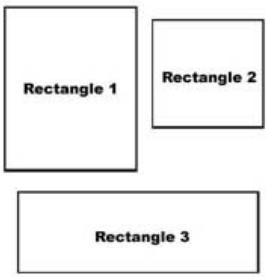
Selanjutnya untuk memasangkan clipper ke surface digunakan fungsi anggota DirectDrawSurface *SetClipper*.

function SetClipper (lpDDClipper: IDirectDrawClipper) : HResult; stdcall;

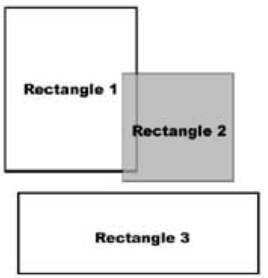
Umumnya surface yang dipasangkan clipper adalah primary surface mengingat surface inilah yang akan dilihat oleh pengguna aplikasi.

2.8.2 Clipper untuk Sprite Clipping.

Menyiapkan clipper yang akan digunakan untuk sprite clipping sedikit berbeda proses menyiapkan clipper untuk aplikasi mode windowed. Pada proses sprite clipping setelah clipper diciptakan kita harus menyiapkan *clip list*. Clip list adalah suatu daftar berisi daerah-daerah pada suatu surface yang boleh digambar oleh DirectDraw. Daerah-daerah yang boleh digambar ini tidak boleh ada yang tumpang tindih (overlapping). Gambar 2.3 menunjukkan suatu clip list yang benar karena semua daerah dalam clip list tidak ada yang saling tumpang tindih, sedangkan Gambar 2.4 menunjukkan gambar clip list yang salah karena rectangle 1 overlapping terhadap rectangle 2.



Gambar 2.3 Clip list yang benar



Gambar 2.4 Clip list yang salah (overlapping)

Untuk menyiapkan clip list kita memerlukan variabel penampung data clipping list. Variabel ini bertipe *PRgnData*. PRgnData berisi field header *rdh* bertipe *TRgnDataHeader* yang berisi informasi struktur serta array berisi area persegi empat tiap-tiap daerah yang akan diklip.

Berikut ini adalah deklarasi tipe PRgnData dan TRgnDataHeader yang ada pada unit *Windows*.

```
PRgnDataHeader = ^TRgnDataHeader;
{$EXTERNALSYM _RGNDATAHEADER}
_RGNDATAHEADER = packed record
```

```
  dwSize: DWORD;
  iType: DWORD;
  nCount: DWORD;
  nRgnSize: DWORD;
  rcBound: TRect;
end;
TRgnDataHeader = _RGNDATAHEADER;
{$EXTERNALSYM RGNDATAHEADER}
RGNDATAHEADER = _RGNDATAHEADER;

PRgnData = ^TRgnData;
{$EXTERNALSYM _RGNDATA}
_RGNDATA = record
  rdh: TRgnDataHeader;
  Buffer: array[0..0] of CHAR;
  Reserved: array[0..2] of CHAR;
end;
TRgnData = _RGNDATA;
{$EXTERNALSYM RGNDATA}
RGNDATA = _RGNDATA;
```

Struktur TRgnData

rdh

Struktur bertipe TRgnDataHeader yang menyimpan informasi tipe region, jumlah rectangle, ukuran seluruh data rectangle dan bounding rectangle.

Buffer

Array berisi rectangle-rectangle yang menyusun clip list.

Reserved

Cadangan

Struktur TRgnDataHeader

dwSize

Ukuran header dalam byte

iType

Tipe region, harus diisi RDH_RECTANGLES

nCount

Jumlah rectangle yang menyusun seluruh clip list

nRgnSize

Ukuran buffer yang diperlukan untuk menampung seluruh rectangle yang menyusun clip list.

rcBound

Bounding rectangle yang membatasi seluruh region.

Contoh bagaimana menyiapkan clip list adalah seperti dijelaskan oleh rutin *SetClipRects* dibawah ini. Fungsi *Min* dan *Max* adalah fungsi yang digunakan untuk membandingkan dua variabel. Min akan mengembalikan nilai yang lebih kecil antara dua nilai yang sedang dibandingkan, sebaliknya Max mengembalikan nilai yang lebih besar.

Untuk menyiapkan clip list langkah awal adalah menyiapkan variabel bertipe TRect yang akan menyimpan informasi koodinat minimum dan maksimum (lihat variabel *BoundsRect* pada contoh rutin untuk menyiapkan clipping list dibawah). Setelah dilakukan looping untuk mencari koordinat minimum dan maksimum maka pada akhir looping BoundsRect akan berisi rectangle yang membatasi seluruh region yang ada dalam clip list. BoundsRect ini kita perlukan untuk mengisi field *rcBound* struktur TRgnDataHeader.

Setelah mencari bounding rectangle maka kita harus mengalokasikan memori untuk RgnData yang bertipe PRgnData. Ukuran memori yang harus disiapkan adalah sebesar:

*Size=ukuran struktur TRgnDataHeader+jumlah rectangle*ukuran struktur TRect*

(lihat rutin SetClipRects pada baris *GetMem(RgnData…);*). Setelah RgnData berhasil dialokasikan maka kita isi field *rdh* milik RgnData dengan informasi yang diperlukan.

Kemudian informasi tiap-tiap rectangle diisikan ke field *Buffer* RgnData. Ada baiknya field *Reserved* diisi nol untuk menginisialisasi field ini.

Pada tahap ini maka clip list kita telah siap untuk digunakan. Untuk menggunakan clip list ini kita panggil fungsi anggota IDirectDrawClipper *SetClipList* .

function SetClipList (lpClipList: PRgnData; dwFlags: DWORD) : HResult; stdcall;

Parameter:

lpClipList

Informasi clip list.

dwFlags

Belum terpakai diisi dengan nol.

Keberhasilan SetClipList bisa diketahui dengan fungsi Failed. Setelah pemanggilan fungsi SetClipList maka clip list sudah tidak diperlukan lagi sehingga memori yang dipakai kita bebaskan lagi (lihat baris *Freemem.*.pada rutin SetClipRects).

```
function Min(const a,b:integer):integer;
var res:integer;
begin
  res:=a;
  if a<b then Res:=a else res:=b;
  Result:=res;
end;
function Max(const a,b:integer):integer;
var res:integer;
begin
  res:=a;
  if a>b then Res:=a else res:=b;
  Result:=res;
end;
procedure SetClipRects(MyClipper:IDirectDrawClipper;const Rects: array of TRect);
type
  PArrayRect = ^TArrayRect;
  TArrayRect = array[0..0] of TRect;
var
  RgnData: PRgnData;
  i: Integer;
  BoundsRect: TRect;
  hr:HResult;
begin
  BoundsRect := Rect(MaxInt, MaxInt, -MaxInt, -MaxInt);
  for i:=Low(Rects) to High(Rects) do
  begin
    with BoundsRect do
    begin
      Left := Min(Rects[i].Left, Left);
      Right := Max(Rects[i].Right, Right);
```

```
    Top := Min(Rects[i].Top, Top);
    Bottom := Max(Rects[i].Bottom, Bottom);
   end;
  end;

  GetMem(RgnData, SizeOf(TRgnDataHeader)+SizeOf(TRect)*(High(Rects)-Low(Rects)+1));
  try
   with RgnData^.rdh do
   begin
    dwSize := SizeOf(TRgnDataHeader);
    iType := RDH_RECTANGLES;
    nCount := High(Rects)-Low(Rects)+1;
    nRgnSize := nCount*SizeOf(TRect);
    rcBound := BoundsRect;
   end;
   for i:=Low(Rects) to High(Rects) do
    PArrayRect(@RgnData^.Buffer)^[i-Low(Rects)] := Rects[i];
   hr := MyClipper.SetClipList(RgnData, 0);
   if failed(hr) then
    raise Exception.Create('Gagal menciptakan clip list.');
  finally
   FreeMem(RgnData);
  end;
 end;
```

Untuk memasangkan clipper yang telah diisi clip list kita gunakan fungsi anggota IDirectDrawSurface SetClipper. Untuk sprite clipping, surface yang biasanya dipasangkan clipper adalah back buffer karena back buffer adalah tempat kita menggambar seluruh sprite, namun jika kita menggambar sprite langsung ke primary surface maka primary surface yang harus kita pasangkan clipper.

2.9 Menciptakan Palette.

Palette umumnya dipakai oleh aplikasi yang menggunakan kedalaman warna 8 bit kebawah. Untuk saat ini kedalaman warna 2 bit dan 4 bit sudah tidak dipakai dalam game-game mengingat sedikitnya warna yang dapat ditampilkan. Palette yang dibahas pada subbab ini hanya untuk kedalaman 8 bit.

Palette dalam DirectDraw mirip palette yg digunakan oleh Windows dengan pengecualian bahwa dalam DirectDraw 256 indeks palette dapat kita gunakan, hal ini berbeda dengan palette Windows. Windows hanya mengijinkan kita menggunakan 236 indeks warna karena 20 indeks merupakan indeks warna tercadang (*reserved*) bagi Windows dan digunakan untuk menampilkan warna teks, window, caption dan lain-lain. Sebenarnya jika kita mau bisa saja kita menggunakan seluruh indeks warna namun hal ini sangat tidak dianjurkan karena mungkin akan menyebabkan aplikasi lain tampak kacau warnanya.

Untuk menciptakan palette, kita menggunakan fungsi anggota objek DirectDraw CreatePalette.

```
Function CreatePalette(dwFlags:cardinal;Palette:IdirecDrawPalette;ColorTable:PaletteEntry;out
pUnkOuter:Iunknown):Hresult;StdCall;
```

Parameter *dwFlags* adalah variabel yang memberitahu DirectDraw tentang format indeks warna. Konstanta yang valid untuk *dwFlags* adalah

DDPCAPS_ALLOW256 jika ingin menggunakan seluruh warna.

DDSCAPS_8BIT format palette adalah palette 256 warna.

Parameter *Palette* adalah objek DirectDrawPalette yang akan dibuat. *ColorTable* adalah data palette entry. Parameter *UnkOuter* bisa diisi dengan nil.

Contoh menciptakan palette adalah seperti berikut:

```
Procedure CreateMyPalette(MyDirectDraw:IDirectDraw;paletteparam:array[256] of TRGB;
out  myPalette:IDirectDrawPalette);
Var I:integer;
    Pe:array[256] of PaletteEntry;
   Hr:Hresult;
Begin
   For I:=0 to 255 do
  Begin
     Pe[I].peRed:=PaletteParam[I].rgbRed;
     Pe[I].peGreen:=PaletteParam[I].rgbGreen;
     Pe[I].peBlue:=PaletteParam[I].rgbBlue;
  End;
  Hr:=MyDirectDraw.CreatePalette(DDPCAPS_8BIT,pe,MyPalette,0);
End;
```

Objek DirectDrawPalette dapat dibuat lebih dari satu jika diinginkan. Setelah objek DirectDrawPalette berhasil dibuat maka agar efek palette yang baru tersebut terlihat kita harus memilihnya\memasangkan ke primary surface dengan fungsi anggota objek DirectDrawSurface *SetPalette*

```
Function  SetPalette(destSurface:IDirectDrawSurface,palette:IDirectDrawPalette);
```

2.10 Merender Surface.

Proses rendering surface adalah proses yang paling sering dikerjakan. Pada bagian ini seluruh tampilan game disusun. Oleh karena itu pada bagian ini segala seuatu yang dikerjakan harus dibuat secepat mungkin.

Agar proses rendering menghasilkan tampilan seperti yang kita harapkan sebaiknya selalu dilakukan pengecekan surface, apakah surface itu hilang atau tidak. Surface bisa hilang oleh karena beberapa hal. Salah satunya adalah jika GDI milik Windows mengambil alih kontrol perangkat keras grafik.

2.10.1 Mengecek Surface yang Hilang.

Untuk mengecek surface yang hilang kita menggunakan fungsi *isLost* yang merupakan fumgsi anggota DirectDrawSurface. Deklarasi fungsi ini adalah sebagai berikut:

```
Function isLost:Hresult;stdcall;
```

Jika keluaran fungsi ini adalah DDERR_SURFACELOST maka surface telah hilang. Untuk mendapatkan kembali surface yang hilang, kita harus merestore ulang surface dengan memanggil fungsi anggota objek DirectDrawSurface yakni *_Restore.*

Berikut ini adalah contoh rutin untuk melakukan pengecekan surface dan merestore surface jika surface hilang.

```
Procedure CheckSurface(MySurface:IdirectDrawSurface);
Var hr:Hresult;
Begin
   Hr:=MySurface.isLost;
  Case Hr of
    DDERR_SURFACELOST:MySurface._Restore;
  End;
End;
```

Rutin di atas hanya akan merestore satu surface. Oleh karena itu rutin di atas harus dipanggil sebanyak jumlah surface yang kita pakai, agar semua surface yang hilang dapat direstore dengan sukses.

2.10.2 Merender Offscreen Surface.

Untuk melakukan rendering offscreen surface kita gunakan fungsi *Blt* atau *BltFast* yang merupakan fungsi anggota Objek DirectDrawSurface dan berfungsi untuk melakukan blitting. Deklarasi fungsi Blt adalah sebagai berikut:

```
Function Blt (lpDestRect: PRect; lpDDSrcSurface: IDirectDrawSurface; lpSrcRect: PRect;
        dwFlags: DWORD; lpDDBltFx: PDDBltFX) : HResult; stdcall;
```

Parameter-parameter fungsi Blt:

lpDestRect

Rectangle tujuan dimana pengkopian akan dilakukan

lpDDSrcSurface

Surface yang akan dikopi datanya.

lpSrcRect

Rectangle sumber

dwFlags

Flag

Konstanta yang valid untuk flag adalah seperti di table berikut:

Tabel Konstanta Flag Fungsi Blt

DDBLT_ALPHADEST	Memberitahukan DirectDraw agar menggunakan informasi alpha yang ada pada pixel format atau surface alpha channel yang dipasangkan pada surface tujuan sebagai alpha channel untuk proses Blt.
DDBLT_ALPHADESTCONSTOVERRIDE	Memberitahukan DirectDraw agar menggunakan informasi alpha yang ada pada field *dwConstAlphaDest* pada struktur TDDBltFX sebagai alpha channel surface tujuan untuk proses Blt.
DDBLT_ALPHADESTNEG	Hampir sama dengan DDBLT_ALPHADEST kecuali bahwa semakin besar nilai alpha maka surface tujuan semakin transparan (0=opaque).
DDBLT_ALPHADESTSURFACEOVERRIDE	Memberitahukan DirectDraw agar menggunakan informasi alpha yang ada pada field *lpDDSAlphaDest* pada struktur TDDBltFX sebagai alpha channel surface tujuan untuk proses Blt.
DDBLT_ALPHAEDGEBLEND	Memberitahukan DirectDraw agar menggunakan informasi alpha yang ada pada field

	dwAlphaEdgeBlend pada struktur TDDBltFX sebagai alpha channel pada tepi gambar yang berbatasan dengan color key untuk proses Blt.
DDBLT_ALPHASRC	Memberitahukan DirectDraw agar menggunakan informasi alpha yang ada pada pixel format atau surface alpha channel yang dipasangkan pada surface sumber sebagai alpha channel untuk proses Blt.
DDBLT_ALPHASRCCONSTOVERRIDE	Memberitahukan DirectDraw agar menggunakan informasi alpha yang ada pada field *dwConstAlphaSrc* pada struktur TDDBltFX sebagai alpha channel surface tujuan untuk proses Blt.
DDBLT_ALPHASRCNEG	Hampir sama dengan DDBLT_ALPHADEST kecuali bahwa semakin besar nilai alpha maka surface sumber semakin transparan (0=opaque).
DDBLT_ALPHASRCSURFACEOVERRIDE	Memberitahukan DirectDraw agar menggunakan informasi alpha yang ada pada field *lpDDSAlphaSrc* pada struktur TDDBltFX sebagai alpha channel surface sumber untuk proses Blt.
DDBLT_ASYNC	Melakukan blit secara ansinkron.
DDBLT_COLORFILL	Memberitahukan Blt untuk menggunakan field *dwFillColor* pada struktur TDDBltFX untuk mengisi rectangle tujuan pada surface tujuan.
DDBLT_DDFX	Memberitahukan Blt untuk menggunakan efek blit yang disimpan di field *dwDDFX* pada struktur TDDBltFX.
DDBLT_WAIT	Blt akan gagal jika surface sedang sibuk. Dengan konstanta ini kita memerintahkan Blt untuk

	menunggu sampai surface tersebut tersedia, baru kemudian melakukan blit.
DDBLT_KEYSRC	Memberitahukan agar Blt menggunakan color key surface sumber sebagai warna transparan. Konstanta ini dipakai bila kita hendak merender sprite.
DDBLT_KEYSRCOVERRIDE	Memberitahukan kepada Blt agar menggunakan color key yang ada pada field *dckSrcColorKey* pada struktur TDDBltFX dan mengabaikan color key surface sumber ketika menampilkan sprite.
DDBLT_KEYDEST	Memberitahukan agar Blt menggunakan color key surface tujuan sebagai warna transparan. Konstanta ini dipakai bila kita hendak merender sprite.
DDBLT_KEYDESTOVERRIDE	Memberitahukan kepada Blt agar menggunakan color key yang ada pada field *dckDestColorKey* pada struktur TDDBltFX dan mengabaikan color key surface tujuan ketika menampilkan sprite.

lpDDBltFx

Pointer yang menunjuk ke struktur *TDDBltFX* yang digunakan untuk menambahkan efek blit. Isikan nil jika tidak ingin menggunakan efek blit. Mengenai efek blit dan struktur TDDBltFX akan dijelaskan pada sub.bab 2.10.4.

Berikut ini adalah deklarasi fungsi BltFast.

```
Function BltFast (dwX: DWORD; dwY: DWORD; lpDDSrcSurface: IDirectDrawSurface;
    lpSrcRect: PRect; dwTrans: DWORD) : HResult; stdcall;
```

Parameter-parameter fungsi BltFast:

dwX

Posisi koordinat X pada surface tujuan dimana surface tujuan akan digambar.

dwY

Posisi koordinat Y pada surface tujuan dimana surface tujuan akan digambar.

lpDDSrcSurface

Surface yang akan dikopi datanya.

lpSrcRect

Rectangle sumber

dwTrans

Flag untuk BltFast

Konstanta yang valid untuk flag adalah seperti tabel berikut:

Tabel Konstanta Flag Fungsi BltFast

DDBLTFAST_SRCCOLORKEY	Memberitahukan BltFast agar menggunakan informasi color key yang ada pada surface sumber.
DDBLTFAST_DESTCOLORKEY	Memberitahukan BltFast agar menggunakan informasi color key yang ada pada surface tujuan.
DDBLTFAST_NOCOLORKEY	Memberitahukan BltFast agar mengabaikan color key.
DDBLTFAST_WAIT	Memerintahkan BltFast untuk menunggu sampai proses blit bisa dikerjakan.
DDBLTFAST_DONOTWAIT	Kebalikan dari DDBLTFAST_WAIT.

Perbedaan antara *Blt* dan *BltFast* terletak pada fasilitas stretching dan efek blit. Pada Blt jika *lpDestRect* dan *lpSrcRect* tidak sama ukurannya maka otomatis Blt akan melakukan stretching source surface sehingga dapat ditampilkan seluruhnya pada destination surface. Dengan Blt kita dapat menambahkan efek-efek tertentu dengan mengisi parameter *lpDDBltFx*.

Pada BltFast tidak ada fasilitas stretching, sehingga jika *lpSrcRect* dan *lpDestRect* tidak sama ukurannya maka akan terjadi pemotongan (clipping) jika *lpSrcRect* lebih besar dari *lpDestRect*. Kelebihan utama BltFast dibandingkan Blt adalah pada kecepatannya, karena pada BltFast tidak dilakukan proses stretching. Efek blit yang tersedia pada BltFast hanyalah blit transparan

Untuk kartu grafis keluaran baru, umumnya mendukung hardware blitter. Dengan blit menggunakan hardware, Blt sama cepatnya bila dibandingkan dengan BltFast, sedangkan bila tidak menggunakan hardware blitter kecepatan BltFast sekitar 10% lebih cepat dari Blt.

Kekurangan lain BltFast adalah tidak dapat melakukan blit ke surface yang dipasangkan clipper. Pada graphic engine yang akan kita buat, semua proses rendering surface akan dikerjakan dengan Blt

2.10.3 Blit atau Flip Offscreen Surface ke Primary Surface.

Jika primary surface adalah struktur page flipping dan aplikasi berjalan pada mode full screen, untuk melakukan rendering ke primary surface kita bisa menggunakan blit atau flip. Jika aplikasi berjalan pada mode windowed maka rendering ke primary surface menggunakan blit.

Proses blit telah dijelaskan di subbab sebelumnya sehingga tidak perlu dijelaskan lagi. Berikut ini adalah deklarasi fungsi *Flip*. Fungsi ini adalah fungsi anggota objek DirectDrawSurface dan surface yang memanggil ini haruslah primary surface.

Function Flip(lpDDSurfaceTargetOverride: IDirectDrawSurface; dwFlags: DWORD) : HResult; stdcall;

Parameter

lpDDSurfaceTargetOverride

Surface yang akan di gunakan untuk menampilkan proses rendering. Isikan sama dengan nil agar proses rendering ditampilkan di primary surface.

Dengan memanggil fungsi ini maka primary surface akan ditampilkan ke layar sehingga bisa dilihat oleh pengguna aplikasi.

dwFlags

Flag berfungsi untuk mengatur proses flipping. Konstanta yang valid untuk *dwFlags* adalah sebagai berikut:

DDFLIP_WAIT

Memerintahkan Flip untuk menunggu jika surface sedang sibuk dan baru melakukan flipping jika surface telah tersedia untuk diakses.

Selain DDFLIP_WAIT ada konstanta-kontanta lain untuk dwFlags seperti DDFLIP_DONOTWAIT, DDFLIP_NOVSNYC, DDFLIP_STEREO dan lain-lain. Untuk sementara belum dibahas.

2.10.4 Efek Blit.

Proses blit dengan Blt memiliki kelebihan yakni kita dapat menambahkan efek-efek yang menarik ketika melakukan proses blit.

Untuk dapat menambahkan efek blit kita harus menyiapkan suatu variabel bertipe TDDBltFX.

2.10.4.1 Struktur TDDBltFX.

Berikut ini adalah deklarasi struktur ini yang terdapat pada unit DirectDraw.Pas,

```
PDDBltFX = ^TDDBltFX;
TDDBltFX = packed record
  dwSize                : DWORD;
```

```
dwDDFX                  : DWORD;
dwROP                   : DWORD;
dwDDROP                 : DWORD;
dwRotationAngle         : DWORD;
dwZBufferOpCode         : DWORD;
dwZBufferLow            : DWORD;
dwZBufferHigh           : DWORD;
dwZBufferBaseDest       : DWORD;
dwZDestConstBitDepth    : DWORD;
case integer of
0: (
 dwZDestConst           : DWORD
);
1: (
 lpDDSZBufferDest       : PDirectDrawSurface;
 dwZSrcConstBitDepth    : DWORD;
 case integer of
 0: (
  dwZSrcConst           : DWORD;
 );
 1: (
  lpDDSZBufferSrc       : PDirectDrawSurface;
  dwAlphaEdgeBlendBitDepth : DWORD;
  dwAlphaEdgeBlend      : DWORD;
  dwReserved            : DWORD;
  dwAlphaDestConstBitDepth : DWORD;
  case integer of
  0: (
   dwAlphaDestConst     : DWORD;
  );
  1: (
   lpDDSAlphaDest       : PDirectDrawSurface;
   dwAlphaSrcConstBitDepth : DWORD;
   case integer of
   0: (
```

```
        dwAlphaSrcConst      : DWORD;
       );
       1: (
        lpDDSAlphaSrc        : PDirectDrawSurface;
        case integer of
        0: (
          dwFillColor        : DWORD;
        );
        1: (
          dwFillDepth        : DWORD;
        );
        2: (
          dwFillPixel        : DWORD;
        );
        3: (
          lpDDSPattern       : PDirectDrawSurface;
          ddckDestColorkey   : TDDColorKey;
          ddckSrcColorkey    : TDDColorKey;
        )
      )
    )
  )
 )
 end;
```

Field-field pada struktur TDDBltFX

dwSize

Ukuran struktur dalam byte. Field ini harus diisi terlebih dahulu sebelum digunakan.

dwDDFX

Tipe operasi. Konstanta berikut adalah nilai-nilai yang valid untuk dwDDFX.

Tabel Konstanta dwDDFX

DDBLTFX_ARITHSTRETCHY	Melakukan stretching aritmetik sepanjang sumbu Y.
DDBLTFX_MIRRORLEFTRIGHT	Pencerminan dari kiri ke kanan.
DDBLTFX_MIRRORUPDOWN	Pencerminan dari atas ke bawah.

DDBLTFX_NOTEARING	Proses blit menunggu proses vertikal retrace selesai, baru melakukan blit guna menghindari adanya tearing.
DDBLTFX_ROTATE180	Memutar surface 180 derajat searah jarum jam.
DDBLTFX_ROTATE270	Memutar surface 270 derajat searah jarum jam.
DDBLTFX_ROTATE90	Memutar surface 90 derajat searah jarum jam.
DDBLTFX_ZBUFFERBASEDEST	Menambahkan field *dwZBufferBaseDest* dengan nilai z tujuan sebelum membandingkan dengan nilai z tujuan.
DDBLTFX_ZBUFFERRANGE	Menggunakan field *dwZBufferLow* dan field *dwZBufferHigh* sebagai nilai jangkauan untuk menentukan batas terhadap bit yang dikopi dari surface tujuan.

dwROP

Operasi raster Win32. Untuk mengecek operasi raster apa yang tersedia digunakan fungsi *GetCaps* milik interface IDirectDraw.

dwDDROP

Operasi raster DirectDraw.

dwRotationAngle

Sudut rotasi yang diinginkan.

dwZBufferOpCode

z-buffer pembanding.

dwZBufferLow

Batas bawah z-buffer.

dwZBufferHigh

Batas atas z-buffer.

dwZBufferBaseDest

Nilai basis tujuan z-buffer.

dwZDestConstBitDepth

Bit depth konstanta z tujuan.

dwZDestConst

Konstanta yang digunakan sebagai z-buffer tujuan.

lpDDSZBufferDest

Surface yang digunakan sebagai z-buffer tujuan.

dwZSrcConstBitDepth

Bit depth konstanta z sumber.

dwZSrcConst

Konstanta yang digunakan sebagai z-buffer dest.

lpDDSZBufferSrc

Surface yang digunakan sebagai z-buffer sumber.

dwAlphaEdgeBlendBitDepth

Bit depth konstanta untuk alpha edge blend.

dwAlphaEdgeBlend

Konstanta alpha yang digunakan untuk edge blending.

dwReserved

Cadangan.

dwAlphaDestConstBitDepth

Bit depth konstanta alpha tujuan.

dwAlphaDestConst

Konstanta yang digunakan sebagai alpha channel tujuan.

lpDDSAlphaDest

Surface yang digunakan sebagai alpha channel tujuan.

dwAlphaSrcConstBitDepth

Bit depth konstanta alpha sumber.

dwAlphaSrcConst

Konstanta yang digunakan sebagai alpha channel sumber.

lpDDSAlphaSrc

Surface yang digunakan sebagai alpha channel sumber.

dwFillColor

Warna yang digunakan untuk mengisi sebuah surface ketika digunakan flag DDBLT_COLORFILL pada fungsi Blt. Nilai ini harus nilai pixel yang sesuai dengan format pixel surface tujuan. Untuk surface yang menggunakan palette,

nilainya adalah indeks palette, untuk surface 16-bit RGB, nilainya adalah warna pixel dalam format 16 bit.

dwFillDepth

Nilai kedalaman z-buffer.

dwFillPixel

Nilai dari pixel untuk RGBA atau RGBZ yang digunakan untuk mengisi sebuah surface. Aplikasi yang mengisi surface dengan RGBZ harus menggunakan dfield ini, bukan *dwFillColor* atau *dwFillDepth*.

lpDDSPattern

Surface yang digunakan sebagai pola. Pola dapat digunakan untuk mengkombinasikan surface sumber dan surface tujuan.

ddckDestColorkey

Jika fungsi Blt menggunakan flag DDBLT_KEYDESTOVERRIDE, maka informasi nilai color key tujuan akan diambil dari field ini, sedangkan informasi color key yang ada di surface akan diabaikan.

ddckSrcColorkey

Jika fungsi Blt menggunakan flag DDBLT_KEYSRCOVERRIDE, maka informasi nilai color key sumber akan diambil dari field ini, sedangkan informasi color key yang ada di surface akan diabaikan.

Karena kompleksnya struktur TDDBltFX serta kita masih dalam tahap perkenalan dengan DirectDraw, maka efek yang akan kita bahas hanya efek-efek sederhana yang akan kita perlukan untuk membuat game, yakni mencerminkan, merotasi dan mengisi surface dengan warna.

2.10.4.2 Mencerminkan Surface Dari Kiri ke Kanan.

Jika anda pernah memainkan game-game 2D side scrolling seperti Sonic atau Mario Bros anda tentu akan melihat jika karakter dalam game bisa menghadap ke kiri dan ke kanan. Apakah kita harus membuat dua gambar karakter untuk menampilkan sprite yang menghadap kiri dan ke kanan? Dengan DirectDraw hal ini tidak perlu, karena DirectDraw menyediakan kemampuan mencerminkan surface dari kiri ke kanan, dengan demikian akan menghemat memori.

Berikut ini adalah contoh rutin untuk melakukan pencerminan dari kiri ke kanan.

```
procedure ShowMirrorLeftRight(BackSurface,FSpriteBuffer:IDirectDrawSurface;
                              X,Y:integer;FFrameRect:TRect);
var aRect:TRect;
   aWidth,aHeight:integer;
  bltFX:TDDBltFX;
```

```
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
    FSpriteBuffer._Restore;
  If BackSurface.IsLost=DDERR_SURFACELOST then
    BackSurface._Restore;
  aWidth:=FFrameRect.Right-FFrameRect.Left;
  aHeight:=FframeRect.Bottom-FframeRect.Top;
  aRect:=Rect(X,Y,X+AWidth,Y+aHeight);
  FillChar(BltFX,sizeOf(TDDBltFX,0);
  BltFX.dwSize:=SizeOf(TDDBltFX);
  BltFX.dwDDFX:=DDBLTFX_MIRRORLEFTRIGHT;
  BackSurface.Blt(@aRect,FSpriteBuffer,
                    @FFrameRect,
                  DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                  @BltFX);
end;
```

Rutin ini mirip contoh rutin untuk melakukan proses rendering dengan Blt. Perbedaannya adalah terdapat tambahan variabel lokal *BltFX* bertipe TDDBltFX. Sebelum memanggil fungsi Blt, BltFX kita siapkan dengan mengisi strukturnya dengan nol,

```
FillChar(BltFX,sizeOf(TDDBltFX,0);
```

kemudian field dwSize milik BltFX kita inisialisasi dengan ukuran struktur dalam byte dan kita isi field dwDDFX kita isi DDBLTFX_MIRRORLEFTRIGHT

```
BltFX.dwSize:=SizeOf(TDDBltFX);
BltFX.dwDDFX:=DDBLTFX_MIRRORLEFTRIGHT;
```

Selanjutnya fungsi Blt kita panggil.

```
BackSurface.Blt(@aRect,FSpriteBuffer,
                  @FFrameRect,
                DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                @BltFX);
```

Yang perlu mendapat perhatian adalah parameter Blt yang ditulis tebal. Agar fungsi Blt mengerti field dwDDFX milik BltFX berisi informasi efek yang ingin kita gunakan maka kita harus menyertakan flag DDBLTFX_DDFX saat memanggil Blt, selain itu juga kita harus memberitahukan Blt dimana alamat struktur yang menyimpan informasi efek blit yakni dengan mengisikan @BltFX ke parameter terakhir fungsi Blt.

Jika komputer anda tidak mendukung pencerminan dari kiri ke kanan, maka surface tidak akan digambar jika anda menggunakan fungsi pencerminan dari kiri ke kanan. Untuk mengetahui apakah pencerminan dari kiri ke kanan ini tersedia, kita dapat menggunakan fungsi GetCaps milik interface IDirectDraw (lihat sub.bab 2.11 Mendapatkan Informasi Kemampuan yang Tersedia).

2.10.4.3 Mencerminkan Surface Dari Atas ke Bawah.

Pencerminan atas bawah mirip dengan proses mencerminkan kiri kanan. Apa yang diubah hanyalah field dwDDFX

Berikut ini adalah contoh rutin untuk melakukan pencerminan dari kiri ke kanan.

```
procedure ShowMirrorUpDown(BackSurface,FSpriteBuffer:IDirectDrawSurface;
                                    X,Y:integer;FFrameRect:TRect);
var aRect:TRect;
   aWidth,aHeight:integer;
  bltFX:TDDBltFX;
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
     FSpriteBuffer._Restore;
  If BackSurface.IsLost=DDERR_SURFACELOST then
     BackSurface._Restore;
  aWidth:=FFrameRect.Right-FFrameRect.Left;
  aHeight:=FframeRect.Bottom-FframeRect.Top;
  aRect:=Rect(X,Y,X+AWidth,Y+aHeight);
  FillChar(BltFX,sizeOf(TDDBltFX,0);
  BltFX.dwSize:=SizeOf(TDDBltFX);
  BltFX.dwDDFX:=DDBLTFX_MIRRORUPDOWN;
  BackSurface.Blt(@aRect,FSpriteBuffer,
                        @FFrameRect,
                     DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                     @BltFX);
end;
```

Sebelum memanggil fungsi Blt, BltFX kita siapkan dengan mengisi strukturnya dengan nol,

```
   FillChar(BltFX,sizeOf(TDDBltFX,0);
```

kemudian field dwSize milik BltFX kita inisialisasi dengan ukuran struktur dalam byte dan kita isi field dwDDFX kita isi DDBLTFX_MIRRORUPDOWN

```
BltFX.dwSize:=SizeOf(TDDBltFX);
BltFX.dwDDFX:=DDBLTFX_MIRRORLEFTRIGHT;
```

Selanjutnya fungsi Blt kita panggil.

```
BackSurface.Blt(@aRect,FSpriteBuffer,
                @FFrameRect,
                DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                @BltFX);
```

Selanjutnya kita panggil Blt seperti pada proses pencerminan kiri kanan.

Jika komputer anda tidak mendukung pencerminan dari atas ke bawah, maka surface tidak akan digambar jika anda menggunakan fungsi pencerminan dari atas ke bawah. Untuk mengetahui apakah pencerminan dari atas ke bawah ini tersedia, kita dapat menggunakan fungsi GetCaps milik interface IDirectDraw (lihat sub.bab 2.11 Mendapatkan Informasi Kemampuan yang Tersedia).

2.10.4.4 Memutar Surface 90 Derajat Searah Jarum Jam.

Fungsi ini sangat membantu jika kita ingin melakukan animasi rotasi, dengan fungsi ini kita cukup menyiapkan satu gambar saja untuk menampilkan animasi. Jika kita tidak menggunakan fungsi ini maka kita harus menyiapkan gambar-gambar animasi untuk tiap-tiap perubahan sudut.

Berikut ini adalah contoh rutin untuk melakukan rotasi 90 derajat searah jarum jam.

```
procedure ShowRotate90(BackSurface,FSpriteBuffer:IDirectDrawSurface;
                       X,Y:integer;FFrameRect:TRect);
var aRect:TRect;
   aWidth,aHeight:integer;
  bltFX:TDDBltFX;
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
    FSpriteBuffer._Restore;
  If BackSurface.IsLost=DDERR_SURFACELOST then
    BackSurface._Restore;
  aWidth:=FFrameRect.Right-FFrameRect.Left;
  aHeight:=FframeRect.Bottom-FframeRect.Top;
  aRect:=Rect(X,Y,X+AWidth,Y+aHeight);
  FillChar(BltFX,sizeOf(TDDBltFX,0);
  BltFX.dwSize:=SizeOf(TDDBltFX);
  BltFX.dwDDFX:=DDBLTFX_ROTATE90;
```

```
BackSurface.Blt(@aRect,FSpriteBuffer,
                  @FFrameRect,
                DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                @BltFX);
end;
```

Rutin ini mirip contoh-contoh rutin sebelumnya untuk melakukan proses rendering dengan Blt. Perbedaannya adalah terdapat tambahan variabel lokal *BltFX* bertipe TDDBltFX. Sebelum memanggil fungsi Blt, BltFX kita siapkan dengan mengisi strukturnya dengan nol,

```
FillChar(BltFX,sizeOf(TDDBltFX,0);
```

kemudian field dwSize milik BltFX kita inisialisasi dengan ukuran struktur dalam byte dan kita isi field dwDDFX kita isi DDBLTFX_ROTATE90

```
BltFX.dwSize:=SizeOf(TDDBltFX);
BltFX.dwDDFX:=DDBLTFX_ROTATE90;
```

Selanjutnya fungsi Blt kita panggil.

```
BackSurface.Blt(@aRect,FSpriteBuffer,
                  @FFrameRect,
                DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                @BltFX);
```

Yang perlu mendapat perhatian adalah parameter Blt yang ditulis tebal. Agar fungsi Blt mengerti field dwDDFX milik BltFX berisi informasi efek yang ingin kita gunakan maka kita harus menyertakan flag DDBLTFX_DDFX saat memanggil Blt, selain itu juga kita harus memberitahukan Blt dimana alamat struktur yang menyimpan informasi efek blit yakni dengan mengisikan @BltFX ke parameter terakhir fungsi Blt.

Jika komputer anda tidak mendukung rotasi 90, maka surface tidak akan digambar jika anda menggunakan fungsi rotasi 90. Untuk mengetahui apakah rotasi 90 ini tersedia, kita dapat menggunakan fungsi GetCaps milik interface IDirectDraw (lihat sub.bab 2.11 Mendapatkan Informasi Kemampuan yang Tersedia).

2.10.4.5 Memutar Surface 180 Derajat Searah Jarum Jam.

Fungsi ini sangat membantu jika kita ingin melakukan animasi rotasi, dengan fungsi ini kita cukup menyiapkan satu gambar saja untuk menampilkan animasi. Jika kita tidak menggunakan fungsi ini maka kita harus menyiapkan gambar-gambar animasi untuk tiap-tiap perubahan sudut.

Berikut ini adalah contoh rutin untuk melakukan rotasi 180 derajat searah jarum jam.

```
procedure ShowRotate180(BackSurface,FSpriteBuffer:IDirectDrawSurface;
                        X,Y:integer;FFrameRect:TRect);
```

```
var aRect:TRect;
   aWidth,aHeight:integer;
  bltFX:TDDBltFX;
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
     FSpriteBuffer._Restore;
  If BackSurface.IsLost=DDERR_SURFACELOST then
     BackSurface._Restore;
  aWidth:=FFrameRect.Right-FFrameRect.Left;
  aHeight:=FframeRect.Bottom-FframeRect.Top;
  aRect:=Rect(X,Y,X+AWidth,Y+aHeight);
  FillChar(BltFX,sizeOf(TDDBltFX,0);
  BltFX.dwSize:=SizeOf(TDDBltFX);
  BltFX.dwDDFX:=DDBLTFX_ROTATE180;
  BackSurface.Blt(@aRect,FSpriteBuffer,
                        @FFrameRect,
                     DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                     @BltFX);
end;
```

Rutin ini mirip contoh-contoh rutin sebelumnya untuk melakukan proses rendering dengan Blt. Perbedaannya adalah terdapat tambahan variabel lokal *BltFX* bertipe TDDBltFX. Sebelum memanggil fungsi Blt, BltFX kita siapkan dengan mengisi strukturnya dengan nol,

```
FillChar(BltFX,sizeOf(TDDBltFX,0);
```

kemudian field dwSize milik BltFX kita inisialisasi dengan ukuran struktur dalam byte dan kita isi field dwDDFX kita isi DDBLTFX_ROTATE180

```
BltFX.dwSize:=SizeOf(TDDBltFX);
BltFX.dwDDFX:=DDBLTFX_ROTATE180;
```

Selanjutnya fungsi Blt kita panggil.

```
BackSurface.Blt(@aRect,FSpriteBuffer,
                        @FFrameRect,
                     DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                     @BltFX);
```

Yang perlu mendapat perhatian adalah parameter Blt yang ditulis tebal. Agar fungsi Blt mengerti field dwDDFX milik BltFX berisi informasi efek yang ingin kita gunakan maka kita harus menyertakan flag DDBLTFX_DDFX saat memanggil Blt,

selain itu juga kita harus memberitahukan Blt dimana alamat struktur yang menyimpan informasi efek blit yakni dengan mengisikan @BltFX ke parameter terakhir fungsi Blt.

Jika komputer anda tidak mendukung rotasi 180, maka surface tidak akan digambar jika anda menggunakan fungsi rotasi 180. Untuk mengetahui apakah rotasi 180 ini tersedia, kita dapat menggunakan fungsi GetCaps milik interface IDirectDraw (lihat sub.bab 2.11 Mendapatkan Informasi Kemampuan yang Tersedia).

2.10.4.6 Memutar Surface 270 Derajat Searah Jarum Jam.

Fungsi ini sangat membantu jika kita ingin melakukan animasi rotasi, dengan fungsi ini kita cukup menyiapkan satu gambar saja untuk menampilkan animasi. Jika kita tidak menggunakan fungsi ini maka kita harus menyiapkan gambar-gambar animasi untuk tiap-tiap perubahan sudut.

Berikut ini adalah contoh rutin untuk melakukan rotasi 270 derajat searah jarum jam.

```
procedure ShowRotate270(BackSurface,FSpriteBuffer:IDirectDrawSurface;
                                    X,Y:integer;FFrameRect:TRect);
var aRect:TRect;
   aWidth,aHeight:integer;
  bltFX:TDDBltFX;
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
    FSpriteBuffer._Restore;
  If BackSurface.IsLost=DDERR_SURFACELOST then
    BackSurface._Restore;
  aWidth:=FFrameRect.Right-FFrameRect.Left;
  aHeight:=FframeRect.Bottom-FframeRect.Top;
  aRect:=Rect(X,Y,X+AWidth,Y+aHeight);
  FillChar(BltFX,sizeOf(TDDBltFX),0);
  BltFX.dwSize:=SizeOf(TDDBltFX);
  BltFX.dwDDFX:=DDBLTFX_ROTATE270;
  BackSurface.Blt(@aRect,FSpriteBuffer,
                    @FFrameRect,
                  DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                  @BltFX);
end;
```

Rutin ini mirip contoh-contoh rutin sebelumnya untuk melakukan proses rendering dengan Blt. Perbedaannya adalah terdapat tambahan variabel lokal *BltFX* bertipe TDDBltFX. Sebelum memanggil fungsi Blt, BltFX kita siapkan dengan mengisi strukturnya dengan nol,

```
FillChar(BltFX,sizeOf(TDDBltFX,0);
```

kemudian field dwSize milik BltFX kita inisialisasi dengan ukuran struktur dalam byte dan kita isi field dwDDFX kita isi DDBLTFX_ROTATE270

```
BltFX.dwSize:=SizeOf(TDDBltFX);
BltFX.dwDDFX:=DDBLTFX_ROTATE270;
```

Selanjutnya fungsi Blt kita panggil.

```
BackSurface.Blt(@aRect,FSpriteBuffer,
                @FFrameRect,
               DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
               @BltFX);
```

Yang perlu mendapat perhatian adalah parameter Blt yang ditulis tebal. Agar fungsi Blt mengerti field dwDDFX milik BltFX berisi informasi efek yang ingin kita gunakan maka kita harus menyertakan flag DDBLTFX_DDFX saat memanggil Blt, selain itu juga kita harus memberitahukan Blt dimana alamat struktur yang menyimpan informasi efek blit yakni dengan mengisikan @BltFX ke parameter terakhir fungsi Blt.

Jika komputer anda tidak mendukung rotasi 270, maka surface tidak akan digambar jika anda menggunakan fungsi rotasi 270. Untuk mengetahui apakah rotasi 270 ini tersedia, kita dapat menggunakan fungsi GetCaps milik interface IDirectDraw (lihat sub.bab 2.11 Mendapatkan Informasi Kemampuan yang Tersedia).

2.10.4.7 Memutar Surface Dengan Sudut Sembarang.

Fungsi ini sangat membantu jika kita ingin melakukan animasi rotasi, dengan fungsi ini kita cukup menyiapkan satu gambar saja untuk menampilkan animasi. Jika kita tidak menggunakan fungsi ini maka kita harus menyiapkan gambar-gambar animasi untuk tiap-tiap perubahan sudut.

Berikut ini adalah contoh rutin untuk melakukan rotasi dengan sudut sembarang.

```
procedure ShowRotate(BackSurface,FSpriteBuffer:IDirectDrawSurface;
                                X,Y:integer;FFrameRect:Trect;const angle:integer);
var aRect:TRect;
   aWidth,aHeight:integer;
  bltFX:TDDBltFX;
begin
```

```
If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
  FSpriteBuffer._Restore;
If BackSurface.IsLost=DDERR_SURFACELOST then
  BackSurface._Restore;
aWidth:=FFrameRect.Right-FFrameRect.Left;
aHeight:=FframeRect.Bottom-FframeRect.Top;
aRect:=Rect(X,Y,X+AWidth,Y+aHeight);
FillChar(BltFX,sizeOf(TDDBltFX,0);
BltFX.dwSize:=SizeOf(TDDBltFX);
BltFX.dwRotationAngle:=Angle;
BackSurface.Blt(@aRect,FSpriteBuffer,
                  @FFrameRect,
                DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_ROTATIONANGLE,
                @BltFX);
end;
```

Rutin ini mirip contoh-contoh rutin sebelumnya untuk melakukan proses rendering dengan Blt. Perbedaannya adalah terdapat tambahan variabel lokal *BltFX* bertipe TDDBltFX. Sebelum memanggil fungsi Blt, BltFX kita siapkan dengan mengisi strukturnya dengan nol,

```
FillChar(BltFX,sizeOf(TDDBltFX,0);
```

kemudian field dwSize milik BltFX kita inisialisasi dengan ukuran struktur dalam byte dan kita isi field dwRotationAngle kita isi dengan sudut rotasi yang kita inginkan.

```
BltFX.dwSize:=SizeOf(TDDBltFX);
BltFX.dwRotationAngle:=Angle;
```

Selanjutnya fungsi Blt kita panggil.

```
BackSurface.Blt(@aRect,FSpriteBuffer,
                  @FFrameRect,
                DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_ROTATIONANGLE,
                @BltFX);
```

Yang perlu mendapat perhatian adalah parameter Blt yang ditulis tebal. Agar fungsi Blt mengerti field dwRotationAngle milik BltFX berisi informasi sudut rotasi yang ingin kita gunakan maka kita harus menyertakan flag DDBLT_ROTATIONANGLE saat memanggil Blt, selain itu juga kita harus memberitahukan Blt dimana alamat struktur yang menyimpan informasi efek blit yakni dengan mengisikan @BltFX ke parameter terakhir fungsi Blt.

Jika komputer anda tidak mendukung rotasi sudut sembarang, maka surface tidak akan digambar jika anda menggunakan fungsi rotasi sudut sembarang. Untuk mengetahui apakah rotasi sudut sembarang ini tersedia, kita dapat menggunakan fungsi GetCaps milik interface IDirectDraw (lihat sub.bab 2.11 Mendapatkan Informasi Kemampuan yang Tersedia).

2.10.4.8 Mengisi Surface Dengan Warna.

Fungsi ini sangat membantu jika kita ingin menghapus surface dengan suatu warna, dengan fungsi ini kita tidak perlu menggambar pixel dengan proses looping untuk mengisi surface dengan warna.

Berikut ini adalah contoh rutin untuk mengisi dengan warna.

```
procedure ShowFillColor(BackSurfac,:IDirectDrawSurface; arect:Trect;
                        const color:integer);
var
  bltFX:TDDBltFX;
begin
  If BackSurface.IsLost=DDERR_SURFACELOST then
    BackSurface._Restore;
  FillChar(BltFX,sizeOf(TDDBltFX,0);
  BltFX.dwSize:=SizeOf(TDDBltFX);
  BltFX.dwFillColor:=Color;
  BackSurface.Blt(@aRect,nil,
                  nil,
                  DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_COLORFILL,
                  @BltFX);
end;
```

Rutin ini mirip contoh-contoh rutin sebelumnya untuk melakukan proses rendering dengan Blt. Perbedaannya adalah terdapat tambahan variabel lokal *BltFX* bertipe TDDBltFX. Sebelum memanggil fungsi Blt, BltFX kita siapkan dengan mengisi strukturnya dengan nol,

```
FillChar(BltFX,sizeOf(TDDBltFX,0);
```

kemudian field dwSize milik BltFX kita inisialisasi dengan ukuran struktur dalam byte dan kita isi field dwFillColor kita isi dengan warna yang kita inginkan. Format warna ini harus sesuai dengan format pixel dari surface tujuan.

```
BltFX.dwSize:=SizeOf(TDDBltFX);
BltFX.dwColorFill:=Color;
```

Selanjutnya fungsi Blt kita panggil.

```
BackSurface.Blt(@aRect,nil,
```

***nil**,*

DDBLT_WAIT or DDBLT_KEYSRC ***or DDBLT_COLORFILL**,*

***@BltFX**);*

Yang perlu mendapat perhatian adalah parameter Blt yang ditulis tebal. Agar fungsi Blt mengerti field dwFillColor milik BltFX berisi informasi warna yang ingin kita gunakan maka kita harus menyertakan flag DDBLT_COLORFILL saat memanggil Blt, selain itu juga kita harus memberitahukan Blt dimana alamat struktur yang menyimpan informasi efek blit yakni dengan mengisikan @BltFX ke parameter terakhir fungsi Blt. Parameter source surface dan source rectangle kita isi nil karena kita ingin mengisi BackSurface dengan warna bukan pengkopian data dari suatu surface sumber.

Jika komputer anda tidak mendukung color fill, maka surface tidak akan digambar jika anda menggunakan fungsi color fill. Untuk mengetahui apakah fungsi color fill ini tersedia, kita dapat menggunakan fungsi GetCaps milik interface IDirectDraw (lihat sub.bab 2.11 Mendapatkan Informasi Kemampuan yang Tersedia).

2.11 Mendapatkan Informasi Kemampuan yang Tersedia.

Beberapa fungsi DirectDraw yang kita gunakan di aplikasi kita mungkin tidak didukung oleh komputer dengan konfigurasi perangkat keras yang berbeda. Aplikasi kita bisa saja langsung menghentikan proses dan keluar ketika fungsi yang kita gunakan tidak tersedia, tapi akan lebih baik jika kita memberitahukan pengguna aplikasi kita mengapa hal terjadi.

Untuk mengetahui kemampuan apa yang tersedia di suatu komputer kita dapat menggunakan fungsi *GetCaps* milik interface IDirectDraw.

function GetCaps (lpDDDriverCaps: PDDCaps; lpDDHELCaps: PDDCaps) : HResult; stdcall;

Parameter

lpDDDriverCaps

Variabel yang menunjuk struktur bertipe *TDDCaps* yang akan mencatat kemampuan yang dimiliki oleh perangkat keras (HAL).

lpDDHELCaps

Variabel yang menunjuk struktur bertipe TDDCaps yang akan mencatat kemampuan yang dimiliki oleh software emulasi (HEL).

Jika pemanggilan fungsi ini berhasil maka kedua parameter tersebut akan memyimpan informasi kemampuan HAL dan HEL.

Struktur TDDCaps deklarasinya adalah sebagai berikut:

```
PDDCaps_DX6 = ^TDDCaps_DX6;
TDDCaps_DX6 = packed record
  dwSize: DWORD;                // size of the DDDRIVERCAPS structure
  dwCaps: DWORD;                // driver specific capabilities
  dwCaps2: DWORD;               // more driver specific capabilites
  dwCKeyCaps: DWORD;            // color key capabilities of the surface
  dwFXCaps: DWORD;              // driver specific stretching and effects capabilites
  dwFXAlphaCaps: DWORD;         // alpha driver specific capabilities
  dwPalCaps: DWORD;             // palette capabilities
  dwSVCaps: DWORD;              // stereo vision capabilities
  dwAlphaBltConstBitDepths: DWORD;      // DDBD_2,4,8
  dwAlphaBltPixelBitDepths: DWORD;      // DDBD_1,2,4,8
  dwAlphaBltSurfaceBitDepths: DWORD;    // DDBD_1,2,4,8
  dwAlphaOverlayConstBitDepths: DWORD;  // DDBD_2,4,8
  dwAlphaOverlayPixelBitDepths: DWORD;  // DDBD_1,2,4,8
  dwAlphaOverlaySurfaceBitDepths: DWORD; // DDBD_1,2,4,8
  dwZBufferBitDepths: DWORD;            // DDBD_8,16,24,32
  dwVidMemTotal: DWORD;         // total amount of video memory
  dwVidMemFree: DWORD;          // amount of free video memory
  dwMaxVisibleOverlays: DWORD;  // maximum number of visible overlays
  dwCurrVisibleOverlays: DWORD; // current number of visible overlays
  dwNumFourCCCodes: DWORD;      // number of four cc codes
  dwAlignBoundarySrc: DWORD;    // source rectangle alignment
  dwAlignSizeSrc: DWORD;        // source rectangle byte size
  dwAlignBoundaryDest: DWORD;   // dest rectangle alignment
  dwAlignSizeDest: DWORD;       // dest rectangle byte size
  dwAlignStrideAlign: DWORD;    // stride alignment
  dwRops: Array [0..DD_ROP_SPACE-1] of DWORD;  // ROPS supported
  ddsOldCaps: TDDSCaps;         // Was dssCaps: TDDSCaps. ddsCaps is of type TDDScaps2 for DX6
  dwMinOverlayStretch: DWORD;   // minimum overlay stretch factor multiplied by 1000, eg 1000 ==
1.0, 1300 == 1.3
  dwMaxOverlayStretch: DWORD;   // maximum overlay stretch factor multiplied by 1000, eg 1000 ==
1.0, 1300 == 1.3
  dwMinLiveVideoStretch: DWORD; // minimum live video stretch factor multiplied by 1000, eg 1000
== 1.0, 1300 == 1.3
  dwMaxLiveVideoStretch: DWORD; // maximum live video stretch factor multiplied by 1000, eg 1000
== 1.0, 1300 == 1.3
```

dwMinHwCodecStretch: DWORD; // minimum hardware codec stretch factor multiplied by 1000, eg 1000 == 1.0, 1300 == 1.3

dwMaxHwCodecStretch: DWORD; // maximum hardware codec stretch factor multiplied by 1000, eg 1000 == 1.0, 1300 == 1.3

dwReserved1: DWORD; // reserved

dwReserved2: DWORD; // reserved

dwReserved3: DWORD; // reserved

dwSVBCaps: DWORD; // driver specific capabilities for System->Vmem blts

dwSVBCKeyCaps: DWORD; // driver color key capabilities for System->Vmem blts

dwSVBFXCaps: DWORD; // driver FX capabilities for System->Vmem blts

dwSVBRops: Array [0..DD_ROP_SPACE-1] of DWORD;// ROPS supported for System->Vmem blts

dwVSBCaps: DWORD; // driver specific capabilities for Vmem->System blts

dwVSBCKeyCaps: DWORD; // driver color key capabilities for Vmem->System blts

dwVSBFXCaps: DWORD; // driver FX capabilities for Vmem->System blts

dwVSBRops: Array [0..DD_ROP_SPACE-1] of DWORD;// ROPS supported for Vmem->System blts

dwSSBCaps: DWORD; // driver specific capabilities for System->System blts

dwSSBCKeyCaps: DWORD; // driver color key capabilities for System->System blts

dwSSBFXCaps: DWORD; // driver FX capabilities for System->System blts

dwSSBRops: Array [0..DD_ROP_SPACE-1] of DWORD;// ROPS supported for System->System blts

// Members added for DX5:

dwMaxVideoPorts: DWORD; // maximum number of usable video ports

dwCurrVideoPorts: DWORD; // current number of video ports used

dwSVBCaps2: DWORD; // more driver specific capabilities for System->Vmem blts

dwNLVBCaps: DWORD; // driver specific capabilities for non-local->local vidmem blts

dwNLVBCaps2: DWORD; // more driver specific capabilities non-local->local vidmem blts

dwNLVBCKeyCaps: DWORD; // driver color key capabilities for non-local->local vidmem blts

dwNLVBFXCaps: DWORD; // driver FX capabilities for non-local->local blts

dwNLVBRops: Array [0..DD_ROP_SPACE-1] of DWORD; // ROPS supported for non-local->local blts

// Members added for DX6 release

ddsCaps : TDDSCaps2 ; // Surface Caps

end;

TDDCaps_DX7 = TDDCaps_DX6;

```
  PDDCaps = ^TDDCaps;

{$IFDEF DIRECTX3}
  TDDCaps = TDDCaps_DX3;
{$ELSE}
  {$IFDEF DIRECTX5}
    TDDCaps = TDDCaps_DX5;
  {$ELSE}
    {$IFDEF DIRECTX6}
      TDDCaps = TDDCaps_DX6;
    {$ELSE}
      TDDCaps = TDDCaps_DX7;
    {$ENDIF}
  {$ENDIF}
{$ENDIF}
```

Field-field:

dwSize

Ukuran struktur TDDCaps

2.12 Mengakses Surface Secara Langsung.

2.12.1 Mengunci Surface.

Jika kita ingin mengakses memori yang digunakan oleh suatu surface secara langsung, kita harus mengunci surface tersebut terlebih dahulu. Tujuan proses penguncian ini agar DirectDraw tahu bahwa kita ingin mendapatkan kontrol penuh atas suatu surface, DirectDraw akan mencegah semua proses untuk mengakses memori surface yang sedang kita kunci agar data di memori ini tidak kacau. Akibat proses penguncian adalah kartu grafis tidak dapat melakukan proses blit dan flip suatu surface yang sedang dikunci.

Untuk melakukan penguncian kita menggunakan fungsi anggota interface IDirectDrawSurface *Lock*.

```
function Lock (lpDestRect: PRect; out lpDDSurfaceDesc:TDDSurfaceDesc; dwFlags: DWORD;
              hEvent: THandle) : HResult; stdcall;
```

Parameter-parameter fungsi Lock

lpDestRect

Area pada surface yang ingin dikunci. Jika kita ingin mengunci seluruh surface maka kita isikan parameter ini dengan nilai nil.

lpDDSurfaceDesc

Deskripsi surface bertipe TDDSurfaceDesc. Sebelum memanggil fungsi Lock field *dwSize* harus diisi dengan ukuran struktur TDDSurfaceDesc, dan sebaiknya field-field lain diisi dengan nol. Parameter ini akan diisi dengan deskripsi surface yang kunci jika pemanggilan fungsi Lock berhasil. Field struktur TDDSurfaceDesc yang paling sering digunakan adalah *lpSurface* yang berisi start alamat memori area pada surface yang dikunci, serta *lpPitch* yang berisi data pitch kartu grafis. Kedua field ini erat kaitannya dalam proses perhitungan alamat memori.

dwFlags

Konstanta flag yang digunakan oleh fungsi Lock seperti pada tabel berikut:

Tabel Konstanta Flag Fungsi Lock

DDLOCK_SURFACEMEMORYPTR	Lock akan mengembalikan alamat memori yang valid yang bisa digunakan untuk memanipulasi surface. Nilai ini adalah nilai flag default Lock.
DDLOCK_WAIT	Dengan flag ini maka Lock akan terus mencoba mengunci surface samapai penguncian berhasil dikerjakan. Penguncian dapat gagal bila surface sedang diakses oleh proses lain seperti sedang diblit atau sedang di kunci oleh proses lain. Jika flag ini diset maka Lock tidak pernah menghasilkan error DDERR_WASSTILLDRAWING.
DDLOCK_EVENT	Jika parameter *hEvent* pada fungsi Lock diset maka jika flag ini digunakan maka Dlock akan menjalankan event yang handle-nya disimpan di hEvent.
DDLOCK_READONLY	Memori surface yang sedang dikunci hanya dapat dibaca.
DDLOCK_WRITEONLY	Memori surface yang sedang dikunci hanya dapat ditulis.
DDLOCK_NOSYSLOCK	Penguncian tidak melakukan penguncian terhadap sistem. Flag ini tidak dapat dipergunakan jika surface yang hendak dikunci adalah primary surface.
DDLOCK_NOOVERWRITE	Flag ini hanya digunakan untuk

	penguncian Vertex Buffer pada Direct3D
DDLOCK_DISCARDCONTENTS	Dengan flag ini data yang ada sebelumnya pada surface akan dibuang, sehingga kita tidak boleh mengasumsikan bahwa surface berisi suatu data yang benar. Jika kita memiliki rencana untuk menulis seluruh data yang ada pada surface atau rectangle yang dikunci maka kita sebaiknya menggunakan flag ini.
DDLOCK_OKTOSWAP	Flag ini sama dengan flag DDLOCK_DISCARDCONTENTS hanya berbeda nama.
DDLOCK_DONOTWAIT	Flag ini merupakan lawan dari flag DDLOCK_WAIT.

hEvent

Handle event yang akan dipanggil ketika fungsi Lock berhasil mengunci suatu surface. Parameter ini erat kaitannya dengan flag DDLOCK_EVENT.

Seperti fungsi-fungsi lainnya, status keberhasilannya dapat diketahui dengan fungsi Failed atau Succedeed. Error yang sering timbul adalah DDERR_WASSTILLDRAWING yang menyatakan bahwa surface sedang diakses oleh proses lain.

2.12.2 Memanipulasi Surface Secara Langsung.

Setelah proses Lock berhasil, kita telah memperoleh akses penuh untuk menggunakan bagian surface yang areanya dikunci. Contoh rutin berikut mengerjakan proses menggambar sebuah pixel ke suatu area yang dikunci pada surface berformat 16 bit.

```
Procedure _PutPixel16(buffer:pointer;const X,Y,pitch:integer; const Color:word);
Var offsets:integer;
Begin
   Offsets:=Integer(buffer)+Y*pitch+X*2;
   Move(Color,Pointer(Offsets)^,2);
End;
Procedure PutPixel16(Asurface:IDirectDrawSurface; const X,Y:integer;
                     const Color:word);
var sd:TDDSurfaceDesc;
    hr:HResult;
Begin
```

```
  ZeroMemory(@sd,sizeOf(TDDSurfaceDesc));
  sd.dwSize:= sizeOf(TDDSurfaceDesc);
  hr:=Asurface.Lock(nil,sd,DDLOCK_SURFACEMEMORYPTR or DDLOCK_WAIT,0);
 If Failed(hr)  then exit;
 _PutPixel16(sd.lpSurface,X,Y,sd.lpPitch,Color);
 hr r:=Asurface.Unlock(nil);
End;
```

2.12.3 Membebaskan Surface yang Dikunci.

Setelah kita tidak membutuhkan akses surface, surface tersebut harus kita bebaskan dari proses penguncian agar dapat diakses oleh proses lain dan kartu grafis dapat melakukan blit atau flip.

Untuk membebaskan surface dari proses penguncian digunakan fungsi anggota IDirectDrawSurface *Unlock.*

```
function Unlock (lpSurfaceData: Pointer) : HResult; stdcall;
```

Paramater fungsi Unlock

lpSurfaceData

Pointer yang menunjuk ke start alamat memori surface yang dikunci. Jika kita mengunci seluruh surface maka parameter ini bisa diisi dengan nil, jika tidak maka parameter ini diisi dengan pointer yang dikembalikan fungsi Lock pada field lpSurface struktur TDDSurfaceDesc. Statusnya dapat diketahui fungsi Failed dan Succedeed.

2.13 Finalisasi DirectDraw.

Setelah aplikasi selesai kita perlu membebaskan memori objek DirectDraw dan memori-memori lain yang kita gunakan seperti surface dan data-data lain. Untuk membebaskan memori objek DirectDraw kita isikan harga objek sama dengan nil. Demikian pula untuk setiap objek DirectDrawSurface serta DirectDrawClipper yang kita buat.

Bab 3
Membuat Unit uDirectDraw.Pas

Unit uDirectDraw adalah unit yang berisi kelas-kelas untuk penanganan grafis. Kelas-kelas ini adalah

- *TGraphicEngine* berguna untuk menyederhanakan proses inisialisasi menggunakan DirectDraw, melakukan double buffering, dan proses finalisasi DirectDraw.
- *TSpriteEngine* berguna untuk proses menampilkan sprite.
- *TBackgroundEngine* berguna untuk memudahkan proses background scrolling.
- *TFontEngine* berguna untuk menampilkan tulisan.
- *TAnimation* berguna untuk proses animasi.
- *TAnimationList* berguna untuk menyimpan daftar animasi.
- *TPrimitive*, kelas dasar objek geometri seperti pixel, rectangle, lingkaran dan lain-lain.
- *TPixel*, kelas turunan TPrimitive yang berguna untuk menggambar pixel.
- *TFillRect*, kelas yang berfungsi untuk melakukan proses color fill.

3.1 Listing Unit uDirectDraw.

```
{---------------------------------------}
{  Unit Grafik dengan DirectDraw       }
{    (c) 2002 Zamrony P Juhara         }
{---------------------------------------}
{File:uDirectDraw.Pas             }
{Code:Zamrony P Juhara            }
{Tgl:Oktober 2002                 }
{---------------------------------------}
{x$DEFINE CREATELOG}
unit UDirectDraw;

interface
uses Windows,Classes,Messages,
   SysUtils,Graphics,
   DirectDraw;
type TGraphicEngineParam=record
```

```
    Handle:HWND;
    Width,Height:integer;
    BitPerPixel:integer;
    BackBufferCount:integer;
    FullScreen:boolean;
    AllowReboot:boolean;
    Clipping:boolean;
  end;
 TCapability=Record
    HardwareBlt:boolean;
    ColorKey:boolean;
    ColorKeySrcBlt:boolean;
    StretchBlt:boolean;
    HardwareRotation:boolean;
    HardWareRotate90:boolean;
    HardwareMirrorUpDown:boolean;
    HardwareMirrorLeftRight:boolean;
    HardwareColorFill:boolean;
    HardwareClipping:boolean;
    SoftwareBlt:boolean;
    SoftwareColorKey:boolean;
    SoftwareColorKeySrcBlt:boolean;
    SoftwareStretchBlt:boolean;
    SoftwareRotation:boolean;
    SoftWareRotate90:boolean;
    SoftwareMirrorUpDown:boolean;
    SoftwareMirrorLeftRight:boolean;
    SoftwareColorFill:boolean;
    SoftwareClipping:boolean;
  end;
EGraphicEngineError=class(Exception);
ESpriteEngineError=class(Exception);
EBackgroundEngineError=class(Exception);

TClipList=Array of TRect;
```

```
PClipList=^TClipList;

TGraphicEngine=class(TComponent)
 private
  GEParam:TGraphicEngineParam;
  MyDirectDraw:IDirectDraw;
  MyPrimarySurface,MyBackSurface:IDirectDrawSurface;
  MyClipper,WindowClipper:IDirectDrawClipper;
  MySrcRect,MyDestRect:TRect;
  MyCaps:TDDCaps;
  MySCaps:TDDSCaps;
  FCapability: TCapability;
  procedure CreateOffScreenSurface(out ASurface:IDirectDrawSurface;const
AWidth,AHeight:Dword);
  procedure SetCapability(const Value: TCapability);
  Function  GetCaps:TCapability;
  procedure SetClipRects(const Rects: array of TRect);
  function  GetClipList:TClipList;
  procedure SetClipList(const Value: TClipList);
 public
  constructor Create(GraphicEngineParam:TGraphicEngineParam);reIntroduce;
  destructor Destroy;override;
  procedure Free;
  procedure Show;
  procedure RestoreClipList;
  property  DirectDrawObject:IDirectDraw read MyDirectDraw;
  property  PrimarySurface:IDirectDrawSurface read MyPrimarySurface;
  property  BackSurface:IDirectDrawSurface read MyBackSurface;
  property  ClipList:TClipList read GetClipList write SetClipList;
 published
  property Parameter:TGraphicEngineParam read GEParam;
  property Capability:TCapability read FCapability write SetCapability;
 end;

TPosition=record
  X,Y,Z:integer;
```

```
  end;
 TVelocity=record
  Vx,Vy,Vz:integer;
  end;
 TAcceleration=record
  Ax,Ay,Az:integer;
  end;
 TDirection=(dirLeft,dirRight,dirUp,dirDown,dirRotate180,dirRotate90,
       dirRotate270,dirRotate);
 TPosRef=(prTopLeft,prTopRight,prBottomLeft,prBottomRight);
 TRGB=record
  B,G,R,A:byte;
  end;
 TSpriteEngine=class(TComponent)
  private
   ParentGraphicEngine:TGraphicEngine;
   FFrameNow,FCount: integer;
   FSpriteBuffer:IDirectDrawSurface;
   FFrameRect:array of TRect;
   FPosition: TPosition;
   FAcceleration:TAcceleration;
   FVelocity:TVelocity;
   BltFX:TDDBLTFX;
   FDirection: TDirection;
   FStatus: integer;
   FAngle: integer;
   FPosRef: TPosRef;
   FTransColor: TRGB;
   FFilename:string;
   procedure CreateOffScreenSurface(out ASurface:IDirectDrawSurface;const
AWidth,AHeight:Dword);
   procedure SetFrameNow(const Value: integer);
   procedure SetDirection(const Value: TDirection);
   procedure SetStatus(const Value: integer);
   procedure _Show;
   procedure _ShowMirror;
```

```
  procedure _ShowMirrorUpDown;
  procedure _ShowRotate90;
  procedure _ShowRotate180;
  procedure _ShowRotate270;
  procedure _ShowRotate(const angle:integer);
  function GetWidth:integer;
  function GetHeight:integer;
  function GetRect:TRect;
  procedure SetAngle(const Value: integer);
  procedure SetPosRef(const Value: TPosRef);
  function MapPosRefToActualPos(const aPos:TPosition;AposRef:TPosRef):TPosition;
  procedure SetTransColor(const Value: TRGB);
  procedure SetColorKey;
  procedure Reload;
public
  constructor Create(AGraphicEngine:TGraphicEngine);reintroduce;
  destructor  Destroy;override;
  procedure   Free;
  procedure   LoadFromFile(const filename:string);
  procedure   Show;
published
  property FrameNow:integer read FFrameNow write SetFrameNow;
  property Count:integer read FCount;
  Property Position:TPosition read FPosition write FPosition;
  Property Velocity:TVelocity read FVelocity write FVelocity;
  Property Acceleration:TAcceleration read FAcceleration write FAcceleration;
  property Direction:TDirection read FDirection write SetDirection;
  property Status:integer read FStatus write SetStatus;
  property Width:integer read GetWidth;
  property Height:integer read GetHeight;
  property RectNow:TRect read GetRect;
  property Angle:integer read FAngle write SetAngle;
  property PosRef:TPosRef read FPosRef write SetPosRef;
  property TransColor:TRGB read FTransColor write SetTransColor;
end;
```

```
  TScrollType=(stNormal,stContinue);
  TScrollDirection=(sdVertical,sdHorizontal,sdBoth);
  TBackgroundEngine=class(TComponent)
   private
    ParentGraphicEngine:TGraphicEngine;
    Fy: integer;
    FX: integer;
    FWidth: integer;
    FHeight: integer;
    FScrollType:TScrollType;
    FClipRect:TRect;
    BackgroundBuffer:IDirectDrawSurface;
    FFilename:string;
    procedure CreateOffScreenSurface(out ASurface:IDirectDrawSurface;const
AWidth,AHeight:Dword);
    procedure SetX(const Value: integer);
    procedure Sety(const Value: integer);
    procedure Reload;
   public
    constructor Create(AGraphicEngine:TGraphicEngine);reintroduce;
    destructor Destroy;override;
    procedure Free;
    procedure Show;
    procedure LoadFromFile(const filename:string);
   published
    property X:integer read FX write SetX;
    property Y:integer read Fy write Sety;
    property Width:Integer read FWidth;
    property Height:integer read FHeight;
    property ScrollType:TScrollType read FScrollType;
   end;

  TCharList=array of integer;

  TFontEngine=class(TSpriteEngine)
    private
```

```
    FLoaded:boolean;
    CharList:TCharList;
    FCharCount:Integer;
    FUseGDI: boolean;
    HandleFont:HFont;
    FColor:TColor;
    freed:boolean;
    procedure SetUseGDI(const Value: boolean);
    procedure SetColor(const Value: TColor);
  public
    constructor Create(AGraphicEngine:TGraphicEngine);
    destructor Destroy;override;
    procedure Free;
    procedure LoadFromFile(const Filename:string);
    function  GetLengthInPixel(const str:string):integer;
    procedure WriteString(const str:string);
  published
    property UseGDI:boolean read FUseGDI write SetUseGDI;
    property Color:TColor read FColor write SetColor;
  end;
TTextureStyle=(tsTile,tsStretch);
EFontEngine2Error=class(Exception);
TFontEngine2=class
  private
    ChangeColor:boolean;
    FColor:TColor;
    FFont:TFont;
    FRects:array[0..255] of TRect;
    FABCWidth:array[0..255] of TABC;
    FabcBPlusC:array[0..255] of integer;
    FTextMetric:TTextMetric;
    FFontSurface:IDirectDrawSurface;
    FFontStream:TMemoryStream;
    FFontTexture:TMemoryStream;
    FGraphicEngine:TGraphicEngine;
```

```
        FSurfaceWidth,FSurfaceHeight:cardinal;
        FFontStreamWidth,FFontStreamHeight:cardinal;
        FUseTexture: boolean;
    FTextureStyle: TTextureStyle;
        function GetTexture:TBitmap;
        procedure SetTexture(const value:TBitmap);
        procedure CreateOffScreenSurface(out ASurface:IDirectDrawSurface;const
AWidth,AHeight:Dword);

        procedure InitFont;
        procedure InitFontStream;
        procedure InitFontTexture;
        procedure InitFontSurface;
        procedure FreeFont;
        procedure FreeFontStream;
        procedure FreeFontTexture;
        procedure FreeFontSurface;

        procedure ReloadSurface;
        procedure NilPrivateVar;
        procedure RecreateFont(fnt:TFont);
        procedure DrawChars(abitmap:TBitmap);
        procedure SetColor(const Value: TColor);
        procedure SetFont(const Value: TFont);
        procedure GetABCWidth;
        procedure GetTextMetric;
        procedure GetRects;
        procedure SetUseTexture(const Value: boolean);
    procedure SetTextureStyle(const Value: TTextureStyle);

      public
        constructor Create(AGraphicEngine:TGraphicEngine);
        destructor Destroy;override;
        procedure Free;
        Procedure WriteString(const x,y:integer;const txt:string);
        function GetWidthInPixel(const txt:string):integer;
```

```
    function GetCharHeight:integer;
  published
    property Color:TColor read FColor write SetColor;
    property Font:TFont read FFont write SetFont;
    property Texture:TBitmap read GetTexture write SetTexture;
    property UseTexture:boolean read FUseTexture write SetUseTexture;
    property TextureStyle:TTextureStyle read FTextureStyle write SetTextureStyle;
  end;

TAnimationEvent=procedure(Sender:TObject) of Object;
TAnimationFrame=record
    Frame:integer;
    Position:TPosition;
    Delay:integer;
  end;
PAnimationFrame=^TAnimationFrame;
EAnimationError=class(Exception);


TAnimation=class
  private
    LastTick,Ticks:integer;
    NoFrameList:TList;
    FrameIndex:integer;
    FOnFinished: TAnimationEvent;
    FOnFrameChanged: TAnimationEvent;
    FSpriteFile:String;
    FOnStarted: TAnimationEvent;
    FDeleteOnFinish: boolean;
    FPlaying: boolean;
    procedure SetOnFinished(const Value: TAnimationEvent);
    procedure SetOnFrameChanged(const Value: TAnimationEvent);
    function StrToIntRaise(const str,ExceptMsg:string;var converted:integer):boolean;
    function GetCount:integer;
    function GetAnimationFrames(index: integer): TAnimationFrame;
```

```
    procedure SetAnimationFrames(index: integer;
                const Value: TAnimationFrame);
    function  GetAnimationFrame:TAnimationFrame;
    procedure SetOnStarted(const Value: TAnimationEvent);
    procedure SetDeleteOnFinish(const Value: boolean);
    procedure SetPlaying(const Value: boolean);
  public
    constructor Create;
    destructor Destroy;override;
    procedure Free;
    procedure Add(const NoFrame,X,Y,Z,Delay:integer);
    procedure Clear;
    procedure Delete(const i:integer);
    Procedure LoadFromFile(const filename:string);
    Procedure LoadFromStringList(AStringList:TStringList);
    Procedure SaveToFile(const filename:string);
    Procedure SaveToStringList(var AStringList:TStringList);
    property  SpriteFile:string read FSpriteFile write FSpriteFile;
    property  AnimationFrames[index:integer]:TAnimationFrame read GetAnimationFrames write
SetAnimationFrames;
  published
    property OnStarted:TAnimationEvent read FOnStarted write SetOnStarted;
    property OnFinished:TAnimationEvent read FOnFinished write SetOnFinished;
    property  OnFrameChanged:  TAnimationEvent  read  FOnFrameChanged  write
SetOnFrameChanged;
    property Count:integer read GetCount;
    property AnimationFrame:TAnimationFrame read GetAnimationFrame;
    property DeleteOnFinish:boolean read FDeleteOnFinish write SetDeleteOnFinish;
    property Playing:boolean read FPlaying write SetPlaying;
  end;


  TAnimationList=class(TList)
  private
    function GetItems(Index: integer): TAnimation;
    procedure SetItems(Index: integer; const Value: TAnimation);
  protected
```

```
 public
  function isAnimationInList(anim:TAnimation;var AnimIndex:integer):boolean;
  function Add(Item:TAnimation):integer;
  function Replace(oldItem,newItem:TAnimation):boolean;overload;
  function Replace(animList:TAnimationList;newItem:TAnimation):boolean;overload;
  procedure Delete(item:TAnimation);
  property Items[Index:integer]:TAnimation read GetItems write SetItems;
 end;
EPrimitiveError=class(Exception);
TPrimitive=class
 private
  parentGraphicEngine:TGraphicEngine;
  procedure SetColor(const Value: integer);
  procedure SetPosition(const Value: TPosition);
 protected
  FColor: integer;
  FPosition: TPosition;
  SurfDesc:TDDSurfaceDesc;
  PixelFormat:TDDPixelFormat;
 public
  constructor Create(AGraphicEngine:TGraphicEngine);
  destructor Destroy;override;
  procedure Free;
  procedure BeginDraw;
  procedure Draw;virtual;abstract;
  procedure EndDraw;
 published
  property Position:TPosition read FPosition write SetPosition;
  property Color:integer read FColor write SetColor;
 end;
TPixel=class(TPrimitive)
  private
  public
    procedure Draw;override;
 end;
```

```
        TFillRect=class
        private
         bltfx:TDDBltFX;
         FSurface: IDirectDrawSurface;
         Fcolor: integer;
         FRect: TRect;
         procedure Setcolor(const Value: integer);
         procedure SetRect(const Value: TRect);
         procedure SetSurface(const Value: IDirectDrawSurface);
        public
          constructor Create;
          destructor Destroy;override;
          procedure Free;
          procedure Draw;
          property Surface:IDirectDrawSurface read FSurface write SetSurface;
          property color:integer read Fcolor write Setcolor;
          property Rect:TRect read FRect write SetRect;
        end;
Function SetPosition(const x,y,z:integer):TPosition;
Function SetVelocity(const vx,vy,vz:integer):TVelocity;
Function SetAcceleration(const ax,ay,az:integer):TAcceleration;

Function AddPosition(const Pos1,Pos2:TPosition):TPosition;
Function AddVelocity(const Vel1,Vel2:TVelocity):TVelocity;
Function AddAcceleration(const Accel1,Accel2:TAcceleration):TAcceleration;

Function RGB16(const R,G,B:byte):word;
Function RGB15(const R,G,B:byte):word;
Function RGB24(const R,G,B:byte):integer;
Function RGB32(const A,R,G,B:byte):integer;

implementation
{$IFDEF CREATELOG}
var log:TLogFile;
{$ENDIF}
```

```
function Min(const a,b:integer):integer;
var res:integer;
begin
  if a<b then Res:=a else res:=b;
  Result:=res;
end;
function Max(const a,b:integer):integer;
var res:integer;
begin
  if a>b then Res:=a else res:=b;
  Result:=res;
end;

Function RGB16(const R,G,B:byte):word;
begin
  Result:=((R and 32) shr 11) or ((G and 32) shr 6)or ((B and 32));
end;
Function RGB15(const R,G,B:byte):word;
begin
  Result:=((R and 32) shr 10) or ((G and 32) shr 5)or ((B and 32));
end;
Function RGB24(const R,G,B:byte):integer;
begin
  Result:=((R and $ff) shr 16) or ((G and $ff) shr 8)or ((B and $ff));
end;
Function RGB32(const A,R,G,B:byte):integer;
begin
  Result:=((A and $ff) shr 24)or ((R and $ff) shr 16) or ((G and $ff) shr 8)or ((B and $ff));
end;
Function SetPosition(const x,y,z:integer):TPosition;
begin
  Result.X:=x;
  Result.Y:=y;
  Result.Z:=z;
end;
```

```
Function SetVelocity(const vx,vy,vz:integer):TVelocity;
begin
  Result.VX:=vx;
  Result.VY:=vy;
  Result.VZ:=vz;
end;
Function SetAcceleration(const ax,ay,az:integer):TAcceleration;
begin
  Result.AX:=ax;
  Result.AY:=ay;
  Result.AZ:=az;
end;

Function AddPosition(const Pos1,Pos2:TPosition):TPosition;
begin
  Result.X:=Pos1.X+Pos2.X;
  Result.Y:=Pos1.Y+Pos2.Y;
  Result.Z:=Pos1.Z+Pos2.Z;
end;
Function AddVelocity(const Vel1,Vel2:TVelocity):TVelocity;
begin
  Result.Vx:=Vel1.Vx+Vel2.Vx;
  Result.Vy:=Vel1.Vy+Vel2.Vy;
  Result.Vz:=Vel1.Vz+Vel2.Vz;
end;
Function AddAcceleration(const Accel1,Accel2:TAcceleration):TAcceleration;
begin
  Result.Ax:=Accel1.Ax+Accel2.Ax;
  Result.Ay:=Accel1.Ay+Accel2.Ay;
  Result.Az:=Accel1.Az+Accel2.Az;
end;
function MyAbs(const Value:integer):integer;
begin
  if Value<0 then Result:=-Value else
  Result:=Value;
```

```
end;

procedure Clip(xDest,yDest,dwDestWidth,dwDestHeight,dwSrcWidth,dwSrcHeight:integer;
          var newxDest,newyDest,dwNewDestWidth,dwNewDestHeight,startX,startY:integer;
          var displayed:boolean);overload;
begin
 dwNewDestWidth:=dwDestWidth;
 dwNewDestHeight:=dwDestHeight;
 newxDest:=xDest;
 newyDest:=yDest;
 startX:=0;
 startY:=0;
 displayed:=not (xDest>dwSrcWidth);
 if not displayed then exit;
 displayed:=not (xDest+dwDestWidth<0);
 if not displayed then exit;
 displayed:=not (yDest>dwSrcHeight);
 if not displayed then exit;
 displayed:=not (yDest+dwDestHeight<0);
 if not displayed then exit;

 if xDest<0 then
 begin
  startX:=MyAbs(xDest);
  newxDest:=0;
  dwNewDestWidth:=dwDestWidth-startX;
 end else
 if (xDest+dwDestWidth)>dwSrcWidth then
 begin
  dwNewDestWidth:=dwSrcWidth-xDest;
 end;
 if yDest<0 then
 begin
  startY:=MyAbs(yDest);
  newyDest:=0;
```

```
    dwNewDestHeight:=dwDestHeight-StartY;
  end else
  if (yDest+dwDestHeight)>dwSrcHeight then
  begin
    dwNewDestHeight:=dwSrcHeight-yDest;
  end;
end;
procedure Clip(xDest,yDest,dwDestWidth,dwDestHeight,dwSrcWidth,dwSrcHeight:longint;
            var newxDest,newyDest,dwNewDestWidth,dwNewDestHeight:longint;
            var displayed:boolean);overload;
begin
  dwNewDestWidth:=dwDestWidth;
  dwNewDestHeight:=dwDestHeight;
  newxDest:=xDest;
  newyDest:=yDest;

  displayed:=not (xDest>dwSrcWidth);
  if not displayed then exit;
  displayed:=not (xDest+dwDestWidth<0);
  if not displayed then exit;
  displayed:=not (yDest>dwSrcHeight);
  if not displayed then exit;
  displayed:=not (yDest+dwDestHeight<0);
  if not displayed then exit;

  if xDest<0 then
  begin
    newxDest:=0;
    dwNewDestWidth:=dwDestWidth-xDest;
  end else
  if (xDest+dwDestWidth)>dwSrcWidth then
  begin
    dwNewDestWidth:=dwSrcWidth-xDest;
  end;
  if yDest<0 then
```

```
  begin
    newyDest:=0;
    dwNewDestHeight:=dwDestHeight-yDest;
  end else
  if (yDest+dwDestHeight)>dwSrcHeight then
  begin
    dwNewDestHeight:=dwSrcHeight-yDest;
  end;
end;

function  DDSetPalette( ADirectDraw :IDirectDraw; Bitmap :TBitmap ) : IDirectDrawPalette;
type TRGB = array[ 0..255 ] of TRGBQuad ;
     PRGB = ^TRGB ;
var i, n            : integer ;
    APE             : array[ 0..255 ] of TPaletteEntry ;
    bfHeader        : TBitmapFileHeader ;
    biHeader        : TBitmapInfoHeader ;
    Temp            : byte ;
    MemoryStream    :TMemoryStream;
begin
  If Not Assigned(Bitmap) Then Exit;
  try
    MemoryStream := TMemoryStream.Create;
    Bitmap.SaveToStream(MemoryStream);
    MemoryStream.Seek(0,0);
    MemoryStream.Read( bfHeader, SizeOf( bfHeader ) ) ;
    MemoryStream.Read( biHeader, SizeOf( biHeader ) ) ;
    MemoryStream.Read( APE, SizeOf( APE ) ) ;
  finally
    MemoryStream.Free;
  end;
  //get the number of colors in the color table
  if biHeader.biSize <> SizeOf( TBitmapInfoHeader )then n := 0
  else
  if biHeader.biBitCount > 8 then n := 0
```

```
  else
  if biHeader.biClrUsed = 0  then n := 1 SHL biHeader.biBitCount
  else n := biHeader.biClrUsed ;

  //  a DIB color table has its colors stored BGR not RGB
  //  so flip them around.
  for i := 0 to n - 1 do
  begin
    Temp         := APE[i].peRed ;
    APE[i].peRed  := APE[i].peBlue ;
    APE[i].peBlue := Temp ;
  end;

  // create the DD palette
  if ADirectDraw.CreatePalette( DDPCAPS_8BIT, @APE[ 0 ], Result, NIL ) <> DD_OK then
    Raise Exception.Create( 'DirectDraw.CreatePalette failed' ) ;
end ;

{ TGraphicEngine }

constructor TGraphicEngine.Create(GraphicEngineParam: TGraphicEngineParam);
var hr:HResult;
    ddsCap:TDDSCAPS;
    surfaceDesc:TDDSurfaceDesc;
    cooplevel:dword;
begin
  GEParam:=GraphicEngineParam;
  hr:=DirectDrawCreate(nil,MyDirectDraw,nil);
  if Succeeded(hr) then
  begin
    FCapability:=GetCaps;
    case GraphicEngineParam.FullScreen of
      true:begin
           coopLevel:=DDSCL_FULLSCREEN or DDSCL_EXCLUSIVE;
           if GraphicEngineParam.AllowReboot then
```

```
  coopLevel:=coopLevel or DDSCL_ALLOWREBOOT;
hr:=MyDirectDraw.SetCooperativeLevel(GraphicEngineParam.Handle,coopLevel);
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
  exit;
end;

hr:=MyDirectDraw.SetDisplayMode(GraphicEngineParam.Width,
                                GraphicEngineParam.Height,
                                GraphicEngineParam.BitPerPixel);
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
  exit;
end;

fillchar(surfacedesc,sizeof(surfacedesc),0);
surfacedesc.dwSize:=sizeof(surfacedesc);
surfacedesc.dwFlags:=DDSD_CAPS or DDSD_BACKBUFFERCOUNT;
surfacedesc.ddsCaps.dwCaps:=DDSCAPS_PRIMARYSURFACE or DDSCAPS_FLIP or
DDSCAPS_COMPLEX;
Surfacedesc.dwBackBufferCount:=GEParam.BackBufferCount;
hr:=MyDirectDraw.CreateSurface(Surfacedesc,MyPrimarySurface,nil);
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
  exit;
end;
ddsCap.dwCaps:=DDSCAPS_BACKBUFFER;
hr:=MyPrimarySurface.GetAttachedSurface(ddsCap,MyBackSurface);
if Failed(hr) then
begin
```

```
      Pointer(MyDirectDraw):=nil;
      Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
      exit;
    end;

    MySrcRect:=Rect(0,0,GEParam.Width,GEParam.Height);
    if GEParam.Clipping then
    begin
      hr:=MyDirectDraw.CreateClipper(0,MyClipper,nil);
      if Failed(hr) then
      begin
        Pointer(MyDirectDraw):=nil;
        Pointer(MyPrimarySurface):=nil;
        Pointer(MyBackSurface):=nil;
        Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
        exit;
      end;
      SetClipRects([mySrcRect]);
      MyBackSurface.SetClipper(MyClipper);
    end;
  end;
false:begin
    coopLevel:=DDSCL_NORMAL;
    hr:=MyDirectDraw.SetCooperativeLevel(GraphicEngineParam.Handle,coopLevel);
    if Failed(hr) then
    begin
      Pointer(MyDirectDraw):=nil;
      Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
      exit;
    end;
    fillchar(surfacedesc,sizeof(surfacedesc),0);
    surfacedesc.dwSize:=sizeof(surfacedesc);
    surfacedesc.dwFlags:=DDSD_CAPS;
    surfacedesc.ddsCaps.dwCaps:=DDSCAPS_PRIMARYSURFACE;
    hr:=MyDirectDraw.CreateSurface(Surfacedesc,MyPrimarySurface,nil);
```

```
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
  exit;
end;
CreateOffScreenSurface(MyBackSurface,GEParam.Width,GEParam.Height);
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
  exit;
end;
MyDestRect:=Rect(0,0,0,0);
MySrcRect:=Rect(0,0,GEParam.Width,GEParam.Height);
hr:=MyDirectDraw.CreateClipper(0,WindowClipper,nil);
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Pointer(MyPrimarySurface):=nil;
  Pointer(MyBackSurface):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
  exit;
end;
WindowClipper.SetHWnd(0,GEParam.Handle);
MyPrimarySurface.SetClipper(WindowClipper);

If GEParam.Clipping then
begin
  hr:=MyDirectDraw.CreateClipper(0,MyClipper,nil);
  if Failed(hr) then
  begin
    Pointer(MyDirectDraw):=nil;
    Pointer(MyPrimarySurface):=nil;
    Pointer(MyBackSurface):=nil;
```

```
            Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
            exit;
          end;
          SetClipRects([MySrcRect]);
          MyBackSurface.SetClipper(MyClipper);
        end;
      end;
  end;
 end else Raise EGraphicEngineError.Create('Error:'+DDErrorString(hr));
end;

procedure TGraphicEngine.SetClipRects(const Rects: array of TRect);
type
 PArrayRect = ^TArrayRect;
 TArrayRect = Array[0..0] of TRect;
var
 RgnData: PRgnData;
 i: Integer;
 BoundsRect: TRect;
 hr:HResult;
begin
 BoundsRect := Rect(MaxInt, MaxInt, -MaxInt, -MaxInt);
 for i:=Low(Rects) to High(Rects) do
 begin
  with BoundsRect do
  begin
   Left := Min(Rects[i].Left, Left);
   Right := Max(Rects[i].Right, Right);
   Top := Min(Rects[i].Top, Top);
   Bottom := Max(Rects[i].Bottom, Bottom);
  end;
 end;

 GetMem(RgnData, SizeOf(TRgnDataHeader)+SizeOf(TRect)*(High(Rects)-Low(Rects)+1));
 try
```

```
    with RgnData^.rdh do
    begin
      dwSize := SizeOf(TRgnDataHeader);
      iType := RDH_RECTANGLES;
      nCount := High(Rects)-Low(Rects)+1;
      nRgnSize := nCount*SizeOf(TRect);
      rcBound := BoundsRect;
    end;
    for i:=Low(Rects) to High(Rects) do
      PArrayRect(@RgnData^.Buffer)^[i-Low(Rects)] := Rects[i];
    hr := MyClipper.SetClipList(RgnData, 0);
    if failed(hr) then
     raise EGraphicEngineError.Create('Unable to set clip list.');
  finally
    FreeMem(RgnData);
  end;
end;

procedure TGraphicEngine.CreateOffScreenSurface(out ASurface:IDirectDrawSurface; const AWidth,
  AHeight: Dword);
var surfaceDesc:TDDSurfaceDesc;
   hr:HResult;
begin
  fillchar(surfacedesc,sizeof(surfacedesc),0);
  surfacedesc.dwSize:=sizeof(surfacedesc);
  surfacedesc.dwFlags:=DDSD_CAPS or DDSD_HEIGHT or DDSD_WIDTH;
  surfacedesc.ddsCaps.dwCaps:=DDSCAPS_OFFSCREENPLAIN;
  Surfacedesc.dwHeight:=AHeight;
  surfacedesc.dwWidth:=AWidth;
  hr:=MyDirectDraw.CreateSurface(Surfacedesc,ASurface,nil);
  if Failed(hr) then raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
end;

destructor TGraphicEngine.Destroy;
begin
```

```
  if GEParam.FullScreen then
  begin
    MyDirectDraw.RestoreDisplayMode;
  end;
  if Pointer(MyClipper)<>nil then Pointer(MyClipper):=nil;
  if Pointer(WindowClipper)<>nil then Pointer(WindowClipper):=nil;
  Pointer(MyPrimarySurface):=nil;
  Pointer(MyBackSurface):=nil;
  Pointer(MyDirectDraw):=nil;
  inherited;
end;

procedure TGraphicEngine.Free;
begin
  if self<>nil then destroy;
end;

function TGraphicEngine.GetCaps: TCapability;
var ddHALCaps,ddHELCaps:TDDCaps;
    hr:HResult;
    caps:TCapability;
begin
  FillChar(ddHALCaps,sizeOf(TDDCaps),0);
  FillChar(ddHELCaps,sizeOf(TDDCaps),0);
  FillChar(caps,sizeOf(TCapability),0);
  ddHALCaps.dwSize:=sizeOf(TDDCaps);
  ddHELCaps.dwSize:=sizeOf(TDDCaps);
  hr:=MyDirectDraw.GetCaps(@ddHALCaps,@ddHELCaps);

  if Failed(hr) then Raise EGraphicEngineError.Create('Error:'+DDErrorString(hr));

  caps.HardwareBlt:=((ddHALCaps.dwCaps and DDCAPS_BLT)=DDCAPS_BLT);
  caps.HardwareColorFill:=((ddHALCaps.dwCaps and
DDCAPS_BLTCOLORFILL)=DDCAPS_BLTCOLORFILL);
  caps.StretchBlt:=((ddHALCaps.dwCaps and DDCAPS_BLTSTRETCH)=DDCAPS_BLTSTRETCH);
  caps.ColorKey:=((ddHALCaps.dwCaps and DDCAPS_COLORKEY)=DDCAPS_COLORKEY);
```

```
  caps.ColorKeySrcBlt:=((ddHALCaps.dwCKeyCaps and
DDCKEYCAPS_SRCBLT)=DDCKEYCAPS_SRCBLT);

  caps.HardwareClipping:=((ddHALCaps.dwCaps and DDCAPS_CANCLIP)=DDCAPS_CANCLIP);

  caps.HardwareRotation:=((ddHALCaps.dwFXCaps and
DDFXCAPS_BLTROTATION)=DDFXCAPS_BLTROTATION);

  caps.HardwareRotate90:=((ddHALCaps.dwFXCaps and
DDFXCAPS_BLTROTATION90)=DDFXCAPS_BLTROTATION90);

  caps.HardwareMirrorUpDown:=((ddHALCaps.dwFXCaps and
DDFXCAPS_BLTMIRRORUPDOWN)=DDFXCAPS_BLTMIRRORUPDOWN);

  caps.HardwareMirrorLeftRight:=((ddHALCaps.dwFXCaps and
DDFXCAPS_BLTMIRRORLEFTRIGHT)=DDFXCAPS_BLTMIRRORLEFTRIGHT);


  caps.SoftwareBlt:=((ddHELCaps.dwCaps and DDCAPS_BLT)=DDCAPS_BLT);

  caps.SoftwareColorFill:=((ddHALCaps.dwCaps and
DDCAPS_BLTCOLORFILL)=DDCAPS_BLTCOLORFILL);

  caps.SoftwareStretchBlt:=((ddHELCaps.dwCaps and
DDCAPS_BLTSTRETCH)=DDCAPS_BLTSTRETCH);

  caps.SoftwareColorKey:=((ddHELCaps.dwCaps and
DDCAPS_COLORKEY)=DDCAPS_COLORKEY);

  caps.SoftwareColorKeySrcBlt:=((ddHELCaps.dwCKeyCaps and
DDCKEYCAPS_SRCBLT)=DDCKEYCAPS_SRCBLT);

  caps.SoftwareClipping:=((ddHELCaps.dwCaps and DDCAPS_CANCLIP)=DDCAPS_CANCLIP);

  caps.SoftwareRotation:=((ddHELCaps.dwFXCaps and
DDFXCAPS_BLTROTATION)=DDFXCAPS_BLTROTATION);

  caps.SoftwareRotate90:=((ddHELCaps.dwFXCaps and
DDFXCAPS_BLTROTATION90)=DDFXCAPS_BLTROTATION90);

  caps.SoftwareMirrorUpDown:=((ddHELCaps.dwFXCaps and
DDFXCAPS_BLTMIRRORUPDOWN)=DDFXCAPS_BLTMIRRORUPDOWN);

  caps.SoftwareMirrorLeftRight:=((ddHELCaps.dwFXCaps and
DDFXCAPS_BLTMIRRORLEFTRIGHT)=DDFXCAPS_BLTMIRRORLEFTRIGHT);


  Result:=Caps;

end;


procedure TGraphicEngine.SetCapability(const Value: TCapability);

begin

  FCapability := Value;

end;
```

```
procedure TGraphicEngine.Show;
var P:TPoint;
begin
  if MyPrimarySurface.IsLost=DDERR_SURFACELOST then
    MyPrimarySurface._Restore;
  if MyBackSurface.IsLost=DDERR_SURFACELOST then
    MyBackSurface._Restore;

  case GEParam.FullScreen of
   true:begin
         MyPrimarySurface.Flip(nil,DDFLIP_WAIT);
        end;
   false:begin
         p:=Point(0,0);
         ClientToScreen(GEParam.Handle,p);
         GetClientRect(GEParam.Handle,MyDestRect);
         OffsetRect(MyDestRect,p.X,p.Y);
         MyPrimarySurface.Blt(@MyDestRect,MyBackSurface,@MySrcRect,DDBLT_WAIT,nil);
        end;
  end;
end;
function TGraphicEngine.GetClipList: TClipList;
var r:PRect;
    reg:PRgnData;
    siz,i:cardinal;
begin
 if GEParam.Clipping then
 begin
  if MyClipper<>nil then
  begin
    MyClipper.GetClipList(r,reg,siz);
    SetLength(Result,High(reg.Buffer)-Low(reg.Buffer)+1);
    for i:=Low(reg.Buffer) to High(reg.Buffer) do
    begin
      Result[i]:=PClipList(@Reg^.Buffer)^[i-Low(reg.buffer)];
```

```
    end;
  end;
 end;
end;

procedure TGraphicEngine.SetClipList(const Value: TClipList);
begin
  if GEParam.Clipping then
    SetClipRects(Value);
end;

procedure TGraphicEngine.RestoreClipList;
begin
  SetClipRects([mySrcRect]);
end;

{ TSpriteEngine }
constructor TSpriteEngine.Create(AGraphicEngine: TGraphicEngine);
begin
  FAngle:=0;
  FCount:=0;
  FFrameRect:=nil;
  FFrameNow:=0;
  FDirection:=dirLeft;
  FPosRef:=prTopLeft;
  FTransColor.B:=0;
  FTransColor.G:=0;
  FTransColor.R:=0;
  FTransColor.A:=0;
  FFilename:='';
  FPosition:=SetPosition(0,0,0);
  FVelocity:=SetVelocity(0,0,0);
  FAcceleration:=SetAcceleration(0,0,0);
  Pointer(FSpriteBuffer):=nil;
  ParentGraphicEngine:=AGraphicEngine;
```

```
  FillChar(BltFX,sizeOf(TDDBLTFX),0);
  BltFX.dwSize:=sizeOf(TDDBLTFX);
  BltFX.dwDDFX:=DDBLTFX_MIRRORLEFTRIGHT;
end;


procedure TSpriteEngine.CreateOffScreenSurface(
  out ASurface: IDirectDrawSurface; const AWidth, AHeight: Dword);
var surfaceDesc:TDDSurfaceDesc;
   hr:HResult;
   zz:TDDColorKey;
begin
  fillchar(surfacedesc,sizeof(surfacedesc),0);
  surfacedesc.dwSize:=sizeof(surfacedesc);
  surfacedesc.dwFlags:=DDSD_CAPS or DDSD_HEIGHT or DDSD_WIDTH;
  surfacedesc.ddsCaps.dwCaps:=DDSCAPS_OFFSCREENPLAIN;
  Surfacedesc.dwHeight:=AHeight;
  surfacedesc.dwWidth:=AWidth;
  hr:=ParentGraphicEngine.DirectDrawObject.CreateSurface(Surfacedesc,ASurface,nil);
  if Failed(hr) then raise ESpriteEngineError.Create('Error'+DDErrorString(hr));
  zz.dwColorSpaceLowValue:=0;
  zz.dwColorSpaceHighValue:=0;
  hr:=ASurface.SetColorKey(DDCKEY_SRCBLT,@zz);
  if Failed(hr) then raise ESpriteEngineError.Create('Error'+DDErrorString(hr));
end;

destructor TSpriteEngine.Destroy;
begin
  if FSpriteBuffer<>nil then
  begin
    Pointer(FSpriteBuffer):=nil;
  end;
  Finalize(FFrameRect);
  inherited;
end;
```

```
procedure TSpriteEngine.Free;
begin
  if Self<>nil then Destroy;
end;

procedure TSpriteEngine.LoadFromFile(const filename: string);
var filestr:TextFile;
    ID,imagename,framecountstr:string;
    leftstr,rightstr,topstr,bottomstr:string;
    bmp:TBitmap;
    ctr:integer;
    DC:HDC;
    hr:HResult;
begin
  if FileExists(filename) then
  begin
    AssignFile(filestr,filename);
    Reset(Filestr);
    readln(filestr,ID);
    if ID='SPRITE' then
    begin
      readln(filestr,ImageName);
      if FileExists(Imagename) then
      begin
       try
        bmp:=TBitmap.Create;
        bmp.loadFromFile(imagename);
        CreateOffScreenSurface(FSpriteBuffer,bmp.Width,bmp.Height);
        hr:=FSpriteBuffer.GetDC(DC);
        if Failed(hr) then
        begin
          raise ESpriteEngineError.Create('Error:'+DDErrorString(hr));
          Exit;
        end;
```

```
      BitBlt(DC,0,0,bmp.Width,bmp.Height,bmp.Canvas.Handle,0,0,SRCCOPY);
      hr:=FSpriteBuffer.ReleaseDC(DC);
      if Failed(hr) then
      begin
        raise ESpriteEngineError.Create('Error:'+DDErrorString(hr));
        Exit;
      end;
     finally
      bmp.free;
     end;
    end else
    begin
      Raise ESpriteEngineError.Create('Error: Sprite bitmap '+imagename+' not found.');
      Exit;
    end;
    readln(FileStr,framecountstr);
    FCount:=StrToInt(framecountStr);
    setLength(FframeRect,FCount);
    for ctr:=0 to FCount-1 do
    begin
      readln(filestr,leftstr);
      FFrameRect[ctr].Left:=StrToInt(leftstr);
      readln(filestr,topstr);
      FFrameRect[ctr].Top:=StrToInt(topstr);
      readln(filestr,rightstr);
      FFrameRect[ctr].Right:=StrToInt(Rightstr);
      readln(filestr,bottomstr);
      FFrameRect[ctr].Bottom:=StrToInt(bottomstr);
    end;
  end;
  FFilename:=filename;
  Closefile(filestr);
end else
begin
  raise ESpriteEngineError.Create('Error: Sprite file '+ filename+' not found.');
```

```
    exit;
  end;
end;

procedure TSpriteEngine.SetDirection(const Value: TDirection);
begin
  FDirection := Value;
end;

procedure TSpriteEngine.SetFrameNow(const Value: integer);
begin
  if (Value>=0) and (Value<FCount) then
  FFrameNow := Value;
end;

procedure TSpriteEngine._Show;
var aRect:TRect;
    aWidth,aHeight:integer;
    aPos:TPosition;
    h:HResult;
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
  begin
     h:=FSpriteBuffer._Restore;
     if failed(h) then Reload;
  end;
  If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
     ParentGraphicEngine.BackSurface._Restore;
  aWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
  aHeight:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
  aPos:=MapPosRefToActualPos(FPosition,FPosRef);
  aRect:=Rect(aPos.X,aPos.Y,
           aPos.X+AWidth,aPos.Y+aHeight);
  ParentGraphicEngine.BackSurface.Blt(@aRect,FSpriteBuffer,@FFrameRect[FFRameNow],
                        DDBLT_WAIT or DDBLT_KEYSRC,
```

```
                    nil);
end;

procedure TSpriteEngine._ShowMirror;
var aRect:TRect;
   aWidth,aHeight:integer;
   aPos:TPosition;
   h:HResult;
begin
 If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
 begin
   h:=FSpriteBuffer._Restore;
   if failed(h) then Reload;
 end;
 If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
    ParentGraphicEngine.BackSurface._Restore;
 aWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
 aHeight:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
 aPos:=MapPosRefToActualPos(FPosition,FPosRef);
 aRect:=Rect(aPos.X,aPos.Y,
         aPos.X+AWidth,aPos.Y+aHeight);
 BltFX.dwDDFX:=DDBLTFX_MIRRORLEFTRIGHT;
 ParentGraphicEngine.BackSurface.Blt(@aRect,FSpriteBuffer,@FFrameRect[FFRameNow],
                    DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                    @BltFX);
end;

procedure TSpriteEngine._ShowMirrorUpDown;
var aRect:TRect;
   aWidth,aHeight:integer;
   aPos:TPosition;
   h:HResult;
begin
 If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
 begin
```

```
    h:=FSpriteBuffer._Restore;
    if failed(h) then Reload;
  end;
  If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
     ParentGraphicEngine.BackSurface._Restore;
  aWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
  aHeight:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
  aPos:=MapPosRefToActualPos(FPosition,FPosRef);
  aRect:=Rect(aPos.X,aPos.Y,
           aPos.X+AWidth,aPos.Y+aHeight);
  BltFX.dwDDFX:=DDBLTFX_MIRRORUPDOWN;
  ParentGraphicEngine.BackSurface.Blt(@aRect,FSpriteBuffer,@FFrameRect[FFRameNow],
                        DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                        @BltFX);
end;

procedure TSpriteEngine.Show;
begin
  case FDirection of
    dirLeft:_Show;
    dirRight:begin
              if (ParentGraphicEngine.Capability.HardwareMirrorLeftRight) or
                (ParentGraphicEngine.Capability.SoftwareMirrorLeftRight) then
                _ShowMirror;
             end;
    dirUp:_Show;
    dirDown:begin
              if (ParentGraphicEngine.Capability.HardwareMirrorUpDown) or
                (ParentGraphicEngine.Capability.SoftwareMirrorUpDown) then
                _showMirrorUpDown;
            end;
    dirRotate90:begin
              if (ParentGraphicEngine.Capability.HardwareRotate90) or
                (ParentGraphicEngine.Capability.SoftwareRotate90) then
                _ShowRotate90;
```

```
            end;
   dirRotate180:begin
             if (ParentGraphicEngine.Capability.HardwareRotate90) or
              (ParentGraphicEngine.Capability.SoftwareRotate90) then
              _ShowRotate180;
             end;
   dirRotate270:begin
             if (ParentGraphicEngine.Capability.HardwareRotate90) or
              (ParentGraphicEngine.Capability.SoftwareRotate90) then
               _ShowRotate270;
             end;
   dirRotate:begin
            if (ParentGraphicEngine.Capability.HardwareRotation) or
              (ParentGraphicEngine.Capability.SoftwareRotation) then
   _ShowRotate(FAngle);
           end;
  end;
end;

procedure TSpriteEngine.SetStatus(const Value: integer);
begin
  FStatus := Value;
end;

function TSpriteEngine.GetHeight: integer;
begin
  Result:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
end;

function TSpriteEngine.GetWidth: integer;
begin
  Result:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
end;

function TSpriteEngine.GetRect: TRect;
```

```
begin
  Result:=FFrameRect[FFrameNow];
end;

procedure TSpriteEngine._ShowRotate180;
var aRect:TRect;
    aWidth,aHeight:integer;
    aPos:TPosition;
    h:HResult;
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
  begin
    h:=FSpriteBuffer._Restore;
    if failed(h) then Reload;
  end;
  If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
     ParentGraphicEngine.BackSurface._Restore;
  aWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
  aHeight:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
  aPos:=MapPosRefToActualPos(FPosition,FPosRef);
  aRect:=Rect(aPos.X,aPos.Y,
           aPos.X+AWidth,aPos.Y+aHeight);
  BltFX.dwDDFX:=DDBLTFX_ROTATE180;
  ParentGraphicEngine.BackSurface.Blt(@aRect,FSpriteBuffer,@FFrameRect[FFRameNow],
                        DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                        @BltFX);
end;

procedure TSpriteEngine._ShowRotate270;
var aRect:TRect;
    aWidth,aHeight:integer;
    aPos:TPosition;
    h:HResult;
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
```

```
 begin
   h:=FSpriteBuffer._Restore;
   if failed(h) then Reload;
 end;
 If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
    ParentGraphicEngine.BackSurface._Restore;
 aWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
 aHeight:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
 aPos:=MapPosRefToActualPos(FPosition,FPosRef);
 aRect:=Rect(aPos.X,aPos.Y,
          aPos.X+AWidth,aPos.Y+aHeight);
 BltFX.dwDDFX:=DDBLTFX_ROTATE270;
 ParentGraphicEngine.BackSurface.Blt(@aRect,FSpriteBuffer,@FFrameRect[FFRameNow],
                      DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                      @BltFX);
end;

procedure TSpriteEngine._ShowRotate90;
var aRect:TRect;
   aWidth,aHeight:integer;
   aPos:TPosition;
   h:HResult;
begin
 If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
 begin
   h:=FSpriteBuffer._Restore;
   if failed(h) then Reload;
 end;
 If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
    ParentGraphicEngine.BackSurface._Restore;
 aWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
 aHeight:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
 aPos:=MapPosRefToActualPos(FPosition,FPosRef);
 aRect:=Rect(aPos.X,aPos.Y,
          aPos.X+AWidth,aPos.Y+aHeight);
```

```
  BltFX.dwDDFX:=DDBLTFX_ROTATE90;
  ParentGraphicEngine.BackSurface.Blt(@aRect,FSpriteBuffer,@FFrameRect[FFRameNow],
                        DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_DDFX,
                        @BltFX);
end;

procedure TSpriteEngine._ShowRotate(const angle: integer);
var aRect:TRect;
    aWidth,aHeight:integer;
    aPos:TPosition;
    h:HResult;
begin
  If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
  begin
    h:=FSpriteBuffer._Restore;
    if failed(h) then Reload;
  end;
  If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
     ParentGraphicEngine.BackSurface._Restore;
  aWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
  aHeight:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
  aPos:=MapPosRefToActualPos(FPosition,FPosRef);
  aRect:=Rect(aPos.X,aPos.Y,
           aPos.X+AWidth,aPos.Y+aHeight);
  BltFX.dwRotationAngle:=Fangle;
  ParentGraphicEngine.BackSurface.Blt(@aRect,FSpriteBuffer,@FFrameRect[FFRameNow],
                        DDBLT_WAIT or DDBLT_KEYSRC or DDBLT_ROTATIONANGLE,
                        @BltFX);
end;

procedure TSpriteEngine.SetAngle(const Value: integer);
begin
  FAngle := Value;
end;
```

```
procedure TSpriteEngine.SetPosRef(const Value: TPosRef);
begin
  FPosRef := Value;
end;

function TSpriteEngine.MapPosRefToActualPos(const aPos: TPosition;
  AposRef: TPosRef): TPosition;
begin
  result:=apos;
  case aPosRef of
    prTopLeft:;
    prTopRight:begin
                 result.X:=aPos.X-(FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left);
               end;
    prBottomLeft:begin
                 result.Y:=aPos.Y-(FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top);
               end;
    prBottomRight:begin
                 result.X:=aPos.X-(FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left);
                 result.Y:=aPos.Y-(FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top);
               end;
  end;
end;

procedure TSpriteEngine.SetTransColor(const Value: TRGB);
begin
  FTransColor := Value;
  SetColorKey;
end;

procedure TSpriteEngine.SetColorKey;
var hr:HResult;
    zz:TDDColorKey;
    col:cardinal;
    pxformat:TDDPixelFormat;
```

```
  pal:IDirectDrawPalette;
begin
  ZeroMemory(@pxformat,SizeOf(TDDPixelFormat));
  pxFormat.dwSize:=SizeOf(TDDPixelFormat);
  hr:=FSpriteBuffer.GetPixelFormat(pxFormat);
  if Failed(hr) then
  begin
    raise ESpriteEngineError.Create('Error'+DDErrorString(hr));
    exit;
  end;
  case pxFormat.dwFlags of
    DDPF_RGB          :begin
                       Case pxFormat.dwRGBBitCount of
                        15:col:=cardinal(RGB15(FTransColor.R,FTransColor.G,FTransColor.B));
                        16:col:=cardinal(RGB16(FTransColor.R,FTransColor.G,FTransColor.B));
                        24:col:=cardinal(RGB24(FTransColor.R,FTransColor.G,FTransColor.B));
32:col:=cardinal(RGB32(FTransColor.A,FTransColor.R,FTransColor.G,FTransColor.B));
                       end;
                      end;
    DDPF_PALETTEINDEXED8:begin
                       FSpriteBuffer.GetPalette(pal);
                      end;
  end;
  zz.dwColorSpaceLowValue:=col;
  zz.dwColorSpaceHighValue:=col;
  hr:=FSpriteBuffer.SetColorKey(DDCKEY_SRCBLT,@zz);
  if Failed(hr) then
  begin
    raise ESpriteEngineError.Create('Error'+DDErrorString(hr));
    exit;
  end;
end;

procedure TSpriteEngine.Reload;
begin
```

```
  if FFilename<>'' then
  begin
    FSpriteBuffer:=nil;
    Finalize(FFrameRect);
    LoadFromFile(FFilename);
  end;
end;

{ TBackgroundEngine }

constructor TBackgroundEngine.Create(AGraphicEngine: TGraphicEngine);
var bltfx:TDDBltFX;
begin
  ParentGraphicEngine:=AGraphicEngine;
  FX:=0;
  FY:=0;
  FWidth:=0;
  FHeight:=0;
  FFilename:='';
  FScrollType:=stNormal;
  BackgroundBuffer:=nil;
  CreateOffScreenSurface(backgroundbuffer,
                ParentGraphicEngine.Parameter.Width,
                ParentGraphicEngine.Parameter.Height);

  ZeroMemory(@bltfx,SizeOf(TDDBltFX));
  bltFX.dwSize:=SizeOf(TDDBltFX);
  bltfx.dwFillColor:=0;
  BackgroundBuffer.Blt(nil,nil,nil,DDBLT_WAIT or DDBLT_COLORFILL,@bltfx);
end;

procedure TBackgroundEngine.CreateOffScreenSurface(
  out ASurface: IDirectDrawSurface; const AWidth, AHeight: Dword);
var surfaceDesc:TDDSurfaceDesc;
   hr:HResult;
```

```
begin
  fillchar(surfacedesc,sizeof(surfacedesc),0);
  surfacedesc.dwSize:=sizeof(surfacedesc);
  surfacedesc.dwFlags:=DDSD_CAPS or DDSD_HEIGHT or DDSD_WIDTH;
  surfacedesc.ddsCaps.dwCaps:=DDSCAPS_OFFSCREENPLAIN;
  Surfacedesc.dwHeight:=AHeight;
  surfacedesc.dwWidth:=AWidth;
  hr:=ParentGraphicEngine.DirectDrawObject.CreateSurface(Surfacedesc,ASurface,nil);
  if Failed(hr) then raise EBackgroundEngineError.Create('Error'+DDErrorString(hr));
end;

destructor TBackgroundEngine.Destroy;
begin
  if BackgroundBuffer<>nil then
  begin
    Pointer(BackgroundBuffer):=nil;
  end;
  inherited;
end;

procedure TBackgroundEngine.Free;
begin
  if Self<>nil then Destroy;
end;

procedure TBackgroundEngine.LoadFromFile(const filename: string);
var bmp:TBitmap;
    DC:HDC;
    hr:HResult;
begin
 try
  if FileExists(filename) then
  begin
    try
      bmp:=TBitmap.Create;
```

```
    bmp.LoadFromFile(filename);
    FFilename:=Filename;
    if bmp.Width<ParentGraphicEngine.Parameter.Width then
      FWidth:=ParentGraphicEngine.Parameter.Width else
      FWidth:=bmp.Width;
    if bmp.Height<ParentGraphicEngine.Parameter.Height then
      FHeight:=ParentGraphicEngine.Parameter.Height else
      FHeight:=bmp.Height;
    BackgroundBuffer:=nil;
    CreateOffScreenSurface(BackgroundBuffer,FWidth,FHeight);
    hr:=BackgroundBuffer.GetDC(DC);
    if Failed(hr) then EBackgroundEngineError.Create('Error:'+DDErrorString(hr));
    BitBlt(DC,0,0,bmp.Width,bmp.Height,bmp.Canvas.Handle,0,0,SRCCOPY);
    hr:=BackgroundBuffer.ReleaseDC(DC);
    if Failed(hr) then EBackgroundEngineError.Create('Error:'+DDErrorString(hr));
   finally
    bmp.Free;
   end;
  end else Raise EBackgroundEngineError.Create('Error:Background bitmap '+filename+' not found.');
 except
  Raise EBackgroundEngineError.Create('Error:');
 end;
end;

procedure TBackgroundEngine.Reload;
begin
 if FFilename<>'' then
 begin
  BackgroundBuffer:=nil;
  LoadfromFile(FFilename);
 end;
end;

procedure TBackgroundEngine.SetX(const Value: integer);
begin
```

```
  case FScrollType of
   stNormal:begin
            if (Value>=0)and
              ((Value+ParentGraphicEngine.Parameter.Width)<FWidth) then
              FX := Value;
           end;
   stContinue:;
  end;
end;

procedure TBackgroundEngine.Sety(const Value: integer);
begin
  case FScrollType of
   stNormal:begin
            if (Value>=0)and
              ((Value+ParentGraphicEngine.Parameter.Height)<FHeight) then
              FY := Value;
           end;
   stContinue:;
  end;
end;

procedure TBackgroundEngine.Show;
var aRect:TRect;
    h:HResult;
begin
  FClipRect:=Rect(FX,FY,
                  FX+ParentGraphicEngine.GEParam.Width,
                  FY+ParentGraphicEngine.GEParam.Height);
  aRect:=Rect(0,0,
              ParentGraphicEngine.GEParam.Width,
              ParentGraphicEngine.GEParam.Height);
  If BackgroundBuffer.IsLost=DDERR_SURFACELOST then
  begin
    h:=BackgroundBuffer._Restore;
```

```
    if failed(h) then Reload;
  end;
  If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
     ParentGraphicEngine.BackSurface._Restore;
//  ParentGraphicEngine.BackSurface.BltFast(0,0,BackgroundBuffer,@FClipRect,DDBLTFAST_WAIT);
  ParentGraphicEngine.BackSurface.Blt(@aRect,BackgroundBuffer,@FClipRect,DDBLT_WAIT,nil)
end;

{ TFontEngine }

constructor TFontEngine.Create(AGraphicEngine: TGraphicEngine);
begin
  inherited Create(AGraphicEngine);
  FLoaded:=false;
  FUseGDI:=false;
  FColor:=clWhite;
//  HandleFont:=CreateFont(
end;

destructor TFontEngine.Destroy;
begin
//  if FUseGDI then
//  DeleteObject(HandleFont);
   Finalize(CharList);
  inherited;
end;

procedure TFontEngine.Free;
begin
  if self<>nil then destroy;
end;

function TFontEngine.GetLengthInPixel(const str: string): integer;
var i,lenstr,chrWidth:integer;
begin
```

```
  Result:=0;
  if FLoaded then
  begin
    chrWidth:=0;
    lenstr:=Length(str);
    for i:=1 to LenStr do
    begin
      FFrameNow:=CharList[Ord(str[i])-32];
      chrWidth:=chrWidth+FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left+1;
    end;
    Result:=chrWidth;
  end;
end;

procedure TFontEngine.LoadFromFile(const Filename: string);
var filestr:TextFile;
    ID,imagename,charcountstr,
    framecountstr,framestr,chstr:string;
    leftstr,rightstr,topstr,bottomstr:string;
    bmp:TBitmap;
    ctr,frameNo,i:integer;
    ch:char;
    DC:HDC;
    hr:HResult;
begin
  if FileExists(filename) then
  begin
    AssignFile(filestr,filename);
    Reset(Filestr);
    readln(filestr,ID);
    if ID='FONTSPR' then
    begin
      readln(filestr,ImageName);
      if FileExists(Imagename) then
      begin
```

```
  try
   bmp:=TBitmap.Create;
   bmp.loadFromFile(imagename);
   CreateOffScreenSurface(FSpriteBuffer,bmp.Width,bmp.Height);
   hr:=FSpriteBuffer.GetDC(DC);
   if Failed(hr) then
   begin
     ESpriteEngineError.Create('Error:'+DDErrorString(hr));
     exit;
   end;
   BitBlt(DC,0,0,bmp.Width,bmp.Height,bmp.Canvas.Handle,0,0,SRCCOPY);
   hr:=FSpriteBuffer.ReleaseDC(DC);
   if Failed(hr) then
   begin
     ESpriteEngineError.Create('Error:'+DDErrorString(hr));
     exit;
   end
  finally
   bmp.free;
  end;
  FLoaded:=true;
end;
readln(FileStr,framecountstr);
readln(FileStr,Charcountstr);
FCount:=StrToInt(framecountStr);
FCharCount:=StrToInt(charcountStr);
setLength(FFrameRect,FCount);
setLength(CharList,{FCharCount}256);
for i:=0 to FCount-1 do
begin
  readln(filestr,leftstr);
  FFrameRect[i].Left:=StrToInt(leftstr);
  readln(filestr,topstr);
  FFrameRect[i].Top:=StrToInt(topstr);
  readln(filestr,rightstr);
```

```
      FFrameRect[i].Right:=StrToInt(Rightstr);
      readln(filestr,bottomstr);
      FFrameRect[i].Bottom:=StrToInt(bottomstr);
    end;

    for i:=0 to FCharCount-1 do
    begin
      readln(filestr,ch);
      readln(filestr,framestr);
      FrameNo:=StrToInt(frameStr);
      if (ch=' ') then ch:=#32;
      CharList[Ord(ch)-32]:=FrameNo;
    end;
  end;
  Closefile(filestr);
 end;
end;

procedure TFontEngine.SetColor(const Value: TColor);
begin
 FColor := Value;
end;

procedure TFontEngine.SetUseGDI(const Value: boolean);
begin
 FUseGDI := Value;
end;

procedure TFontEngine.WriteString(const str: string);
var i,lenstr,chrWidth:integer;
   dc:HDC;
   ch:char;
begin
 if FUseGDI then
 begin
```

```
    ParentGraphicEngine.BackSurface.GetDC(dc);
    SetBkColor(dc,TRANSPARENT);
    SetTextColor(DC,FColor);
//    SelectObject(DC,HandleFont);
    TextOut(dc,FPosition.X,FPosition.Y,PChar(str),length(str));
    ParentGraphicEngine.BackSurface.ReleaseDC(dc);
  end else
  begin
  if FLoaded then
  begin
    lenstr:=Length(str);
    for i:=1 to LenStr do
    begin
      if (str[i]=' ') then
       ch:=#32 else
        ch:=str[i];
      FFrameNow:=CharList[Ord(ch)-32];
      chrWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
      Show;
      FPosition.x:=FPosition.x+chrWidth;
    end;
  end;
  end;
end;

{ TAnimation }

procedure TAnimation.Add(const NoFrame, X, Y,Z, Delay: integer);
var item:PAnimationFrame;
begin
  new(item);
  item.Frame:=NoFrame;
  item.Position:=SetPosition(x,y,z);
  item.Delay:=Delay;
  NoFrameList.Add(item);
```

```
end;

procedure TAnimation.Clear;
var i:integer;
    item:PAnimationFrame;
begin
  for i:=NoFrameList.Count-1 downto 0 do
  begin
    item:=PAnimationFrame(NoFrameList.Items[i]);
    dispose(item);
    item:=nil;
    NoFrameList.Delete(i);
  end;
end;

constructor TAnimation.Create;
begin
  LastTick:=0;
  Ticks:=0;
  FrameIndex:=0;
  NoFrameList:=TList.Create;
  FOnFrameChanged:=nil;
  FOnFinished:=nil;
  FOnStarted:=nil;
  FDeleteOnFinish:=false;
  FSpriteFile:='';
  FPlaying:=false;
end;

procedure TAnimation.Delete(const i: integer);
var item:PAnimationFrame;
begin
  item:=PAnimationFrame(NoFrameList.items[i]);
  dispose(item);
  NoFrameList.Delete(i);
```

```
end;

destructor TAnimation.Destroy;
begin
  Clear;
  NoFrameList.Free;
  NoFrameList:=nil;
  inherited;
end;

procedure TAnimation.Free;
begin
  if self<>nil then destroy;
end;

function TAnimation.GetAnimationFrame: TAnimationFrame;
begin
   if (FrameIndex=0) and (Assigned(FOnStarted)) then
   begin
     FPlaying:=true;
     FOnStarted(self);
   end;
   Result:=PAnimationFrame(NoFrameList.Items[FrameIndex])^;
   Ticks:=GetTickCount;
   if (Ticks-LastTick)>PAnimationFrame(NoFrameList.Items[FrameIndex]).Delay then
   begin
     LastTick:=Ticks;
     Inc(FrameIndex);
     if Assigned(FOnFrameChanged) then
       FOnFrameChanged(self);
     if FrameIndex=NoFrameList.Count then
     begin
       FrameIndex:=0;
       FPlaying:=false;
       if Assigned(FOnFinished) then
```

```
        FOnFinished(self);
      end;
    end;
end;

function TAnimation.GetAnimationFrames(index: integer): TAnimationFrame;
var anim:TAnimationFrame;
begin
  anim.Frame:=0;
  anim.Position:=setPosition(0,0,0);
  anim.Delay:=0;
  if (index>=0) and (index<NoFrameList.Count) then
  begin
    anim:=PAnimationFrame(NoFrameList.Items[index])^;
  end;
  Result:=anim;
end;

function TAnimation.GetCount: integer;
begin
  result:=NoFrameList.Count;
end;

procedure TAnimation.LoadFromFile(const filename: string);
var i,noFrame,delay,framectr:integer;
    aPos:TPosition;
    str:string;
    FT:TextFile;
begin
  if FileExists(Filename) then
  begin
    try
      AssignFile(FT,Filename);
      Reset(FT);
      Readln(FT,str);
```

```
if str='ANIM' then
begin
  Readln(FT,str);
  if FileExists(str) then
  begin
    FSpriteFile:=str;
  end else
  begin
    Raise EAnimationError.Create('Sprite file:'+str+' not found.');
    exit;
  end;
  Readln(FT,str);
  if StrToIntRaise(str,
     Filename+' contains invalid format.Unable to read number of frame.',
     framectr)=false then exit;
  Clear;
  for i:=0 to frameCtr-1 do
  begin
    Readln(FT,str);
    if StrToIntRaise(str,
       Filename+' contains invalid format.Unable to read frame No.'+IntTostr(i),
       Noframe)=false then exit;
    Readln(FT,str);
    if StrToIntRaise(str,
       Filename+' contains invalid format.Unable to read delay No.'+IntTostr(i),
       Delay)=false then exit;
    Readln(FT,str);
    if StrToIntRaise(str,
       Filename+' contains invalid format.Unable to read pos x No.'+IntTostr(i),
       aPos.x)=false then exit;
    Readln(FT,str);
    if StrToIntRaise(str,
       Filename+' contains invalid format.Unable to read pos y No.'+IntTostr(i),
       aPos.y)=false then exit;
    Readln(FT,str);
```

```
        if StrToIntRaise(str,
          Filename+' contains invalid format.Unable to read pos z No.'+IntTostr(i),
          aPos.z)=false then exit;
        Add(NoFrame,aPos.X,apos.Y,aPos.Z,delay);
      end;
    end else
      Raise EAnimationError.Create(Filename+' contains invalid format.');
  finally
    CloseFile(FT);
  end;
 end else
  Raise EAnimationError.Create(Filename+' not found.');
end;

procedure TAnimation.LoadFromStringList(AStringList: TStringList);
var i,noFrame,delay,framectr:integer;
  aPos:TPosition;
  str:string;
begin
  try
    str:=AStringList[0];
    if str='ANIM' then
    begin
      Str:=AStringList[1];
      if FileExists(str) then
      begin
        FSpriteFile:=str;
      end else
      begin
        Raise EAnimationError.Create('Sprite File:'+FSpriteFile+' not found.');
        exit;
      end;
      str:=AStringList[2];
      if StrToIntRaise(str,
        'String list contains invalid format.Unable to read number of frame.',
```

```
        framectr)=false then exit;
      Clear;
      for i:=0 to frameCtr-1 do
      begin
       str:=AStringList[i+3];
       if StrToIntRaise(str,
         'String list contains invalid format.Unable to read frame No.'+IntTostr(i),
         Noframe)=false then exit;
       str:=AStringList[i+4];
       if StrToIntRaise(str,
         'String list contains invalid format.Unable to read delay No.'+IntTostr(i),
         Delay)=false then exit;
       str:=AStringList[i+5];
       if StrToIntRaise(str,
         'String list contains invalid format.Unable to read pos x No.'+IntTostr(i),
         aPos.x)=false then exit;
       str:=AStringList[i+6];
       if StrToIntRaise(str,
         'String contains invalid format.Unable to read pos y No.'+IntTostr(i),
         aPos.y)=false then exit;
       str:=AStringList[i+7];
       if StrToIntRaise(str,
         'String list contains invalid format.Unable to read pos z No.'+IntTostr(i),
         aPos.z)=false then exit;
       Add(NoFrame,aPos.X,apos.Y,aPos.Z,delay);
      end;
    end else
      Raise EAnimationError.Create('String list contains invalid format.');
  finally
  end;
end;

procedure TAnimation.SaveToFile(const filename: string);
var i,framectr:integer;
  aPos:TPosition;
```

```
  str:string;
  FT:TextFile;
  anim:PAnimationFrame;
begin
  try
   AssignFile(FT,Filename);
   Rewrite(FT);
   str:='ANIM';
   Writeln(FT,str);
   str:=FSpriteFile;
   Writeln(FT,str);
   str:=IntToStr(NoFrameList.Count);
   Writeln(FT,str);
   framectr:=NoFrameList.Count;
   for i:=0 to frameCtr-1 do
   begin
     anim:=PAnimationFrame(NoFrameList.Items[i]);
     Writeln(FT,IntToStr(anim.Frame));
     Writeln(FT,IntToStr(anim.Delay));
     Writeln(FT,IntToStr(anim.Position.X));
     Writeln(FT,IntToStr(anim.Position.Y));
     Writeln(FT,IntToStr(anim.Position.Z));
   end;
  finally
   CloseFile(FT);
  end;
end;

procedure TAnimation.SaveToStringList(var AStringList: TStringList);
var i,framectr:integer;
  aPos:TPosition;
  str:string;
  FT:TextFile;
  anim:PAnimationFrame;
begin
```

```
    AStringList.Clear;
    AStringList[0]:='ANIM';
    AStringList[1]:=FSpriteFile;
    AStringList[2]:=IntToStr(NoFrameList.Count);
    framectr:=NoFrameList.Count;
    for i:=0 to frameCtr-1 do
    begin
      anim:=PAnimationFrame(NoFrameList.Items[i]);
      AStringList[i+3]:=IntToStr(anim.Frame);
      AStringList[i+4]:=IntToStr(anim.Delay);
      AStringList[i+5]:=IntToStr(anim.Position.X);
      AStringList[i+6]:=IntToStr(anim.Position.Y);
      AStringList[i+7]:=IntToStr(anim.Position.Z);
    end;
end;

procedure TAnimation.SetAnimationFrames(index: integer;
  const Value: TAnimationFrame);
var anim:PAnimationFrame;
begin
  if (index>=0) and (index<NoFrameList.Count) then
  begin
    anim:=PAnimationFrame(NoFrameList.Items[index]);
    anim^.Frame:=Value.Frame;
    anim^.Position:=Value.Position;
    anim^.Delay:=Value.Delay;
  end;
end;

procedure TAnimation.SetDeleteOnFinish(const Value: boolean);
begin
  FDeleteOnFinish := Value;
end;

procedure TAnimation.SetOnFinished(const Value: TAnimationEvent);
```

```
begin
  FOnFinished := Value;
end;

procedure TAnimation.SetOnFrameChanged(const Value: TAnimationEvent);
begin
  FOnFrameChanged := Value;
end;

procedure TAnimation.SetOnStarted(const Value: TAnimationEvent);
begin
  FOnStarted := Value;
end;

procedure TAnimation.SetPlaying(const Value: boolean);
begin
  FPlaying := Value;
end;

function TAnimation.StrToIntRaise(const str, ExceptMsg: string;
  var converted: integer): boolean;
begin
  result:=true;
  try
    converted:=StrToInt(str);
  except
    On EConvertError do
    begin
      result:=false;
      Raise EAnimationError.Create(ExceptMsg);
    end;
  end;
end;

{ TAnimationList }
```

```
function TAnimationList.Add(Item: TAnimation): integer;
begin
  result:=Inherited Add(Pointer(item));
end;

function TAnimationList.GetItems(Index: integer): TAnimation;
begin
  Result:=TAnimation(Inherited Items[index]);
end;

function TAnimationList.Replace(oldItem, newItem: TAnimation): boolean;
var i:integer;
begin
  result:=false;
  if isAnimationInList(olditem,i) then
  begin
    inherited insert(i,Pointer(newitem));
    inherited Delete(i+1);
    result:=true;
  end;
  if result=false then Add(newitem);
end;

function TAnimationList.isAnimationInList(anim: TAnimation;var animIndex:integer): boolean;
var i:integer;
    item:TAnimation;
begin
  result:=false;
  for i:=0 to Count-1 do
  begin
    item:=GetItems(i);
    if item=anim then
    begin
      animIndex:=i;
```

```
      result:=true;
      exit;
    end;
  end;
end;

function TAnimationList.Replace(animList: TAnimationList;
  newItem: TAnimation): boolean;
var i,j:integer;
    item:TAnimation;
begin
  result:=false;
  for i:=0 to Count-1 do
  begin
    item:=GetItems(i);
    for j:=0 to animList.Count-1 do
    begin
      if item=animlist.Items[j] then
      begin
        inherited Insert(i,newitem);
        inherited Delete(i+1);
        result:=true;
        exit;
      end;
    end;
  end;
  if result=false then Add(newitem);
end;

procedure TAnimationList.SetItems(Index: integer; const Value: TAnimation);
begin
  Inherited Items[index]:=Pointer(Value);
end;

procedure TAnimationList.Delete(item: TAnimation);
```

```
var indx:integer;
begin
  if isAnimationInList(item,indx) then
  begin
    inherited Delete(indx);
  end;
end;
{ TPrimitive }
procedure TPrimitive.BeginDraw;
var hr:HResult;
begin
  if (ParentGraphicEngine<>nil) and
     (ParentGraphicEngine.BackSurface<>nil) then
  begin
    hr:=ParentGraphicEngine.BackSurface.Lock(nil,SurfDesc,DDLOCK_SURFACEMEMORYPTR or
DDLOCK_WAIT,0);
    if Failed(hr) then
    begin
      raise EPrimitiveError.Create(DDErrorString(hr));
      exit;
    end;
  end;
end;
procedure TPrimitive.EndDraw;
var hr:HResult;
begin
  if (ParentGraphicEngine<>nil) and
     (ParentGraphicEngine.BackSurface<>nil) then
  begin
    hr:=ParentGraphicEngine.BackSurface.Unlock(nil);
    if Failed(hr) then
    begin
      raise EPrimitiveError.Create(DDErrorString(hr));
      exit;
    end;
  end;
```

```
end;

constructor TPrimitive.Create(AGraphicEngine: TGraphicEngine);
begin
  ParentGraphicEngine:=AGraphicEngine;
  FColor:=$ffffffff;
  FPosition:=uDirectDraw.SetPosition(0,0,0);

  ZeroMemory(@SurfDesc,SizeOf(TDDSurfaceDesc));
  SurfDesc.dwSize:=SizeOf(TDDSurfaceDesc);

  ZeroMemory(@PixelFormat,SizeOf(TDDPixelFormat));
  if(ParentGraphicEngine<>nil) and
    (ParentGraphicEngine.BackSurface<>nil)then
  begin
    PixelFormat.dwSize:=Sizeof(TDDPixelFormat);
    ParentGraphicEngine.BackSurface.GetPixelFormat(PixelFormat);
  end;
end;

destructor TPrimitive.Destroy;
begin
  inherited;
end;

procedure TPrimitive.Free;
begin
  if self<>nil then destroy;
end;

procedure TPrimitive.SetColor(const Value: integer);
begin
  FColor := Value;
end;
```

```
procedure TPrimitive.SetPosition(const Value: TPosition);
begin
  FPosition := Value;
end;

{ TPixel }
procedure TPixel.Draw;
var offsets,pixelwidth:integer;
begin
  inherited;
  case PixelFormat.dwFlags of
    DDPF_RGB:begin
              case PixelFormat.dwRGBBitCount of
               15,16:pixelwidth:=2;
               24:pixelwidth:=3;
               32:pixelwidth:=4;
              end;
            end;
    DDPF_PALETTEINDEXED8:begin
                  pixelwidth:=1;
                       end;
  end;
  offsets:=Integer(SurfDesc.lpSurface)+FPosition.Y*SurfDesc.lPitch+FPosition.X*pixelwidth;
  move(FColor,Pointer(offsets)^,pixelwidth);
end;

{ TFillRect }

constructor TFillRect.Create;
begin
  ZeroMemory(@bltfx,SizeOf(TDDBltFX));
  FColor:=0;
  ZeroMemory(@FRect,SizeOf(TRect));
  FSurface:=nil;
end;
```

```
destructor TFillRect.Destroy;
begin

  inherited;
end;

procedure TFillRect.Draw;
begin
 if FSurface<>nil then
 begin
  ZeroMemory(@bltfx,SizeOf(TDDBltFX));
  bltFX.dwSize:=SizeOf(TDDBltFX);
  bltfx.dwFillColor:=FColor;
  FSurface.Blt(@FRect,nil,nil,DDBLT_WAIT or DDBLT_COLORFILL,@bltfx);
 end;
end;

procedure TFillRect.Free;
begin
  if self<>nil then Destroy;
end;

procedure TFillRect.Setcolor(const Value: integer);
begin
  Fcolor := Value;
end;

procedure TFillRect.SetRect(const Value: TRect);
begin
  FRect := Value;
end;

procedure TFillRect.SetSurface(const Value: IDirectDrawSurface);
begin
```

```
  FSurface := Value;
end;

{ TFontEngine2 }
constructor TFontEngine2.Create(AGraphicEngine: TGraphicEngine);
begin
  FGraphicEngine:=AGraphicEngine;
  nilPrivateVar;
  InitFont;
  GetTextMetric;
  GetABCWidth;
  GetRects;
  InitFontStream;
  InitFontTexture;
  InitFontSurface;
end;

procedure TFontEngine2.CreateOffScreenSurface(
  out ASurface: IDirectDrawSurface; const AWidth, AHeight: Dword);
var surfaceDesc:TDDSurfaceDesc;
   hr:HResult;
   zz:TDDColorKey;
begin
  fillchar(surfacedesc,sizeof(surfacedesc),0);
  surfacedesc.dwSize:=sizeof(surfacedesc);
  surfacedesc.dwFlags:=DDSD_CAPS or DDSD_HEIGHT or DDSD_WIDTH;
  surfacedesc.ddsCaps.dwCaps:=DDSCAPS_OFFSCREENPLAIN;
  Surfacedesc.dwHeight:=AHeight;
  surfacedesc.dwWidth:=AWidth;
  hr:=FGraphicEngine.DirectDrawObject.CreateSurface(Surfacedesc,ASurface,nil);
  if Failed(hr) then raise ESpriteEngineError.Create('Error'+DDErrorString(hr));
  zz.dwColorSpaceLowValue:=0;
  zz.dwColorSpaceHighValue:=0;
  hr:=ASurface.SetColorKey(DDCKEY_SRCBLT,@zz);
  if Failed(hr) then raise EFontEngine2Error.Create('Error'+DDErrorString(hr));
```

```
end;

destructor TFontEngine2.Destroy;
begin
  FreeFont;
  FreeFontStream;
  FreeFontTexture;
  FreeFontSurface;
  nilprivateVar;
  FGraphicEngine:=nil;
  inherited;
end;

procedure TFontEngine2.DrawChars(abitmap: TBitmap);
var i,j:integer;
    c:byte;
    str:string;
begin
  abitmap.Width:=FFontStreamWidth;
  abitmap.Height:=FFontStreamHeight;
  abitmap.Canvas.Brush.Color:=clBlack;
  abitmap.Canvas.Font:=FFont;
  abitmap.Canvas.FloodFill(abitmap.width shr 1,abitmap.Height shr 1,clBlack,fsBorder);
  SetBkMode(abitmap.Canvas.Handle,TRANSPARENT);
  for i:=0 to 13 do
  for j:=0 to 15 do
  begin
    c:=(i shl 4)+j+32;
    str:='';
    str:=str+chr(c);
    abitmap.Canvas.TextOut(j*FTextMetric.tmMaxCharWidth-FABCWidth[c].abcA,
              i*FTextMetric.tmHeight,str);
  end;
end;
```

```
procedure TFontEngine2.Free;
begin
  if self<>nil then destroy;
end;

procedure TFontEngine2.FreeFont;
begin
  FFont.Free;
end;

procedure TFontEngine2.FreeFontStream;
begin
  FFontStream.Free;
end;

procedure TFontEngine2.FreeFontSurface;
begin
  FFontSurface:=nil;
end;

procedure TFontEngine2.FreeFontTexture;
begin
  FFontTexture.Free;
end;

procedure TFontEngine2.GetABCWidth;
var dc:HDC;
    i:integer;
    sz:TSize;
    ch:Char;
    pch:PAnsiChar;
begin
  if FFont.Handle<>0 then
  begin
    dc:=CreateCompatibleDC(0);
```

```
    SelectObject(dc,FFont.Handle);
    if GetCharABCWidths(dc,0,255,FABCWidth[0])=false then
    begin
      FillChar(FABCWidth[0],SizeOf(TABC) shl 8,0);
      for i:=32 to 255 do
      begin
        ch:=Chr(i);
        pch:=@ch;
        GetTextExtentPoint32(dc,pch,1,sz);
        FABCWidth[i].abcB:=sz.cx;
      end;
    end;
    DeleteDC(dc);
  end;
end;

function TFontEngine2.GetCharHeight: integer;
begin
  result:=FTextmetric.tmHeight;
end;

procedure TFontEngine2.GetRects;
var i,cellWidth,cellHeight:integer;
begin
  CellWidth:=FFontStreamWidth shr 4;
  CellHeight:=FFontStreamHeight div 14;
  for i:=32 to 255 do
  begin
    FRects[i].Left:=((i-32) mod 16)*CellWidth;
    FRects[i].Top:=((i-32) shr 4)*CellHeight;
    FRects[i].Right:=FRects[i].Left+FABCWidth[i].abcB;
    FRects[i].Bottom:=FRects[i].Top+CellHeight;
    FabcBPlusC[i]:=FABCWidth[i].abcB+FABCWidth[i].abcC;
  end;
end;
```

```
procedure TFontEngine2.GetTextMetric;
var dc:HDC;
begin
  if FFont.Handle<>0 then
  begin
    dc:=CreateCompatibleDC(0);
    SelectObject(dc,FFont.Handle);
    GetTextMetrics(dc,FTextMetric);
    DeleteDC(dc);
    FFontStreamWidth:=FTextMetric.tmMaxCharWidth shl 4;
    FFontStreamHeight:=FTextMetric.tmHeight*14;
  end;
end;

function TFontEngine2.GetTexture: TBitmap;
var abmp:TBitmap;
begin
  abmp:=TBitmap.Create;
  abmp.LoadFromStream(FFontTexture);
end;

function TFontEngine2.GetWidthInPixel(const txt: string): integer;
var i,len,wd:integer;
    c:byte;
begin
  len:=length(txt);
  wd:=0;
  for i:=1 to len do
  begin
    c:=Ord(txt[i]);
    wd:=wd+FABCWidth[c].abcA+FabcBPlusC[c];
  end;
  result:=wd;
end;
```

```
procedure TFontEngine2.InitFont;
begin
  FFont.Free;
  FFont:=nil;
  FFont:=TFont.Create;
  FFont.Color:=FColor;
  FFont.Name:='Arial';
  FFont.Charset:=DEFAULT_CHARSET;
  FFont.Pitch:=fpDefault;
  FFont.Size:=8;
end;

procedure TFontEngine2.InitFontStream;
var abmp:TBitmap;
begin
  FFontStream.Free;
  FFontStream:=nil;
  FFontStream:=TMemoryStream.Create;
  try
    abmp:=TBitmap.Create;
    DrawChars(abmp);
    abmp.SaveToStream(FFontStream);
  finally
    abmp.Free;
  end;
end;

procedure TFontEngine2.InitFontSurface;
var awidth,aheight:cardinal;
    abmp,abmpTexture,abmpfont:TBitmap;
    hr:HResult;
    dc:HDC;
begin
  FFontSurface:=nil;
```

```
abmp:=TBitmap.Create;
abmpFont:=TBitmap.Create;
try
 FFontStream.Seek(0,soFromBeginning);
 abmpFont.LoadFromStream(FFontStream);
 abmpFont.PixelFormat:=pf24Bit;

 awidth:=abmpFont.Width;
 aheight:=abmpFont.Height;
 abmp.Width:=aWidth;
 abmp.Height:=aHeight;

 abmp.Canvas.CopyMode:=cmSrcCopy;
 abmp.Canvas.Draw(0,0,abmpFont);
 if FUseTexture Then
 begin
  abmpTexture:=TBitmap.Create;
  try
   FFontTexture.Seek(0,soFromBeginning);
   abmpTexture.LoadFromStream(FFontTexture);
   abmpTexture.PixelFormat:=pf24Bit;
   abmp.Canvas.CopyMode:=cmSrcAnd;
   abmp.Canvas.Draw(0,0,abmpTexture);
  finally
   abmpTexture.Free;
  end;
 end;
 CreateOffScreenSurface(FFontSurface,awidth,aHeight);
 hr:=FFontSurface.GetDC(dc);
 if succeeded(hr) then
 begin
  BitBlt(dc,0,0,abmp.Width,abmp.Height,
       aBmp.Canvas.Handle,0,0,SRCCopy);
  FFontSurface.ReleaseDC(dc);
 end else raise EFontEngine2Error.Create('Error'+DDErrorString(hr));
```

```
    FSurfaceWidth:=aWidth;
    FSurfaceHeight:=aHeight;
  finally
    abmp.Free;
    abmpFont.Free;
  end;
end;

procedure TFontEngine2.InitFontTexture;
var abmp:TBitmap;
begin
  if FFontStream<>nil then
  begin
    FFontTexture.Free;
    FFontTexture:=nil;
    FFontTexture:=TMemoryStream.Create;
    abmp:=TBitmap.Create;
    try
      abmp.Width:=FFontStreamWidth;
      abmp.Height:=FFontStreamHeight;
      abmp.Canvas.Brush.Color:=FColor;
      abmp.Canvas.FloodFill(abmp.Width shr 1, abmp.Height shr 1,clBlack,fsBorder);
      abmp.SaveToStream(FFontTexture);
    finally
      abmp.Free;
    end;
  end;
end;

procedure TFontEngine2.NilPrivateVar;
begin
  FFont:=nil;
  FFontStream:=nil;
  FColor:=clWhite;
```

```
  FFontTexture:=nil;
  FFontSurface:=nil;
  FUseTexture:=false;
  ChangeColor:=false;
  FTextureStyle:=tsStretch;
end;

procedure TFontEngine2.RecreateFont(fnt: TFont);
begin
  if fnt<>nil then
    FFont.Assign(Fnt) else
  begin
    if ChangeColor then
    FFont.Color:=FColor;
  end;
  GetTextMetric;
  GetABCWidth;
  GetRects;
  InitFontStream;
  if FUseTexture=false then InitFontTexture;
  InitFontSurface;
end;

procedure TFontEngine2.ReloadSurface;
var awidth,aheight:cardinal;
    abmp,abmpTexture,abmpfont:TBitmap;
    hr:HResult;
    dc:HDC;
begin
  abmp:=TBitmap.Create;
  abmpFont:=TBitmap.Create;
  try
    FFontStream.Seek(0,soFromBeginning);
    abmpFont.LoadFromStream(FFontStream);
    abmpFont.PixelFormat:=pf24Bit;
```

```
  awidth:=abmpFont.Width;
  aheight:=abmpFont.Height;
  abmp.Width:=aWidth;
  abmp.Height:=aHeight;

  abmp.Canvas.CopyMode:=cmSrcCopy;
  abmp.Canvas.Draw(0,0,abmpFont);
  if FUseTexture Then
  begin
   abmpTexture:=TBitmap.Create;
   try
    FFontTexture.Seek(0,soFromBeginning);
    abmpTexture.LoadFromStream(FFontTexture);
    abmpTexture.PixelFormat:=pf24Bit;
    abmp.Canvas.CopyMode:=cmSrcAnd;
    abmp.Canvas.Draw(0,0,abmpTexture);
   finally
    abmpTexture.Free;
   end;
  end;
  hr:=FFontSurface.GetDC(dc);
  if succeeded(hr) then
  begin
   BitBlt(dc,0,0,abmp.Width,abmp.Height,
         aBmp.Canvas.Handle,0,0,SRCCopy);
   FFontSurface.ReleaseDC(dc);
  end else raise EFontEngine2Error.Create('Error'+DDErrorString(hr));

  FSurfaceWidth:=aWidth;
  FSurfaceHeight:=aHeight;
 finally
  abmp.Free;
  abmpFont.Free;
 end;
```

```
end;

procedure TFontEngine2.SetColor(const Value: TColor);
begin
  FColor := Value;
  ChangeColor:=true;
  RecreateFont(nil);
  ChangeColor:=false;
end;

procedure TFontEngine2.SetFont(const Value: TFont);
begin
  if value<>nil then
  begin
    RecreateFont(value);
  end;
end;

procedure TFontEngine2.SetTexture(const value: TBitmap);
var abmp:TBitmap;
begin
   abmp:=TBitmap.Create;
  try
    abmp.Width:=FFontStreamWidth;
    abmp.Height:=FFontStreamHeight;
    if (value.Width<>FFontStreamWidth) or
       (value.Height<>FFontStreamHeight) then
    begin
      if FTextureStyle=tsStretch then
        abmp.Canvas.StretchDraw(Rect(0,0,FFontStreamWidth,FFontStreamHeight),value);
    end else abmp.Assign(value);
    FFontTexture.Clear;
    abmp.SaveToStream(FFontTexture);
  finally
    abmp.Free;
```

```
  end;
end;

procedure TFontEngine2.SetTextureStyle(const Value: TTextureStyle);
begin
  FTextureStyle := Value;
end;

procedure TFontEngine2.SetUseTexture(const Value: boolean);
begin
  FUseTexture := Value;
  RecreateFont(nil);
end;

procedure TFontEngine2.WriteString(const x,y:integer;const txt: string);
var i,len,xx,yy:integer;
    sr,dr:TRect;
    c:byte;
    hr:HResult;
begin
  xx:=x;
  yy:=y;
  len:=Length(txt);
  for i:=1 to len do
  begin
    c:=Ord(txt[i]);
    sr:=FRects[c];
    inc(xx,FABCWidth[c].abcA);
    dr:=Rect(xx,yy,xx+sr.Right-sr.Left,yy+sr.Bottom-sr.Top);
    if FGraphicEngine.MyBackSurface.IsLost=DDERR_SURFACELOST then
       FGraphicEngine.MyBackSurface._Restore;
    if FFontSurface.IsLost=DDERR_SURFACELOST then
    begin
       hr:=FFontSurface._Restore;
       if Failed(hr) then ReloadSurface;
```

```
   end;
   FGraphicEngine.MyBackSurface.Blt(@dr, FFontSurface,@sr, DDBLT_WAIT or DDBLT_KEYSRC,
nil);
   inc(xx,FabcBPlusC[c]);
  end;
end;


end.
```

3.2 Kelas TGraphicEngine.

3.2.1 Pendahuluan.

Kelas TGraphicEngine adalah kelas yang akan kita gunakan untuk menyederhanakan proses inisialisasi DirectDraw, proses double buffering serta proses finalisasi DirectDraw. Kelas ini harus diciptakan sebelum menciptakan kelas-kelas lain seperti TSpriteEngine, TBackgroundEngine, TFontEngine dan TPrimitive.

3.2.2 Penyusunan Struktur Parameter.

Pada saat akan melakukan proses inisialisasi TGraphicEngine terlebih dahulu kita harus mendeskripsikan aplikasi yang akan kita buat kepada DirectDraw seperti handle aplikasi, full screen atau windowed, video mode, kedalaman warna dan jumlah backbuffer. Mengingat banyaknya parameter yang harus diinputkan, agar ringkas parameter-parameter tersebut kita gabung menjadi satu struktur. Deskripsi strukturnya adalah sebagai berikut:

```
type   TGraphicEngineParam=record
       Handle:HWND;
       Width,Height:integer;
       BitPerPixel:integer;
       BackBufferCount:integer;
       FullScreen:boolean;
       AllowReboot:boolean;
      end;
```

3.2.3 Inisialisasi

Inisialisasi dikerjakan dengan memanggil konstruktor Create milik kelas TGraphicEngine. Parameter inputnya adalah variabel bertipe struktur TGraphicEngineParam.

```
constructor TGraphicEngine.Create(GraphicEngineParam:TGraphicEngineParam);
```

Kita siapkan variabel-variabel sementara untuk menampung data-data yang kita perlukan.

```
var hr:HResult;
```

Untuk menampung status keberhasilan proses.

```
  ddsCap:TDDSCAPS;
```

Variabel yang menampung informasi kapabilitas surface yang akan dibuat.

```
  surfaceDesc:TDDSurfaceDesc;
```

Variabel yang menyimpan deskripsi surface

```
  cooplevel:dword;
```

Level kooperatif yang diinginkan

```
begin
 GEParam:=GraphicEngineParam;
```

Simpan parameter graphic engine karena informasi ini masih dibutuhkan.

```
 hr:=DirectDrawCreate(nil,MyDirectDraw,nil);
```

Ciptakan objek DirectDraw dari default device driver.

```
 if Succeeded(hr) then
 begin
```

Jika inisialisasi berhasil

```
  FCapability:=GetCaps;
```

Dapatkan kapabilitas DirectDraw dan simpan informasi kapabilitas. Prosesnya akan dijelaskan pada sub.bab

```
  case GraphicEngineParam.FullScreen of
   true:begin
```

Jika full screen maka jalankan rutin inisialisasi mode full screen.

3.2.3.1 Mode Full Screen.

```
  coopLevel:=DDSCL_FULLSCREEN or DDSCL_EXCLUSIVE;
```

Atur level kooperatif untuk mode full screen.

```
if GraphicEngineParam.AllowReboot then
  coopLevel:=coopLevel or DDSCL_ALLOWREBOOT;
```

Jika reboot difungsikan tambahkan level kooperatif DDSCL_ALLOWREBOOT ke level kooperatif.

```
hr:=MyDirectDraw.SetCooperativeLevel(GraphicEngineParam.Handle,coopLevel);
```

Atur level kooperatif

```
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
 Exit;
end;
```

Jika gagal timbulkan eksepsi EGraphicEngineError.

```
hr:=MyDirectDraw.SetDisplayMode(GraphicEngineParam.Width, GraphicEngineParam.Height,
GraphicEngineParam.BitPerPixel);
```

Ubah video mode sesuai deskripsi yang ada pada GraphicEngineParam.

```
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
 Exit
end;
```

Jika gagal timbulkan eksepsi EGraphicEngineError.

```
fillchar(surfacedesc,sizeof(surfacedesc),0);
surfacedesc.dwSize:=sizeof(surfacedesc);
surfacedesc.dwFlags:=DDSD_CAPS or DDSD_BACKBUFFERCOUNT;
surfacedesc.ddsCaps.dwCaps:=DDSCAPS_PRIMARYSURFACE or DDSCAPS_FLIP or
DDSCAPS_COMPLEX;
Surfacedesc.dwBackBufferCount:=GEParam.BackBufferCount;
```

Siapkan deskripsi surface untuk primary surface dengan kemampuan page flipping dan merupakan struktur surface kompleks dengan jumlah back buffer yang ada pada parameter GraphicEngineParam

```
hr:=MyDirectDraw.CreateSurface(Surfacedesc,MyPrimarySurface,nil);
```

Ciptakan primary surface

```
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
 Exit;
end;
```

Jika gagal timbulkan eksepsi EGraphicEngineError

```
ddsCap.dwCaps:=DDSCAPS_BACKBUFFER;
```

```
hr:=MyPrimarySurface.GetAttachedSurface(ddsCap,MyBackSurface);
```

Dapatkan pointer back surface

```
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
 Exit;
end;
```

Jika gagal timbulkan eksepsi EGraphicEngineError

```
MySrcRect:=Rect(0,0,GEParam.Width,GEParam.Height);
hr:=MyDirectDraw.CreateClipper(0,MyClipper,nil);
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Pointer(MyPrimarySurface):=nil;
  Pointer(MyBackSurface):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
  Exit;
end;
```

Ciptakan clipper untuk sprite clipping

```
SetClipRects([mySrcRect]);
MyBackSurface.SetClipper(MyClipper);
end;
```

3.2.3.2 Mode Windowed.

```
false:begin
```

Jika mode windowed

```
coopLevel:=DDSCL_NORMAL;
hr:=MyDirectDraw.SetCooperativeLevel(GraphicEngineParam.Handle, coopLevel);
```

Set level kooperatif DDSCL_NORMAL

```
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
 Exit;
```

```
end;
```

Jika gagal timbulkan eksepsi EGraphicEngineError

```
fillchar(surfacedesc,sizeof(surfacedesc),0);
surfacedesc.dwSize:=sizeof(surfacedesc);
surfacedesc.dwFlags:=DDSD_CAPS;
surfacedesc.ddsCaps.dwCaps:=DDSCAPS_PRIMARYSURFACE;
```

Siapkan deskripsi surface untuk primary surface

```
hr:=MyDirectDraw.CreateSurface(Surfacedesc,MyPrimarySurface,nil);
```

Ciptakan primary surface

```
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
 Exit;
end;
```

Jika gagal timbulkan eksepsi

```
CreateOffScreenSurface(MyBackSurface,GEParam.Width,GEParam.Height);
```

Ciptakan back surface.

```
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
  Exit;
end;
```

Jika gagal timbulkan eksepsi

```
MyDestRect:=Rect(0,0,0,0);
MySrcRect:=Rect(0,0,GEParam.Width,GEParam.Height);
hr:=MyDirectDraw.CreateClipper(0,WindowClipper,nil);
if Failed(hr) then
begin
  Pointer(MyDirectDraw):=nil;
  Pointer(MyPrimarySurface):=nil;
  Pointer(MyBackSurface):=nil;
  Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
```

```
        Exit;
      end;
      WindowClipper.SetHWnd(0,GEParam.Handle);
      MyPrimarySurface.SetClipper(WindowClipper);
```

Ciptakan clipper untuk mode windowed dan pasang ke primary surface.

```
      hr:=MyDirectDraw.CreateClipper(0,MyClipper,nil);
      if Failed(hr) then
      begin
        Pointer(MyDirectDraw):=nil;
        Pointer(MyPrimarySurface):=nil;
        Pointer(MyBackSurface):=nil;
        Raise EGraphicEngineError.Create('Error'+DDErrorString(hr));
        Exit;
      end;
      SetClipRects([MySrcRect]);
      MyBackSurface.SetClipper(MyClipper);
    end;
  end;
 end else Raise EGraphicEngineError.Create('Error:'+DDErrorString(hr));
end;
```

Terakhir adalah menciptakan clipper untuk proses sprite clipping serta memasangkan clipper ke back surface.

3.2.4 Mendapatkan Informasi Kemampuan yang Tersedia

Fungsi ini kita panggil pada saat inisialisasi kelas TGraphicEngine, berfungsi untuk mendapatkan informasi kemampuan yang tersedia.

```
function TGraphicEngine.GetCaps: TCapability;
var ddHALCaps,ddHELCaps:TDDCaps;
  hr:HResult;
  caps:TCapability;
```

Kita siapkan beberapa variabel lokal yang kita perlukan. *ddHALCaps* akan menyimpan informasi kemampuan yang tersedia pada perangkat keras, sedangkan *ddHELCaps* menyimpan informasi kemampuan yang tersedia pada software emulasi.

```
begin
 FillChar(ddHALCaps,sizeOf(TDDCaps),0);
```

```
FillChar(ddHELCaps,sizeOf(TDDCaps),0);

FillChar(caps,sizeOf(TCapability),0);
```

Inisialisasi ddHALCaps, ddHELCaps dan caps dengan nol

```
ddHALCaps.dwSize:=sizeOf(TDDCaps);

ddHELCaps.dwSize:=sizeOf(TDDCaps);
```

Isikan field dwSize ddHALCaps dan ddHELCaps dengan ukuran struktur TDDCaps

```
hr:=MyDirectDraw.GetCaps(@ddHALCaps,@ddHELCaps);
```

Panggil fungsi GetCaps milik interface IDirectDraw

```
if Failed(hr) then Raise EGraphicEngineError.Create('Error:'+DDErrorString(hr));
```

Jika gagal timbulkan eksepsi, jika berhasil isikan informasi kemampuan yang tersedia ke variabel caps

```
caps.HardwareBlt:=((ddHALCaps.dwCaps and DDCAPS_BLT)=DDCAPS_BLT);

caps.StretchBlt:=((ddHALCaps.dwCaps and DDCAPS_BLTSTRETCH)=DDCAPS_BLTSTRETCH);

caps.ColorKey:=((ddHALCaps.dwCaps and DDCAPS_COLORKEY)=DDCAPS_COLORKEY);

caps.ColorKeySrcBlt:=((ddHALCaps.dwCKeyCaps and
DDCKEYCAPS_SRCBLT)=DDCKEYCAPS_SRCBLT);

caps.HardwareClipping:=((ddHALCaps.dwCaps and DDCAPS_CANCLIP)=DDCAPS_CANCLIP);

caps.HardwareRotation:=((ddHALCaps.dwFXCaps and
DDFXCAPS_BLTROTATION)=DDFXCAPS_BLTROTATION);

caps.HardwareRotate90:=((ddHALCaps.dwFXCaps and
DDFXCAPS_BLTROTATION90)=DDFXCAPS_BLTROTATION90);

caps.HardwareMirrorUpDown:=((ddHALCaps.dwFXCaps and
DDFXCAPS_BLTMIRRORUPDOWN)=DDFXCAPS_BLTMIRRORUPDOWN);

caps.HardwareMirrorLeftRight:=((ddHALCaps.dwFXCaps and
DDFXCAPS_BLTMIRRORLEFTRIGHT)=DDFXCAPS_BLTMIRRORLEFTRIGHT);


caps.SoftwareBlt:=((ddHELCaps.dwCaps and DDCAPS_BLT)=DDCAPS_BLT);

caps.SoftwareStretchBlt:=((ddHELCaps.dwCaps and
DDCAPS_BLTSTRETCH)=DDCAPS_BLTSTRETCH);

caps.SoftwareColorKey:=((ddHELCaps.dwCaps and
DDCAPS_COLORKEY)=DDCAPS_COLORKEY);

caps.SoftwareColorKeySrcBlt:=((ddHELCaps.dwCKeyCaps and
DDCKEYCAPS_SRCBLT)=DDCKEYCAPS_SRCBLT);

caps.SoftwareClipping:=((ddHELCaps.dwCaps and DDCAPS_CANCLIP)=DDCAPS_CANCLIP);

caps.SoftwareRotation:=((ddHELCaps.dwFXCaps and
DDFXCAPS_BLTROTATION)=DDFXCAPS_BLTROTATION);

caps.SoftwareRotate90:=((ddHELCaps.dwFXCaps and
DDFXCAPS_BLTROTATION90)=DDFXCAPS_BLTROTATION90);
```

```
  caps.SoftwareMirrorUpDown:=((ddHELCaps.dwFXCaps and
DDFXCAPS_BLTMIRRORUPDOWN)=DDFXCAPS_BLTMIRRORUPDOWN);

  caps.SoftwareMirrorLeftRight:=((ddHELCaps.dwFXCaps and
DDFXCAPS_BLTMIRRORLEFTRIGHT)=DDFXCAPS_BLTMIRRORLEFTRIGHT);

  Result:=Caps;
```

Isikan nilai Caps ke Result

```
End;
```

Fungsi ini mengembalikan nilai bertipe *TCapability*,

```
    TCapability=Record
      HardwareBlt:boolean;
      ColorKey:boolean;
      ColorKeySrcBlt:boolean;
      StretchBlt:boolean;
      HardwareRotation:boolean;
      HardWareRotate90:boolean;
      HardwareMirrorUpDown:boolean;
      HardwareMirrorLeftRight:boolean;
      HardwareClipping:boolean;
      SoftwareBlt:boolean;
      SoftwareColorKey:boolean;
      SoftwareColorKeySrcBlt:boolean;
      SoftwareStretchBlt:boolean;
      SoftwareRotation:boolean;
      SoftWareRotate90:boolean;
      SoftwareMirrorUpDown:boolean;
      SoftwareMirrorLeftRight:boolean;
      SoftwareClipping:boolean;
    end;
```

Struktur data ini adalah struktur yang penulis ciptakan untuk memudahkan proses pengecekan kemampuan yang tersedia. Field-field yang memiliki nama dengan awalan *Software* adalah fungsi-fungsi yang tersedia di software emulasi, sedangkan field dengan awalan Hardware adalah fungsi yang tersedia di perangkat keras.

3.2.5 Rutin Double Buffering.

Rutin double buffering dikerjakan oleh metode *Show* kelas TGraphicEngine. Metode ini merupakan jantung dari proses animasi yang ditangani

oleh TGraphicEngine, oleh karena itu metode ini harus dibuat seringkas dan proses eksekusinya harus secepat mungkin untuk menghasilkan animasi yang halus.

```
procedure TGraphicEngine.Show;
var P:TPoint;
```

Kita siapkan variabel sementara *P* bertipe TPoint untuk menyimpan offset posisi window

```
begin
 if MyPrimarySurface.IsLost=DDERR_SURFACELOST then
   MyPrimarySurface._Restore;
 if MyBackSurface.IsLost=DDERR_SURFACELOST then
   MyBackSurface._Restore;
```

Cek apakah primary surface dan back surface hilang atau tidak. Jika hilang lakukan proses restorasi.

```
 case GEParam.FullScreen of
  true:begin
       MyPrimarySurface.Flip(nil,DDFLIP_WAIT);
      end;
```

Jika full screen maka lakukan page flipping,

```
  false:begin
       p:=Point(0,0);
       ClientToScreen(GEParam.Handle,p);
       GetClientRect(GEParam.Handle,MyDestRect);
       OffsetRect(MyDestRect,p.X,p.Y);
       MyPrimarySurface.Blt(@MyDestRect,MyBackSurface,
                  @MySrcRect,DDBLT_WAIT,nil);
      end;
 end;
end;
```

Jika windowed maka inisialisasi P dengan nol. Panggil *ClientToScreen* untuk mendapatkan koordinat window aplikasi relatif terhadap koordinat layar, koordinat ini kemudian disimpan di P. Dapatkan informasi client rectangle window aplikasi kita dengan memanggil *GetClientRect*, retangle ini kemudian kita simpan di *MyDestRect*. Geser MyDestRect dengan *OffsetRect*, sehingga *MyDestRect.Left* akan sama dengan *P.X* dan *MyDestRect.Top* sama dengan *P.Y* setelah fungsi ini dijalankan.

Setelah pemanggilan OffsetRect maka MyDestRect akan berisi informasi rectangle dimana kita harus melakukan proses blit. Blit back surface ke primary surface agar terlihat kelayar.

3.2.6 Finalisasi.

Proses finalisasi dilakukan dengan memanggil destruktor *Destroy*

```
destructor TGraphicEngine.Destroy;
begin
  if GEParam.FullScreen then
  begin
    MyDirectDraw.RestoreDisplayMode;
  end;
```

Jika full screen kembalikan video mode ke video mode semula.

```
  if Pointer(MyClipper)<>nil then Pointer(MyClipper):=nil;
```

Jika Clipper untuk sprite clipping tidak sama dengan nil maka bebaskan memori clipper.

```
  if Pointer(WindowClipper)<>nil then Pointer(WindowClipper):=nil;
```

Jika Clipper untuk mode windowed tidak sama dengan nil maka bebaskan memori windowclipper.

```
  Pointer(MyPrimarySurface):=nil;
```

Bebaskan memori primary surface

```
  Pointer(MyBackSurface):=nil;
```

Bebaskan memori back surface

```
  Pointer(MyDirectDraw):=nil;
```

Bebaskan memori objek DirectDraw

```
  inherited;
end;
```

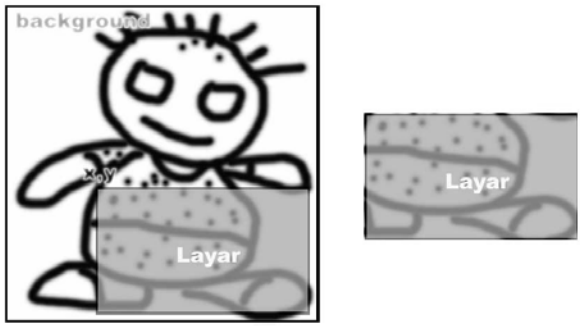
3.3 Kelas TBackgroundEngine.

3.3.1 Pendahuluan.

Kelas TBackgroundEngine adalah kelas yang fungsi utamanya untuk menangani proses background scrolling, walaupun demikian kelas ini juga dapat digunakan untuk menampilkan background statis yang tidak dapat digeser-geser.

Proses scrolling yang akan kita gunakan cara kerjanya dapat dijelaskan oleh gambar dibawah,

Gambar 3.1 Background scrolling posisi sebelum rectangle digeser.



Gambar 3.2 Background scrolling setelah rectangle digeser.

Pada kedua gambar di atas, gambar yang terletak di kiri merupakan gambaran cara kerja proses scrolling, sedangkan gambar yang di kanan adalah tampilan di layar yang akan dilihat oleh pengguna.

Untuk menghasilkan efek background yang dapat digeser kita akan menyediakan buffer yang akan menampung seluruh gambar. Gambar background harus lebih besar dari resolusi layar (untuk mode full screen) atau lebar dan tinggi window aplikasi (untuk mode windowed) untuk menghasilkan efek scrolling. Jika tidak maka gambar tidak akan discroll. Selain itu kita juga perlu menyiapkan sebuah variabel internal bertipe TRect untuk menyimpan informasi bagian gambar background yang akan ditampilkan. Semua yang berada dalam rectangle adalah apa yang akan dilihat oleh pengguna.

Proses scrollingnya sederhana, kita cukup menggeser rectangle ini untuk menghasilkan efek background yang bergeser.

3.3.2 Properti.

Kelas TBackgroundEngine diberi properti-properti sebagai berikut:

- *X* bertipe integer, properti ini properti published. Properti ini mencatat koordinat x relatif terhadap gambar background. Dengan mengubah properti ini maka kita dapat menggeser background secara horizontal. Untuk mengubah nilai properti ini digunakan prosedur *SetX*, sedangkan nilainya dibaca dari variabel privat *FX*.
- *Y* bertipe integer, sama dengan properti Y bersifat published, digunakan untuk mencatat koordinat y relatif terhadap gambar background. Untuk mengubah nilai properti ini digunakan prosedur *SetY*, sedangkan nilainya dibaca dari variabel privat *FY*.
- *Width* bertipe integer, bersifat published dan hanya dapat dibaca, digunakan untuk mencatat informasi lebar background.
- *Height* bertipe integer, bersifat published dan hanya dapat dibaca, digunakan untuk mencatat informasi tinggi background.
- *ScrollType* bertipe TScrollType, bersifat published, digunakan untuk menyimpan tipe scrolling yaitu scrolling normal atau kontinyu. Pada scrolling normal bila background telah habis maka proses scrolling dihentikan, sedangkan pada scrolling kontinyu, background akan diulang.
- *ScrollDir* bertipe TScrollDirection, bersifat published, digunakan untuk menyimpan arah scrolling yaitu scrolling horizontal, vertikal atau kedua-duanya.

Tipe TScrollType dan TScrollDirection deklarasinya adalah sebagai berikut:

```
TScrollType=(stNormal,stContinue);
TScrollDirection=(sdVertical,sdHorizontal,sdBoth);
```

3.3.3 Inisialisasi.

Proses ini dikerjakan oleh konstruktor *Create* milik kelas TBackgroundEngine.

```
constructor TBackgroundEngine.Create(AGraphicEngine: TGraphicEngine);
var bltfx:TDDBltFX;
begin
  ParentGraphicEngine:=AGraphicEngine;
  FX:=0;
  FY:=0;
  FWidth:=0;
  FHeight:=0;
  FScrollType:=stNormal;
  CreateOffScreenSurface(backgroundbuffer,
          ParentGraphicEngine.Parameter.Width,
```

```
        ParentGraphicEngine.Parameter.Height);

  ZeroMemory(@bltfx,SizeOf(TDDBltFX));
  bltFX.dwSize:=SizeOf(TDDBltFX);
  bltfx.dwFillColor:=0;
  BackgroundBuffer.Blt(nil,nil,nil,DDBLT_WAIT or DDBLT_COLORFILL,@bltfx);
end;
```

Create membutuhkan informasi graphic engine untuk menggambar backgroundnya. Informasi graphic engine ini disimpan di objek ParentGraphicEngine karena akan diperlukan pada saat melakukan proses penggambaran background ke back buffer.

```
  ParentGraphicEngine:=AGraphicEngine;
  FX:=0;
  FY:=0;
  FWidth:=0;
  FHeight:=0;
```

Pada konstruktor ini, dilakukan juga inisialisasi variabel-variabel internal yaitu FX, FY, FWidth, FHeight, FScrollType dan FScrollDir, serta melakukan proses penciptaan BackgroundBuffer. Lebar dan tinggi BackgroundBuffer kita asumsikan sama dengan lebar dan tinggi primary surface ParentGraphicEngine karena pada tahap ini kita belum tahu berapa lebar background yang akan digunakan.

Pada bagian akhir rutin ini kita isi warna BackgroundBuffer dengan warna hitam.

```
  ZeroMemory(@bltfx,SizeOf(TDDBltFX));
  bltFX.dwSize:=SizeOf(TDDBltFX);
  bltfx.dwFillColor:=0;
  BackgroundBuffer.Blt(nil,nil,nil,DDBLT_WAIT or DDBLT_COLORFILL,@bltfx);
```

3.3.4 Membaca File Background.

Format background yang digunakan adalah file BMP. Proses membaca file background dilakukan oleh *LoadFromFile*, parameternya adalah nama file BMP yang akan digunakan sebagai gambar background.

```
procedure TBackgroundEngine.LoadFromFile(const filename: string);
var bmp:TBitmap;
   DC:HDC;
   hr:HResult;
```

Kita siapkan variabel dan objek sementara. *DC* digunakan untuk menyimpan handel device context. Handel device context ini kita perlukan untuk proses mengkopi data bitmap dari objek *bmp* ke buffer yang menampung gambar background.

```
begin
 try
  if FileExists(filename) then
```

Kita lakukan tes apakah file BMP yang diinputkan ada,

```
  begin
   try
    bmp:=TBitmap.Create;
    bmp.LoadFromFile(filename);
```

Jika ada ciptakan objek bmp dan baca file bitmap,

```
    if bmp.Width<ParentGraphicEngine.Parameter.Width then
     FWidth:=ParentGraphicEngine.Parameter.Width else
     FWidth:=bmp.Width;
    if bmp.Height<ParentGraphicEngine.Parameter.Height then
     FHeight:=ParentGraphicEngine.Parameter.Height else
     FHeight:=bmp.Height;
```

Kita lakukan tes perbandingan lebar tinggi gambar background dengan lebar dan tinggi layar. Lebar dan tinggi buffer background adalah lebar dan tinggi yang lebih besar diantara keduanya.

```
 BackgroundBuffer:=nil;
 CreateOffScreenSurface(BackgroundBuffer,FWidth,FHeight);
```

Ciptakan buffer untuk background dengan lebar dan tinggi dari variabel FWidth dan FHeight

```
    hr:=BackgroundBuffer.GetDC(DC);
    if Failed(hr) then EBackgroundEngineError.Create('Error:'+DDErrorString(hr));
```

Dapatkan informasi handel devixe context buffer background. Jika gagal timbulkan eksepsi.

```
    BitBlt(DC,0,0,bmp.Width,bmp.Height,bmp.Canvas.Handle,0,0,SRCCOPY);
```

Kopikan data bitmap ke buffer background dengan fungsi BitBlt milik Windows API.

```
    hr:=BackgroundBuffer.ReleaseDC(DC);
    if Failed(hr) then EBackgroundEngineError.Create('Error:'+DDErrorString(hr));
```

Bebaskan handle device context buffer background. Sebagai catatan untuk fungsi GetDC dan ReleaseDC, sebaiknya handle device context segera kita bebaskan setelah handle device context tersebut tidak kita perlukan lagi.

```
  finally
    bmp.Free;
  end;
```

Akhirnya kita bebaskan memori yang dipakai objek bmp

```
 end else Raise EBackgroundEngineError.Create('Error:Background bitmap '+filename+' not found.');
```

Jika file BMP tidak ada, timbulkan eksepsi

```
 except
  Raise EBackgroundEngineError.Create('Error:');
 end;
end;
```

3.3.5 Proses Mengubah Properti X dan Y.

Properti X dan Y tidak bisa diubah sembarangan mengingat jika nilainya tidak valid dapat menyebabkan *access violation*. Oleh karena itu kita harus menguji nilai yang diinputkan terlebih dahulu sebelum mengubah nilai properti ini.

Untuk saat ini tipe scroll kontinyu belum didukung, mungkin pada edisi berikutnya kita akan menambahkan kemampuan scrolling secara kontinyu.

```
procedure TBackgroundEngine.SetX(const Value: integer);
begin
 case FScrollType of
  stNormal:begin
          if (Value>=0)and
            ((Value+ParentGraphicEngine.Parameter.Width)<FWidth) then
           FX := Value;
```

Kita lakukan tes apakah nilai yang ada di *Value* masih berada dalam background. Jika masih maka nilai yang ada di variabel Value dapat kita gunakan untuk nilai X yang baru.

Dengan cara ini, jika background telah habis maka background tidak akan dapat digeser lagi ke arah horizontal.

```
        end;
  stContinue:;
 end;
end;
```

```
procedure TBackgroundEngine.Sety(const Value: integer);
begin
  case FScrollType of
   stNormal:begin
            if (Value>=0)and
              ((Value+ParentGraphicEngine.Parameter.Height)<FHeight) then
             FY := Value;
```

Dengan cara yang sama kita lakukan tes apakah nilai yang ada di Value masih berada dalam background untuk properti Y.

```
          end;
   stContinue:;
  end;
end;
```

3.3.6 Proses Rendering Background ke Back Buffer.

Proses penggambaran background ke back buffer dikerjakan oleh metode Show milik kelas TBackgroundEngine.

```
procedure TBackgroundEngine.Show;
var aRect:TRect;
begin
  FClipRect:=Rect(FX,FY,
              FX+ParentGraphicEngine.GEParam.Width,
              FY+ParentGraphicEngine.GEParam.Height);
```

Siapkan FClipRect. Rectangle ini adalah rectangle sumber. Gambar apa yang ada dalam rectangle ini adalah apa yang akan terlihat oleh pengnuna

```
 aRect:=Rect(0,0,
            ParentGraphicEngine.GEParam.Width,
            ParentGraphicEngine.GEParam.Height);
```

Siapkan rectangle tujuan tempat pengkopian gambar yaitu ukuran seluruh layar.

```
  If BackgroundBuffer.IsLost=DDERR_SURFACELOST then
     BackgroundBuffer._Restore;
  If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
     ParentGraphicEngine.BackSurface._Restore;
```

Tes apakah buffer background dan back buffer hilang, jika hilang lakukan proses restorasi.

```
 ParentGraphicEngine.BackSurface.Blt(@aRect,BackgroundBuffer,@FClipRect,DDBLT_WAIT,nil)
```

Lakukan penggambaran buffer background ke back buffer.

```
end;
```

3.3.7 Finalisasi.

Proses finalisasi dikerjakan oleh destruktor *Destroy*. Pada destruktor ini kita bebaskan memori yang dipakai oleh buffer background.

```
destructor TBackgroundEngine.Destroy;
begin
  if BackgroundBuffer<>nil then
  begin
    Pointer(BackgroundBuffer):=nil;
  end;
  inherited;
end;
```

Prosedur *Free* kita gunakan untuk proses finalisasi yang aman.

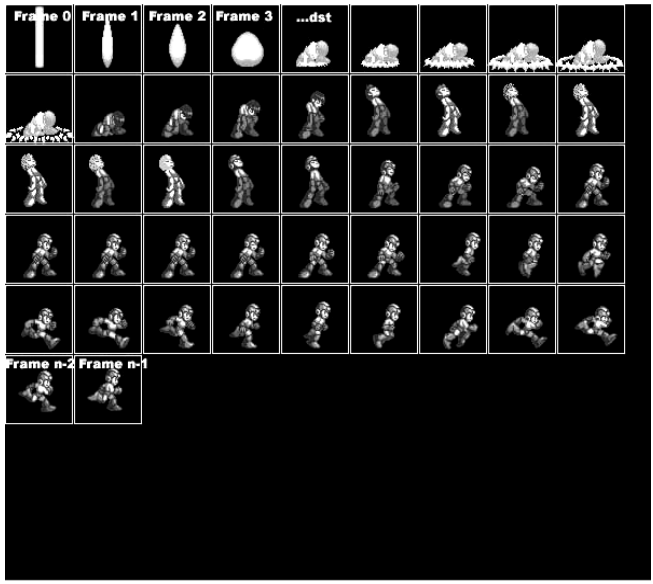
```
procedure TBackgroundEngine.Free;
begin
  if Self<>nil then Destroy;
end;
```

3.4 Kelas TSpriteEngine.

3.4.1 Pendahuluan.

Kelas ini penulis ciptakan untuk menangani proses penggambaran sprite. Kelas ini menyimpan semua informasi gambar sprite dalam satu surface besar, selain itu kelas ini juga membutuhkan informasi rectangle tiap-tiap gambar yang ada dalam surface. Dengan cara ini kita cukup mengindeks untuk mendapatkan rectangle dari suatu frame. Frame 0 bersesuaian dengan rectangle 0, Frame 1 bersesuaian dengan Rectangle 1 dan seterusnya. Rectangle-rectangle ini disimpan dalam sebuah array dinamis yang akan diinisialisasi saat membaca file data sprite.

Rectangle-rectangle ini ukurannya tidak harus sama antara frame yang satu dengan frame yang lainnya dengan demikian berbagai gambar yang ukurannya berbeda-beda dapat kita gabungkan dalam satu bitmap.



Gambar 3.3 Frame dalam surface sprite.

3.4.2 Properti.

Kelas TSpriteEngine memiliki properti-properti berikut:

- *FrameNow*, bertipe integer. Properti ini digunakan untuk mengakses frame yang akan ditampilkan saat ini.
- *Count*, bertipe integer. Properti ini mencatat banyaknya jumlah frame yang disimpan oleh kelas ini dan bersifat read-only.
- *Position*, bertipe TPosition. Properti ini mencatat posisi dimana sprite harus ditampilkan. Deklarasi TPosition adalah sebagai berikut:

```
TPosition=record
    X,Y,Z:integer;
end;
```

- *Velocity*, bertipe TVelocity mencatat informasi kecepatan gerak sprite. Deklarasi TVelocity adalah sebagai berikut:

```
TVelocity=record
    Vx,Vy,Vz:integer;
end;
```

- *Acceleration*, bertipe TAcceleration mencatat informasi percepatan gerak sprite. Deklarasi tipe TAcceleration adalah sebagai berikut:

```
TAcceleration=record
```

```
    Ax,Ay,Az:integer;
  end;
```

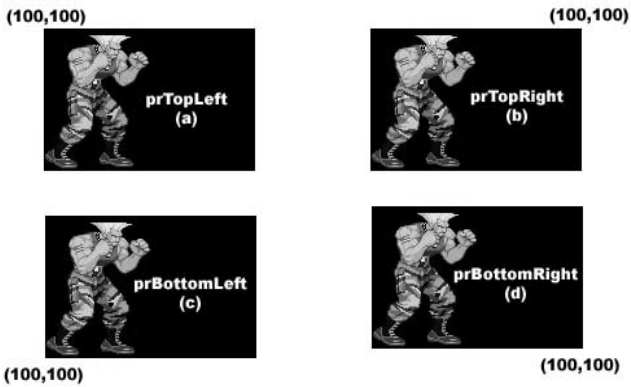
- *Direction*, bertipe Tdirection berguna untuk menyimpan informasi arah sprite, apakah menghadap ke kiri, ke kanan, atas dan sebagainya. Deklarasi TDirection adalah sebagai berikut:

```
TDirection=(dirLeft,dirRight,dirUp,dirDown,dirRotate180,dirRotate90,
            dirRotate270,dirRotate);
```

- *Status*, bertipe integer untuk menyimpan status sprite.
- *Width*, beripe integer untuk menyimpan informasi lebar sprite yang sedang ditunjuk oleh properti FrameNow. Properti ini bersifat read-only.
- *Height*, beripe integer untuk menyimpan informasi tinggi sprite yang sedang ditunjuk oleh properti FrameNow. Properti ini bersifat read-only.
- *RectNow*, bertipe TRect mencatat rectangle yang sedang ditunjuk oleh properti FrameNow. Properti ini juga bersifat read-only.
- *Angle*, bertipe integer berfungsi untuk merotasi sprite. Properti ini erat kaitannya dengan properti Direction. Jika Direction sama dengan *dirRotate* maka sprite akan dirotasi dengan sudut yang dibaca dari nilai properti Angle.
- *PosRef*, bertipe TPosRef. Properti ini mencatat posisi referensi sprite yang digunakan. Properti ini erat kaitannya dengan properti Position. Deklarasi TPosRef adalah sebagai berikut:

```
TPosRef=(prTopLeft,prTopRight,prBottomLeft,prBottomRight);
```

Penjelasan fungsi properti ini dapat dijelaskan melalui gambar berikut.



Gambar 3.4 Properti PosRef

Misalkan kita hendak menggambar sprite di atas ke suatu surface di koordinat (100,100). Jika kita mengisi PosRef dengan nilai prTopLeft maka titik kiri atas pada rectangle sprite akan diletakkan pada koordinat (100,100), jika PosRef sama dengan prBottomLeft maka titik kiri bawah yang akan diletakkan pada koordinat ini.

- *TransColor*, bertipe TRGB. Properti ini digunakan untuk membaca atau mengubah warna yang dianggap sebagai warna transparan. Deklarasi TRGB adalah sebagai berikut:

```
TRGB=record
  B,G,R,A:byte;
end;
```

3.4.3 Inisialisasi.

Inisialisasi kelas TSpriteEngine ditangani oleh konstruktor *Create*. Konstruktor ini membutuhkan informasi GraphicEngine meliputi properti BackSurface dan properti Parameter.

Pada konstruktor ini kita lakukan inisialisasi variabel-variabel internal milik kelas ini dengan nilai-nilai defaultnya.

```
constructor TSpriteEngine.Create(AGraphicEngine: TGraphicEngine);
begin
  FAngle:=0;
  FCount:=0;
  FFrameRect:=nil;
  FFrameNow:=0;
  FDirection:=dirLeft;
  FPosRef:=prTopLeft;
  FTransColor.B:=0;
  FTransColor.G:=0;
  FTransColor.R:=0;
  FTransColor.A:=0;
  FPosition:=SetPosition(0,0,0);
  FVelocity:=SetVelocity(0,0,0);
  FAcceleration:=SetAcceleration(0,0,0);
  Pointer(FSpriteBuffer):=nil;
  ParentGraphicEngine:=AGraphicEngine;
  FillChar(BltFX,sizeOf(TDDBLTFX),0);
  BltFX.dwSize:=sizeOf(TDDBLTFX);
  BltFX.dwDDFX:=DDBLTFX_MIRRORLEFTRIGHT;
end;
```

3.4.4 Format File Data Sprite.

Karena kita akan meletakkan beberapa gambar dalam satu surface maka kita membutuhkan informasi rectangle tiap-tiap gambar. Sebenarnya kita bisa saja membuat ukuran tiap rectangle sama, dengan cara ini kita cukup melakukan perhitungan untuk mengkonversi nomer frame yang ingin kita gambar menjadi rectangle yang bersesuaian dengan rectangle yang berisi gambar yang akan kita tampilkan.

Penulis tidak akan menggunakan cara di atas karena tidak fleksibel, mengingat semua rectangle sprite harus dibuat sama ukurannya. Penulis akan menggunakan array dinamis yang berisi rectangle-rectangle yang ada pada suatu surface. Dengan demikian kita cukup mengindeks rectangle berdasarkan nomer frame yang akan ditampilkan dan tidak perlu melakukan proses perhitungan untuk mengkonversi nomer frame menjadi informasi rectangle.

Karena proses penyusunan data sprite adalah proses yang terpisah dari engine game dan tidak dilakukan di program aplikasi game, maka kita perlu menyimpan data-data sprite agar bisa digunakan oleh engine game setelah data sprite ini siap. Untuk itu kita membutuhkan suatu format file data yang memudahkan proses pembacaan data sprite.

Penulis akan menggunakan file teks biasa sebagai file input karena mudah diedit ulang selain itu formatnya akan disusun sesederhana mungkin. Format file data tersebut adalah sebagai berikut:

<ID Pengenal='SPRITE'>

<Nama file bitmap yang berisi gambar sprite>

<Jumlah rectangle=N>

<Rectangle Ke-0.Left>

<Rectangle Ke-0.Top>

<Rectangles[0].Right>

<Rectangles[0].Bottom>

<Rectangles[1].Left>

<Rectangles[1].Top>

<Rectangles[1].Right>

<Rectangles[1].Bottom>

<Rectangles[2].Left>

<Rectangles[2].Top>

<Rectangles[2].Right>

<Rectangles[2].Bottom>

…..dst

<Rectangles[N-1].Left>

<Rectangles[N-1].Top>

<Rectangles[N-1].Right>

<Rectangles[N-1].Bottom>

Contoh file data sprite:

SPRITE

C:\My Documents\My Pictures\sprite.bmp

3

0

0

100

100

150

0

250

100

300

0

350

100

Yang perlu menjadi perhatian ketika mengedit file ini adalah antara baris yang satu dengan baris yang lainnya tidak boleh ada spasi dan string koordinat rectangle yang dimasukkan adalah string yang mewakili nilai integer, karena nantinya string ini akan dikonversi ke nilai integernya. Rutin pembaca file ini tidak melakukan pengecekan apakah nilai string ini valid untuk dikonversi atau tidak.

3.4.5 Membaca File Data Sprite.

Proses membaca file data sprite dikerjakan oleh *LoadFromFile* milik kelas TSpriteEngine. Parameter filename pada prosedur ini adalah nama file yang akan dibaca.

```
procedure TSpriteEngine.LoadFromFile(const filename: string);
var filestr:TextFile;
   ID,imagename,framecountstr:string;
   leftstr,rightstr,topstr,bottomstr:string;
   bmp:TBitmap;
   ctr:integer;
```

```
DC:HDC;
hr:HResult;
```

Kita siapkan variabel-variabel sementara, *FileStr* digunakan untuk proses pembacaan file, *ID*, *ImageName*, *FrameCountStr* digunakan untuk menyimpan hasil pembacaan pengenal, nama file dan jumlah rectangle yang berada di file. *LeftStr*,*RightStr*,*TopStr* dan *BottomStr* digunakan untuk menyimpan informasi hasil pembacaan rectangle dari file. *Bmp* akan menampung hasil pembacaan file bitmap yang namanya disimpan di variabel ImageName. *Ctr* adalah variabel yang berfungsi sebagai pencacah. *DC* adalah variabel yang menyimpan handle device context surface yang akan menampung data bitmap variabel Bmp. *Hr* adalah variabel status keberhasilan proses.

```
begin
 if FileExists(filename) then
 begin
```

Lakukan pengecekan apakah file yang diinputkan ada.

```
  AssignFile(filestr,filename);
  Reset(Filestr);
```

Jika ada, maka buka file tersebut.

```
  readln(filestr,ID);
```

Baca pengenal file yang terletak pada baris teratas.

```
  if ID='SPRITE' then
  begin
   readln(filestr,ImageName);
```

Tes apakah file yang sedang dibaca adalah berformat data sprite. Jika ya, baca baris selanjutnya yang berisi nama file bitmap sprite.

```
    if FileExists(Imagename) then
    begin
     try
      bmp:=TBitmap.Create;
      bmp.loadFromFile(imagename);
```

Jika file bitmap ini ada maka kita ciptakan objek Bmp dan menggunakan objek ini untuk membaca file bitmap.

```
      CreateOffScreenSurface(FSpriteBuffer,bmp.Width,bmp.Height);
```

Ciptakan offscreen surface untuk menampung gambar sprite, lebar dan tinggi surface sama dengan lebar dan tinggi bitmap.

```
      hr:=FSpriteBuffer.GetDC(DC);
```

Dapatkan handle device context surface dan simpan di DC.

```
    if Failed(hr) then
   begin
     raise ESpriteEngineError.Create('Error:'+DDErrorString(hr));
    exit;
   end;
```

Jika gagal timbulkan eksepsi.

```
    BitBlt(DC,0,0,bmp.Width,bmp.Height,bmp.Canvas.Handle,0,0,SRCCOPY);
```

Copy data bitmap ke surface

```
    hr:=FSpriteBuffer.ReleaseDC(DC);
```

Bebaskan handle device context

```
    if Failed(hr) then
  begin
    raise ESpriteEngineError.Create('Error:'+DDErrorString(hr));
    Exit;
  End;
```

Jika gagal timbulkan eksepsi.

```
    finally
     bmp.free;
```

Bebaskan memori Bmp karena sudah tidak diperlukan lagi.

```
    end;
   end else
  begin
    Raise ESpriteEngineError.Create('Error: Sprite bitmap '+imagename+' not found.');
    Exit;
 End;
```

Jika file bitmap tidak ada timbulkan eksepsi.

```
   readln(FileStr,framecountstr);
   FCount:=StrToInt(framecountStr);
```

Baca jumlah frame rectangle dalam file dan konversi ke nilai integernya.

```
   setLength(FframeRect,FCount);
```

Ubah jumlah indeks array *FFrameRect* menjadi FCount. FFrameRect adalah variabel internal berupa array dinamis yang menampung data tiap-tiap rectangle.

```
  for ctr:=0 to FCount-1 do
   begin
```

```
      readln(filestr,leftstr);
      FFrameRect[ctr].Left:=StrToInt(leftstr);
      readln(filestr,topstr);
      FFrameRect[ctr].Top:=StrToInt(topstr);
      readln(filestr,rightstr);
      FFrameRect[ctr].Right:=StrToInt(Rightstr);
      readln(filestr,bottomstr);
      FFrameRect[ctr].Bottom:=StrToInt(bottomstr);
    end;
```

Lakukan pembacaan data-data rectangle sebanyak FCount.

```
  end;
  Closefile(filestr);
```

Jika telah selesai tutup file data sprite.

```
 end else
begin
   raise ESpriteEngineError.Create('Error: Sprite file '+ filename+' not found.');
  exit;
```

Jika file input tidak ada timbulkan eksepsi.

```
end;
end;
```

3.4.6 Menciptakan Offscreen Surface Untuk Sprite.

Pada LoadFromFile terdapat rutin privat CreateOffscreenSurface yang digunakan untuk menciptakan surface yang akan menampung data gambar sprite. Untuk menciptakan offscreen surface caranya sama seperti pada proses penciptaan offscreen surface pada kelas TBackgroundEngine. Perbedaannya hanyalah adanya tambahan rutin untuk mengatur color key surface sprite.

```
procedure TSpriteEngine.CreateOffScreenSurface(
  out ASurface: IDirectDrawSurface; const AWidth, AHeight: Dword);
var surfaceDesc:TDDSurfaceDesc;
   hr:HResult;
   zz:TDDColorKey;
begin
  fillchar(surfacedesc,sizeof(surfacedesc),0);
  surfacedesc.dwSize:=sizeof(surfacedesc);
  surfacedesc.dwFlags:=DDSD_CAPS or DDSD_HEIGHT or DDSD_WIDTH;
```

```
 surfacedesc.ddsCaps.dwCaps:=DDSCAPS_OFFSCREENPLAIN;
 Surfacedesc.dwHeight:=AHeight;
 surfacedesc.dwWidth:=AWidth;
 hr:=ParentGraphicEngine.DirectDrawObject.CreateSurface(Surfacedesc,ASurface,nil);
 if Failed(hr) then raise ESpriteEngineError.Create('Error'+DDErrorString(hr));
 zz.dwColorSpaceLowValue:=0;
 zz.dwColorSpaceHighValue:=0;
```

Kita gunakan warna hitam sebagai warna transparan. Nantinya nilai ini dapat diubah.

```
 hr:=ASurface.SetColorKey(DDCKEY_SRCBLT,@zz);
```

Tambahkan informasi color key ke surface.

```
 if Failed(hr) then raise ESpriteEngineError.Create('Error'+DDErrorString(hr));
end;
```

Jika gagal timbulkan eksepsi.

3.4.7 Proses Rendering Ofscreen Surface Sprite ke Back Buffer.

Proses pengambaran dikerjakan oleh metode *Show* milik kelas TSpriteEngine. Rutin ini merupakan jantung dari proses animasi sprite.

```
procedure TSpriteEngine.Show;
begin
 case FDirection of
  dirLeft:_Show;
```

Jika properti Direction adalah dirLeft tampilkan gambar sprite tanpa efek blit

```
  dirRight:begin
        if (ParentGraphicEngine.Capability.HardwareMirrorLeftRight) or
          (ParentGraphicEngine.Capability.SoftwareMirrorLeftRight) then
          _ShowMirror;
       end;
```

Jika properti Direction berisi nilai dirRight maka tampilkan gambar dengan mencerminkan sprite kiri ke kanan. Lakukan pengecekan apakah fitur ini tersedia dengan melihat *ParentEngine.Capability.HardwareMirrorLeftRight* dan *ParentEngine.Capability.SoftwareMirrorLeftRight*.

```
  dirUp:_Show;
```

Jika properti Direction berisi nilai dirUp maka tampilkan gambar sprite tanpa efek blit seperti jika Direction berisi dirLeft.

```
dirDown:begin
        if (ParentGraphicEngine.Capability.HardwareMirrorUpDown) or
          (ParentGraphicEngine.Capability.SoftwareMirrorUpDown) then
          _showMirrorUpDown;
      end;
```

Jika properti Direction berisi nilai dirDown maka tampilkan gambar dengan mencerminkan sprite dari atas ke bawah.

```
dirRotate90:begin
        if (ParentGraphicEngine.Capability.HardwareRotate90) or
          (ParentGraphicEngine.Capability.SoftwareRotate90) then
          _ShowRotate90;
          end;
```

Jika properti Direction berisi nilai dirRotate90 maka tampilkan gambar dengan rotasi 90 derajat searah jarum jam.

```
dirRotate180:begin
          if (ParentGraphicEngine.Capability.HardwareRotate90) or
          (ParentGraphicEngine.Capability.SoftwareRotate90) then
          _ShowRotate180;
          end;
```

Jika properti Direction berisi nilai dirRotate180 maka tampilkan gambar dengan rotasi 180 derajat searah jarum jam.

```
dirRotate270:begin
          if (ParentGraphicEngine.Capability.HardwareRotate90) or
          (ParentGraphicEngine.Capability.SoftwareRotate90) then
           _ShowRotate270;
          end;
```

Jika properti Direction berisi nilai dirRotate270 maka tampilkan gambar dengan rotasi 270 derajat searah jarum jam.

```
dirRotate:begin
        if (ParentGraphicEngine.Capability.HardwareRotation) or
          (ParentGraphicEngine.Capability.SoftwareRotation) then
_ShowRotate(FAngle);
      end;
```

Jika properti Direction berisi nilai dirRotate maka tampilkan gambar dengan rotasi sembarang searah jarum jam.

```
end;
```

```
end;
```

3.4.8 Proses Rendering Sprite Tanpa Efek Blit.

Proses rendering tanpa efek blit dilakukan oleh metode *_Show*. Metode ini bersifat privat dan dipanggil oleh metode Show ketika melakukan rendering sprite.

```
procedure TSpriteEngine._Show;
var aRect:TRect;
   aWidth,aHeight:integer;
   aPos:TPosition;
```

Deklarasikan variable sementara *aRect* yang akan kita gunakan untuk menyimpan data rectangle tujuan dimana proses blit akan dilakukan. *aWidth* dan *aHeight* kita gunakan untuk menyimpan lebar dan tinggi sprite yang akan ditampilkan. Kita juga mendeklarasikan *aPos* yang bertipe TPosition yang akan menyimpan posisi peletakan sprite di back buffer.

```
begin
 If FSpriteBuffer.IsLost=DDERR_SURFACELOST then
    FSpriteBuffer._Restore;
 If ParentGraphicEngine.BackSurface.IsLost=DDERR_SURFACELOST then
    ParentGraphicEngine.BackSurface._Restore;
```

Kita lakukan tes untuk mengetahui surface yang kita akses apakah hilang atau tidak. Jika hilang kita lakukan proses restorasi.

```
 aWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
 aHeight:=FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top;
```

Hitung lebar dan tinggi sprite yang akan ditampilkan dengan membaca informasi rectangle yang di simpan di array *FFrameRect*

```
 aPos:=MapPosRefToActualPos(FPosition,FPosRef);
```

Konversi posisi sprite menjadi posisi aktualnya dengan mengacu pada properti PosRef. Data posisi actual ini kita simpan di aPos.

```
 aRect:=Rect(aPos.X,aPos.Y,
        aPos.X+AWidth,aPos.Y+aHeight);
```

Siapkan rectangle tujuan untuk proses blit.

```
ParentGraphicEngine.BackSurface.Blt(@aRect,FSpriteBuffer,@FFrameRect[FFRameNow],
                    DDBLT_WAIT or DDBLT_KEYSRC,
                    nil);
```

Lakukan proses blitting

```
end;
```

Proses rendering di atas membutuhkan informasi posisi actual dimana proses blit akan dilakukan. Proses untuk mengkonversi posisi relatif menjadi posisi aktual dikerjakan oleh metode *MapPosRefToActualPos*.

```
function TSpriteEngine.MapPosRefToActualPos(const aPos: TPosition;
  AposRef: TPosRef): TPosition;
begin
  result:=apos;
```

Asumsikan posisi aktual sama dengan posisi relatif.

```
  case aPosRef of
    prTopLeft:;
```

Jika posisi referensi bernilai prTopLeft maka kita tidak perlu melakukan apa-apa lagi karena posisi aktual sudah sama dengan posisi relatif.

```
    prTopRight:begin
          result.X:=aPos.X-(FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left);
        end;
```

Jika posisi referensi adalah prTopRight maka komponen X posisi aktual adalah komponen X posisi relatif dikurangi lebar sprite. Komponen Y posisi aktual sama dengan posisi Y posisi relatif.

```
    prBottomLeft:begin
          result.Y:=aPos.Y-(FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top);
        end;
```

Jika posisi referensi adalah prBottomLeft maka komponen Y posisi aktual adalah komponen Y posisi relatif dikurangi tinggi sprite. Komponen X posisi actual sama komponen X posisi relatif.

```
    prBottomRight:begin
          result.X:=aPos.X-(FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left);
          result.Y:=aPos.Y-(FFrameRect[FFrameNow].Bottom-FFrameRect[FFrameNow].Top);
        end;
  end;
```

Jika posisi referensi adalah prBottomRight maka komponen X dan Y posisi aktual adalah komponen X dan Y posisi relatif dikurangi lebar dan tinggi sprite.

```
end;
```

3.4.9 Proses Rendering Sprite Dengan Efek Blit.

Rendering sprite dengan efek blit dikerjakan oleh metode-metode *_ShowMirror*, *_ShowMirrorUpDown*, *_ShowRotate90*, *_ShowRotate180*, *_ShowRotate270* dan *_ShowRotate* . Proses yang dikerjakan oleh metode-metode ini

pada dasarnya hampir sama dengan proses yang dikerjakan oleh _Show, bagian-bagian rutin ini yang menangani proses blit menggunakan efek pencerminan dan rotasi tleah dijelaskan pada sub.bab mengenai efek blit sehingga penulis merasa tidak perlu lagi dijelaskan di sini.

3.4.10 Finalisasi.

Proses finalisasi dikerjakan oleh destruktor Destroy. Pada proses finalisasi surface yang menampung data gambar kita bebaskan memorinya demikian pula array dinamis FframeRect.

```
destructor TSpriteEngine.Destroy;
begin
  if FSpriteBuffer<>nil then
  begin
    Pointer(FSpriteBuffer):=nil;
  end;
  Finalize(FFrameRect);
  inherited;
end;
```

3.5 Kelas TFontEngine.

3.5.1 Pendahuluan.

Kelas TFontEngine adalah kelas turunan TSpriteEngine dan sesuai namanya, dimaksudkan untuk menangani proses penulisan teks ke surface. DirectDraw tidak memiliki fungsi-fungsi untuk menulis teks ke surface sehingga untuk menulis teks kita harus membuat rutin-rutinnya sendiri. Rutin-rutin ini dienkapsulasi menjadi kelas TFontEngine. Untuk saat ini kelas ini hanya menggunakan dua cara untuk menuliskan string yaitu dengan GDI dan dengan fungsi Blt menggunakan sprite.

Untuk menulis teks ke layar dengan GDI cukup mudah, hanya saja proses ini lambat, namun hasil tulisan di layar relatif bagus. Cara kedua adalah dengan sprite. Tiap karakter huruf kita buat spritenya, selanjutnya kita harus catat frame sprite ini kaitannya dengan huruf apa dalam suatu array. Untuk menampilkan teks, tiap-tiap karakter huruf dalam teks itu kita cari sprite yang bersesuaian dengan mengindeks array yang mencatat nomer frame. Cara ini relatif lebih sulit karena kita harus menyiapkan gambar bitmap yang berisi gambar-gambar huruf yang akan kita gunakan. Hasil tampilannya sangat tergantung kemampuan kita dalam membuat gambar huruf dan membuat data font sprite.

Data font sprite yang digunakan mirip dengan data sprite yang digunakan pada kelas TSpriteEngine. Perbedaannya adalah ada tambahan data asosisasi

karakter huruf dengan nomer frame yang bersesuaian. Data font sprite ini disimpan ke dalam file teks biasa agar dapat diedit dengan mudah.

3.5.2 Properti.

Kelas ini memiliki properti berikut:

- *UseGDI* Properti ini digunakan untuk mengubah proses penulisan teks ke surface. Jika UseGDI berisi nilai true maka proses penulisan teks menggunakan fungsi-fungsi GDI, bila false maka penulisan menggunakan sprite.
- *Color* warna teks. Properti ini hanya berpengaruh bila properti UseGDI bernilai true.

3.5.3 Inisialisasi.

Inisialisasi dilakukan oleh konstruktor Create. Prosesnya adalah dengan memanggil kontruktor pendahulunya yakni kelas TSpriteEngine. Pada konstruktor Create kita juga melakukan inisialisasi variabel-variabel internal kelas TFontEngine Floaded dan FUseGDI. Floaded adalah variabel internal yang mencatat apakah file data font sprite telah dibaca atau belum.

```
constructor TFontEngine.Create(AGraphicEngine: TGraphicEngine);
begin
  inherited Create(AGraphicEngine);
  FLoaded:=false;
  FUseGDI:=false;
end;
```

3.5.4 Format File Data Font Sprite.

File data font sprite seperti yang telah disebutkan di atas, susunannya mirip dengan file data sprite kelas TSpriteEngine., dimana format ini terbagi atas tiga blok besar yaitu header, blok data rectangle sprite dan blok huruf.

Blok header meliputi ID, nama file bitmap, jumlah rectangle dan jumlah huruf. Blok data rectangle meliputi daftar koordinat rectangle yang ada di sprite. Blok data huruf mencatat huruf yang ada dan hubungannya dengan frame.

Format file data tersebut adalah sebagai berikut:

<ID Pengenal='FONTSPR'>

<Nama file bitmap yang berisi gambar sprite>

<Jumlah rectangle=N>

<Jumlah huruf=Nhuruf>

<Rectangle Ke-0.Left>

<Rectangle Ke-0.Top>

<Rectangles[0].Right>

<Rectangles[0].Bottom>

<Rectangles[1].Left>

<Rectangles[1].Top>

<Rectangles[1].Right>

<Rectangles[1].Bottom>

<Rectangles[2].Left>

<Rectangles[2].Top>

<Rectangles[2].Right>

<Rectangles[2].Bottom>

…..dst

<Rectangles[N-1].Left>

<Rectangles[N-1].Top>

<Rectangles[N-1].Right>

<Rectangles[N-1].Bottom>

<Huruf Ke.0>

<Frame yang bersesuaian dengan Huruf Ke.0>

<Huruf Ke.1>

<Frame yang bersesuaian dengan Huruf Ke.1>

<Huruf Ke.2>

<Frame yang bersesuaian dengan Huruf Ke.2>

…dst

<Huruf Ke.Nhuruf-1>

<Frame yang bersesuaian dengan Huruf Ke.Nhuruf-1>

Contoh file data font sprite:

FONTSPR

C:\My Documents\My Pictures\sprite.bmp

3

3

0

0

100

100

150

0

250

100

300

0

350

100

A

0

B

1

C

2

Yang perlu menjadi perhatian ketika mengedit file ini adalah antara baris yang satu dengan baris yang lainnya tidak boleh ada spasi dan string koordinat rectangle yang dimasukkan adalah string yang mewakili nilai integer, karena nantinya string ini akan dikonversi ke nilai integernya. Rutin pembaca file ini tidak melakukan pengecekan apakah nilai string ini valid untuk dikonversi atau tidak.

Pada contoh di atas didefinisikan 3 huruf, yaitu A, B, C yang frame-framenya adalah 0, 1, 2.

3.5.5 Membaca File Data Font Sprite.

Untuk membaca file data font sprite digunakan metode LoadFromFile. Karena format filenya berbeda dengan kelas pendahulunya, kita tidak dapat menggunakan LoadFromFile milik kelas pendahulunya yaitu TSpriteEngine. Metode LoadFromFile ini akan kita buat sendiri.

```
procedure TFontEngine.LoadFromFile(const Filename: string);
var filestr:TextFile;
   ID,imagename,charcountstr,
   framecountstr,framestr:string;
   leftstr,rightstr,topstr,bottomstr:string;
   bmp:TBitmap;
```

```
  frameNo,i:integer;
  ch:char;
  DC:HDC;
  hr:HResult;
```

Kita deklarasikan beberapa variabel sementara yang kita perlukan untuk menampung data-data. *FileStr* adalah variabel yang mena,pung informasi file teks yang sedang dibaca. *Imagename* digunakan unuk menyimpan nama file bitmap. *CharCountStr* adalah variabel bertipe string yang akan menampung hasil pembacaan jumlah huruf yang ada dalam file teks. *FrameCountStr* menyimpan hasil pembacaan jumlah reactangle dalam file dan *FrameStr* mencatat nomer frame hasil pembacaan file. *LeftStr*, *RightStr*, *TopStr* dan *BottomStr* menyimpan hasil pembacaan koordinat rectangle.

```
begin
 if FileExists(filename) then
 begin
```

Tes apakah file yang hendak dibaca ada.

```
  AssignFile(filestr,filename);
  Reset(Filestr);
```

Jika ada buka file tersebut.

```
  readln(filestr,ID);
  if ID='FONTSPR' then
  begin
```

Baca dan lakukan tes apakah file ini berisi format font sprite.

```
   readln(filestr,ImageName);
```

Baca baris berikutnya yang berisi nama file bitmap dan kita simpan di variabel ImageName.

```
   if FileExists(Imagename) then
```

Kita tes lagi apakah file bitmap ini ada atau tidak.

```
   begin
    try
     bmp:=TBitmap.Create;
     bmp.loadFromFile(imagename);
```

Jika ada, kita ciptakan objek Bmp dan membaca file bitmap yang namanya disimpan di ImageName ke objek Bmp.

```
     CreateOffScreenSurface(FSpriteBuffer,bmp.Width,bmp.Height);
     hr:=FSpriteBuffer.GetDC(DC);
```

```
    if Failed(hr) then
   begin
     ESpriteEngineError.Create('Error:'+DDErrorString(hr));
    Exit;
  End;
    BitBlt(DC,0,0,bmp.Width,bmp.Height,bmp.Canvas.Handle,0,0,SRCCOPY);
    hr:=FSpriteBuffer.ReleaseDC(DC);
```

Kita buat surface penampung bitmap dan mengkopikan data bitmap ke surface.

```
    if Failed(hr) then
   begin
     ESpriteEngineError.Create('Error:'+DDErrorString(hr));
    Exit;
  End;
```

Jika proses mendapatkan dan membebaskan device context surface gagal timbulkan eksepsi.

```
    finally
     bmp.free;
    end;
    FLoaded:=true;
  end;
```

Bebaskan memori yang digunakan objek Bmp, karena objek ini tidak lagi kita perlukan.

```
  readln(FileStr,framecountstr);
  readln(FileStr,Charcountstr);
  FCount:=StrToInt(framecountStr);
  FCharCount:=StrToInt(framecountStr);
```

Baca jumlah rectangle dan jumlah huruf dan kita konversi ke nilai integernya.

```
  setLength(FFrameRect,FCount);
  setLength(CharList,FCharCount);
```

Kita tentukan ukuran array FframeRect dan CharList.

```
 for i:=0 to FCount-1 do
  begin
    readln(filestr,leftstr);
    FFrameRect[i].Left:=StrToInt(leftstr);
    readln(filestr,topstr);
```

```
    FFrameRect[i].Top:=StrToInt(topstr);
    readln(filestr,rightstr);
    FFrameRect[i].Right:=StrToInt(Rightstr);
    readln(filestr,bottomstr);
    FFrameRect[i].Bottom:=StrToInt(bottomstr);
   end;
```

Baca data seluruh rectangle

```
   for i:=0 to FCharCount-1 do
   begin
    readln(filestr,ch);
    readln(filestr,framestr);
    FrameNo:=StrToInt(frameStr);
    if (ch=#13) then ch:=#32;
    CharList[Ord(ch)-32]:=FrameNo;
   end;
```

Baca data semua huruf. Simpan informasi hubungan antara huruf dan nomer frame ke CharList. Karena huruf dibawah 32 (spasi) dalam karakter ASCII jarang sekali digunakan untuk keperluan menulis maka kita hanya akan menggunakan karakter huruf dari 32 ke atas. Konsekuensinya adalah kita harus mengurang nilai ordinal tiap karakter dengan 32 agar indeks bersesuaian dengan indeks 0 hingga FcharCount-1.

```
  end;
  Closefile(filestr);
```

Jika pembacaan telah selesai kita tutup file tersebut.

```
 end;
end;
```

3.5.6 Proses Menulis Teks.

Untuk menulis teks kita menggunakan metode WriteString milik kelas TFontEngine. Parameter metode ini adalah teks yang akan ditulis.

```
procedure TFontEngine.WriteString(const str: string);
var i,lenstr,chrWidth:integer;
  dc:HDC;
begin
 if FUseGDI then
 begin
```

Jika menggunakan GDI maka kita harus mendapatkan device context surface yang akan kita tulis teks.

```
ParentGraphicEngine.BackSurface.GetDC(dc);
SetBkColor(dc,TRANSPARENT);
```

Ubah warna latar menjadi transparan.

```
SetTextColor(DC,FColor);
```

Ganti warna teks dengan warna yang ada di variabel Fcolor.

```
SelectObject(DC,HandleFont);
```

Pilih font yang ingin digunakan dari handle font.

```
TextOut(dc,FPosition.X,FPosition.Y,PChar(str),length(str));
```

Tulis teks

```
ParentGraphicEngine.BackSurface.ReleaseDC(dc);
```

Bebaskan device context surface.

```
 end else
 if FLoaded then
 begin
  lenstr:=Length(str);
  for i:=1 to LenStr do
```

Jika menggunakan sprite maka kita dapatkan data panjang teks yang akan ditulis terlebih dahulu. Lakukan proses berikut ini sebanyak jumlah huruf dalam teks.

```
  begin
   FFrameNow:=CharList[Ord(str[i])-32];
```

Kita ambil data frame yang ada di CharList berdasarkan nilai ordinal dari karakter akan ditulis.

```
   chrWidth:=FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left;
```

Hitung lebar frame ini dan simpan di chrWidth.

```
   Show;
```

Tampilkan gambar karakter huruf ini ke back buffer.

```
   FPosition.x:=FPosition.x+chrWidth;
```

Tambahkan posisi X dengan chrWidth untuk menggambar karakter huruf berikutnya.

```
  end;
 end;
end;
```

3.5.7 Mendapatkan Informas Panjang Teks dalam Pixel.

Ada kalanya kita membutuhkan informasi panjang teks yang akan kita tulis dalam satuan piksel. Untuk keperluan ini kita menggunakan fungsi *GetLengthInPixel*. Fungsi ini menghitung panjang teks dengan menjumlahkan lebar tiap-tiap frame sprite yang bersesuaian dengan karakter teks yang sedang diindeks.

Karena prosesnya cukup mudah, maka penulis merasa hal ini tidak perlu dijelaskan terlalu detil. Pembaca bisa langsung memahami dari kode programnya.

```
function TFontEngine.GetLengthInPixel(const str: string): integer;
var i,lenstr,chrWidth:integer;
begin
  Result:=0;
  if FLoaded then
  begin
    chrWidth:=0;
    lenstr:=Length(str);
    for i:=1 to LenStr do
    begin
      FFrameNow:=CharList[Ord(str[i])-32];
      chrWidth:=chrWidth+FFrameRect[FFrameNow].Right-FFrameRect[FFrameNow].Left+1;
    end;
    Result:=chrWidth;
  end;
end;
```

3.5.8 Finalisasi.

Proses finalisasi seperti kelas-kelas lain ditangani oleh destruktor Destroy. Pada Destroy dilakukan pembebasan memori yang dipakai oleh Charlist serta handle font. Selanjutnya kita panggil destruktor pendahulunya.

```
destructor TFontEngine.Destroy;
begin
  if FUseGDI then
  DeleteObject(HandleFont);
 Finalize(CharList);
  inherited;
end;
```

3.6 Kelas TFontEngine2.

3.6.1 Pendahuluan.

Kelas ini adalah perbaikan kelas TFontEngine. Kekurangan terbesar TFontEngine adalah dibutuhkannya kesabaran dan keahlian untuk menghasilkan bitmap font yang bagus.

Dengan kelas TFontEngine2, maka bitmap ini dihasilkan secara otomatis dan kita tidak perlu menyediakan gambar sprite ini lagi. Untuk mengganti font, cukup dengan mengganti properti Font milik kelas TFontEngine2, maka tampilan huruf otomatis akan berubah.

Kekurangan kelas ini adalah style bold, italic beberapa font mungkin tidak dapat ditampilkan dengan benar.

TFontEngine2 sengaja penulis turunkan langsung dari kelas TObject (tidak seperti TFontEngine yang diturunkan dari TSpriteEngine) karena cara menampilkan gambar font tidak menggunakan nomer frame. Walaupun demikian prosesnya hampir sama karena sama-sama menggunakan sprite untuk menampilkan karakter.

3.6.2 Properti.

Kelas ini dilengkapi dengan beberapa properti yang digunakan untuk memodifikasi tampilan tulisan yaitu:

- Color, bertipe Tcolor, digunakan untuk menentukan warna tulisan. Properti ini menggunakan variabel internal Fcolor untuk menyimpan informasi warna. Untuk mengganti warna, dipanggil rutin *SetColor*. SetColor akan menciptakan ulang bitmap sprite dengan warna yang diinputkan.
- Texture, bertipe Tbitmap, digunakan untuk mengganti tampilan tulisan dengan gambar. Jika tekstur ini berisi gambar warna solid, maka efeknya sama dengan mengganti property. Jika anda hanya ingin mengganti warna teks, gunakan property Color karena prosesnya lebih cepat. Untuk mengganti property tekstur dipanggil SetTexture. SetTexture hanya akan mengganti objek FfontTexture yang bertipe TmemoryStream. Agar tampilan teks berubah maka property UseTexture harus diubah nmenjadi true.
- UseTexture, bertipe Boolean, mencatat status digunakannya tekstur. Jika true maka tekstur akan digunakan untuk menampilkan tulisan, jika false maka tampilan tulisan menggunakan warna solid yang ada di Color.
- TextureStyle, bertipe TTextureStyle, menyimpan informasi bagaimana tekstur harus ditampilkan. Ada tiga pilihan yakni tsStretch, tsTile dan tsStretchTile. Jika TextureStyle nilainya sama dengan tsStretch, maka gambar tekstur akan di-stretch sehingga berukuran sama dengan gambar sprite font. Jika tsTile maka tekstur akan digambar secara berulang (tile), sedangkan jika tsStretchTile, tekstur akan di-stretch sehingga sesuai dengan ukuran gambar satu karakter kemudian digambar berulang (tile).

- Font, bertipe Tfont, digunakan untuk mengubah tipe huruf. Jika anda membaca property font dan menyimpan alamatnya do objek font lain, maka objek tersebut jangan dibebaskan ketika sudah tidak diperlukan karena akan dibebaskan sendiri oleh kelas TfontEngine2, kecuali jika informasi propertyiFont ini di kopi ke objek font lain dengan rutin Assign milik kelas Tfont.

3.6.3 Inisialisasi.

Proses inisialisasi dikerjakan oleh konstruktor Create. Proses yang dikerjakan oleh Create adalah mengisi nilai-nilai awal variable internal, menciptakan objek font, objek-objek memory stream yang akan menampung bitmap font dan tekstur, serta surface. Kelas ini membutuhkan informasi alamat instance kelas TGraphicEngine, oleh karena itu alamat tersebut juga disimpan.

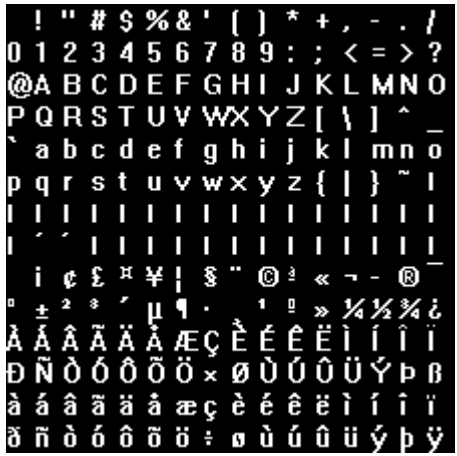
```
constructor TFontEngine2.Create(AGraphicEngine: TGraphicEngine);
begin
  FGraphicEngine:=AGraphicEngine;
  nilPrivateVar;
  InitFont;
  GetTextMetric;
  GetABCWidth;
  GetRects;
  InitFontStream;
  InitFontTexture;
  InitFontSurface;
end;
```

Create memanggil beberapa rutin internal untuk mendapatkan informasi font yakni *GetTextMetric*, *GetABCWidth* dan *GetRects*. Ketiga rutin ini akan dijelaskan pada sub bab berikutnya.

3.6.4 Mendapatkan Informasi Font.

Sebelum kita dapat menampilkan tulisan dengan benar, maka kita perlu mengetahui informasi font yang akan kita pergunakan. Informasi ini meliputi: tinggi, lebar maksimum karakter dan lebar ABC font. Apa itu lebar ABC akan penulis jelaskan berikutnya.

Font tersebut akan digambar dalam suatu bitmap seperti berikut,



Gambar 3.5 Bitmap Font.

Untuk mendapatkan tinggi dan lebar maksimum karakter, kita akan menggunakan fungsi Windows API, yakni GetTextMetric. Parameter fungsi ini adalah handle device context font dan variabel bertipe TTextMetric yang akan menampung data text metric font. Field TTextMetric yang kita perlukan tmHeight dan tmMaxCharWidth. Tinggi karakter untuk satu jenis font sama semua walau secara visual tampak berbeda. Lebar ABC dan deklarasi struktur TABC adalah berikut:

```
TABC=record
  abcA:integer;
  abcB:integer;
  abcC:integer;
end;
```

Field abcA menentukan jarak A karakter. Jarak A adalah suatu nilai yang harus ditambahkan ke posisi saat ini, sebelum menggambar karakter. Field abcB menentukan jarak B karakter. Jarak B adalah lebar bagian karakter yang digambar. Field abcC menentukan jarak C karakter. Jarak C adalah nilai yang ditambahkan ke posisi saat ini untuk menghasilkan pemisah pada sisi kanan karakter. Perhatikan gambar di bawah ini:

Gambar 3.6 Deskripsi lebar ABC.

Garis no.1 menandai posisi paling kiri huruf “W”, garis no.3 menandai lebar karakter “W”. Dapat kita lihat bahwa huruf “i” dimulai sebelum “W” benar-benar berakhir. GDI ketika menggambar teks akan memindahkan/memundurkan posisi penggambaran sebanyak beberapa piksel setelah selesai menggambar “W” dan sebelum menggambar “i”. Pada posisi inilah (garis no.2) karakter “i” mulai digambar . Jarak antara garis no.2 dan garis no.3 adalah jarak C.

Jumlah total lebar sebuah karakter adalah penjumlahhan jarak A, B, C. Oleh karena itu untuk mendapatkan lebar sebuah teks dalam piksel (fungsi GetTextWidthInPixel) kita cukup melakukan looping tiap karakter dalam teks dan menjumlahkan jarak A, B, C. Hasilnya adalah lebar teks tersebut dalam piksel. Jarak A dan C dapat bernilai positif atau negatif.

Untuk mendapatkan lebar ABC font kita menggunakan fungsi GetCharABCWidths. Paramater fungsi ini adalah handle device context font, karakter pertama yang akan diambil data lebar ABC, karakter terakhir yang akan diambil lebar ABC, array ABC yang akan menampung informasi ABC.

Fungsi GetCharABCWidths(dc:HDC;firstchar,lastchar:byte;var abc:array of TABC);

dc

Handle device context font.

firstchar

Karakter pertama yang akan diambil informasi lebar ABC.

lastchar

Karakter terakhir yang akan diambil informasi lebar ABC.

abc

Array ABC yang menampung informasi lebar ABC.

Pada kelas TFontEngine2 fungsi-fungsi yang digunakan untuk mendapatkan informasi font adalah GetTextMetric dan GetCharWidth.

```
procedure TFontEngine2.GetTextMetric;
var dc:HDC;
begin
 if FFont.Handle<>0 then
 begin
```

Jika font kita berisi data font yang valid, maka kita ciptakan handle device context.

```
  dc:=CreateCompatibleDC(0);
  SelectObject(dc,FFont.Handle);
```

Kita asosiasikan font kita dengan handle device context yang baru kita buat.

```
  GetTextMetrics(dc,FTextMetric);
```

Panggil fungsi Windows API GetTextMetrics untuk mendapatkan informasi textmetric font yang informasi device contextnya disimpan di variabel dc.

```
  DeleteDC(dc);
```

Hapus handle device context karena sudah tidak diperlukan lagi.

```
  FFontStreamWidth:=FTextMetric.tmMaxCharWidth shl 4;
  FFontStreamHeight:=FTextMetric.tmHeight*14;
```

Kita hitung lebar dan tinggi stream bitmap font. Kita lakukan perkalian dengan 16 (shl 4) dan perkalian dengan 14 karena kita akan menyusun karakter-karakter dalam bitmap sebanyak 16x14 (lihat Gambar 3.5).

```
 end;
end;
```

Berikut ini adalah rutin GetABCWidth. Fungsi ini digunakan untuk mendapatkan informasi lebar ABC tiap karakter.

```
procedure TFontEngine2.GetABCWidth;
var dc:HDC;
  i:integer;
  sz:TSize;
  ch:Char;
  pch:PAnsiChar;
begin
 if FFont.Handle<>0 then
 begin
```

Jika font kita berisi data font yang valid, maka kita ciptakan handle device context.

```
  dc:=CreateCompatibleDC(0);
```

Kita asosiasikan font kita dengan handle device context yang baru kita buat.

```
  SelectObject(dc,FFont.Handle);
  if GetCharABCWidths(dc,0,255,FABCWidth[0])=false then
  begin
```

Kita panggil fungsi Windows API GetCharABCWidths untuk mendapatkan lebar ABC semua karakter. Jika nilai yang dikembalikan fungsi ini adalah false berarti font bukan font TrueType (karena hanya fontTrueType yang menggunakan lebar ABC) , oleh karena itu kita harus mendapatkan lebar karakter dengan cara lain.

```
   FillChar(FABCWidth[0],SizeOf(TABC) shl 8,0);
```

Kita asumsikan lebar ABC sama dengan nol. Field yang kita gunakan untuk menyimpan lebar tiap karakter adalah abcB, field lainnya kita isikan nol.

```
   for i:=32 to 255 do
```

```
  begin
```

Kita lakukan looping dari mulai karakter dengan kode ACII 32 (spasi) hingga karakter ASCII 255. Karakter dibawah 32 tidak kita pergunakan karena karakter tersebut biasanya tidak terlihat.

```
    ch:=Chr(i);
    pch:=@ch;
    GetTextExtentPoint32(dc,pch,1,sz);
```

Kita ambil lebar karakter yang ada di pch dengan fungsi Windows API GetTextExtentPoint32. Informasi lebar karakter ini kita simpan di sz yang bertipe Tsize. Setelah pemaggilan fungsi ini maka field cx akan berisi lebar karakter yang kita cari. Kita isikan nilai ini ke field abcB.

```
    FABCWidth[i].abcB:=sz.cx;
   end;
  end;
  DeleteDC(dc);
```

Kita hapus device context.

```
 end;
end;
```

Fungsi GetRects digunakan untuk mendapatkan rectangle-rectangle tiap karakter dalam bitmap font, serta mengisi array FBPlusC yang merupakan penjumlahan field abcB dan abcC tiap karakter.Penulis menghitung nilai-nilai ini agar tidak perlu melakukan proses penjumlahan field abcB dan abcC lagi pada saat menampilkan teks. Array FBPlusC dan FRects ini berperan penting dalam proses menulis teks ke layar.

```
procedure TFontEngine2.GetRects;
var i,cellWidth,cellHeight:integer;
begin
 CellWidth:=FFontStreamWidth shr 4;
 CellHeight:=FFontStreamHeight div 14;
Kita hitung jumlah total karakter horizontal dan vertical dalam bitmap font.
 for i:=32 to 255 do
 begin
```

Kita lakukan looping dari karakter ASCII 32 hingga 255 untuk mendapatkan rectangle tiap karakter dan nilai penjumlahan abcB dan abcC.

```
  FRects[i].Left:=((i-32) mod 16)*CellWidth;
  FRects[i].Top:=((i-32) shr 4)*CellHeight;
  FRects[i].Right:=FRects[i].Left+FABCWidth[i].abcB;
```

```
    FRects[i].Bottom:=FRects[i].Top+CellHeight;
    FabcBPlusC[i]:=FABCWidth[i].abcB+FABCWidth[i].abcC;
  end;
end;
```

3.6.5 Menampilkan Gambar Karakter ke dalam Bitmap Font.

Proses ini dikerjakan oleh rutin *DrawChars*. DrawChars akan menggambar tiap karakter mulai dari posisi paling kiri pada tiap sel rectangle sehingga dihasilkan bitmap seperti Gambar 3.5.

```
procedure TFontEngine2.DrawChars(abitmap: TBitmap);
var i,j:integer;
    c:byte;
    str:string;
begin
  abitmap.Width:=FFontStreamWidth;
  abitmap.Height:=FFontStreamHeight;
  abitmap.Canvas.Brush.Color:=clBlack;
  abitmap.Canvas.Font:=FFont;
  abitmap.Canvas.FloodFill(abitmap.width shr 1,abitmap.Height shr 1,clBlack,fsBorder);
  SetBkMode(abitmap.Canvas.Handle,TRANSPARENT);
Kita isi warna latar belakang bitmap dengan warna hitam.
  for i:=0 to 13 do
  for j:=0 to 15 do
  begin
```

Kita lakukan looping sebanyak 224 karakter (16x14 karakter)

```
    c:=(i shl 4)+j+32;
```

Untuk tiap-tiap sel kita hitung kode ASCII karakter yang bersesuaian. Contoh untuk kolom 1 (j=1) baris 2 (i=2) maka kode ASCI yang bersesuaian adalah c=(2 shl 4)+1+32=65. Kode ASCII 65 adalah karakter “2”. Adanya penjumlahan dengan 32 karena kita menggunakan karakter spasi (ASCII 32) sebagai karakter pertama yang kita gambar.

```
    str:='';
    str:=str+chr(c);
    abitmap.Canvas.TextOut(j*FTextMetric.tmMaxCharWidth-FABCWidth[c].abcA,
          i*FTextMetric.tmHeight,str);
```

Kita gambar karakter dengan fungsi TextOut milik Canvas TBitmap. Jika anda tidak menggunakan kelas TBitmap, maka anda dapat menggambar karakter dengan fungsi

Windows API *TextOut*. Posisi koordinat x dimana kita harus menggambar karakter dihitung dengan mengalikan kolom dengan lebar karakter maksimum (tmCharWidth) dikurangi dengan lebar abcA. Tujuannya agar karakter digambar rata kiri terhadap rectangle sel-nya. Sedangkan posisi koordinat y diperoleh dengan mengalikan baris dan tinggi karakter (tmHeight).

```
  end;
end;
```

3.6.6 Menciptakan Bitmap Font.

Bitmap font akan disimpan dalam memory stream. Stream yang digunakan untuk menyimpan bitmap adalah FFontStream bertipe TMemoryStream. Alasan penulis menggunakan stream adalah untuk menghindari penggunaan TBitmap. Penggunaan TBitmap yang terlalu banyak dapat menyebabkan Windows kehabisan resource, mengingat handle bitmap adalah resource yang terbatas jumlahnya. Kekurangan model seperti ini adalah kita harus bolak-balik membaca stream tiap kali kita hendak mengakses data bitmap. Pembaca dipersilahkan memodifikasinya jika tidak ingin menggunakan stream. Proses menciptakan bitmap font dikerjakan oleh rutin *InitFontStream*.

```
procedure TFontEngine2.InitFontStream;
var abmp:TBitmap;
begin
  FFontStream.Free;
  FFontStream:=nil;
  FFontStream:=TMemoryStream.Create;
```

Ciptakan ulang stream.

```
  abmp:=TBitmap.Create;
```

Ciptakan bitmap sementara.

```
  try
    DrawChars(abmp);
```

Gambar semua karakter ke dalam bitmap sementara.

```
    abmp.SaveToStream(FFontStream);
```

Simpan data yang ada di bitmap sementara ke stream.

```
  finally
    abmp.Free;
  end;
```

Bebaskan memory bitmap sementara karena tidak diperlukan lagi.

```
end;
```

Sebagai catatan, data yang ada di FFontStream adalah data-data sprite yang tidak akan diubah-ubah lagi, kecuali jika property font diganti dengan data font yang lain.

3.6.7 Menciptakan Tekstur.

Proses menciptakan tekstur dikerjakan oleh *InitFontTexture*. Sebelum menciptakan tekstur, kita harus mengetahui apakah stream bitmap font sudah ada atau belum. Jika sudah ada, maka stream tekstur kita ciptakan ulang.

```
procedure TFontEngine2.InitFontTexture;
var abmp:TBitmap;
begin
  if FFontStream<>nil then
  begin
    FFontTexture.Free;
    FFontTexture:=nil;
    FFontTexture:=TMemoryStream.Create;
    abmp:=TBitmap.Create;
    try
      abmp.Width:=FFontStreamWidth;
      abmp.Height:=FFontStreamHeight;
      abmp.Canvas.Brush.Color:=FColor;
      abmp.Canvas.FloodFill(abmp.Width shr 1, abmp.Height shr 1,clBlack,fsBorder);
      abmp.SaveToStream(FFontTexture);
    finally
      abmp.Free;
    end;
  end;
end;
```

3.6.8 Menciptakan Surface Sprite Font.

Proses penciptaan surface untuk sprite font sama dengan proses penciptaan sprite pada kelas TSpriteEngine.

```
procedure TFontEngine2.InitFontSurface;
var awidth,aheight:cardinal;
    abmp,abmpTexture,abmpfont:TBitmap;
    hr:HResult;
```

```
    dc:HDC;
  begin
   FFontSurface:=nil;
   abmp:=TBitmap.Create;
   abmpFont:=TBitmap.Create;
   try
    FFontStream.Seek(0,soFromBeginning);
    abmpFont.LoadFromStream(FFontStream);
    abmpFont.PixelFormat:=pf24Bit;

    awidth:=abmpFont.Width;
    aheight:=abmpFont.Height;
    abmp.Width:=aWidth;
    abmp.Height:=aHeight;

    abmp.Canvas.CopyMode:=cmSrcCopy;
    abmp.Canvas.Draw(0,0,abmpFont);
    if FUseTexture Then
    begin
     abmpTexture:=TBitmap.Create;
     try
      FFontTexture.Seek(0,soFromBeginning);
      abmpTexture.LoadFromStream(FFontTexture);
      abmpTexture.PixelFormat:=pf24Bit;
      abmp.Canvas.CopyMode:=cmSrcAnd;
      abmp.Canvas.Draw(0,0,abmpTexture);
     finally
      abmpTexture.Free;
     end;
    end;
    CreateOffScreenSurface(FFontSurface,awidth,aHeight);
    hr:=FFontSurface.GetDC(dc);
    if succeeded(hr) then
    begin
     BitBlt(dc,0,0,abmp.Width,abmp.Height,
```

```
        aBmp.Canvas.Handle,0,0,SRCCopy);
     FFontSurface.ReleaseDC(dc);
    end else raise EFontEngine2Error.Create('Error'+DDErrorString(hr));


    FSurfaceWidth:=aWidth;
    FSurfaceHeight:=aHeight;
   finally
    abmp.Free;
    abmpFont.Free;
   end;
end;
```

3.6.9 Menulis Teks.

Proses menulis teks ke back buffer dikerjakan oleh rutin *WriteString*. Berbeda dengan WriteString milik TFontEngine yang hanya membutuhkan satu parameter yaitu, teks yang akan ditulis, WriteString milik TFontEngine2 memiliki 3 parameter yaknim parameter koordinat x,y dimana teks akan ditulis dan string yang berisi teks yang hendak ditulis.

```
procedure TFontEngine2.WriteString(const x,y:integer;const txt: string);
var i,len,xx,yy:integer;
    sr,dr:TRect;
    c:byte;
    hr:HResult;
begin
  xx:=x;
  yy:=y;
  len:=Length(txt);
```

Kita simpan panjang teks dan koordinat penulisan teks.

```
  for i:=1 to len do
  begin
```

Kita lakukan proses looping untuk tiap-tiap karakter dalam teks. Kita mulai dari i=1 karena penyimpanan data karakter pada string dimulai dari indeks ke-1.

```
    c:=Ord(txt[i]);
    sr:=FRects[c];
```

Kita cari nilai ordinal karakter yang sedang kita akses. Nilai ordinal ini kita gunakan untuk mencari rectangle pada bitmap font yang bersesuaian dengan karakter yang sedang kita proses. Rectangle yang kita dapatkan kita simpan di variabel sementara

sr. Variabel ini nantinya merupakan rectangle sumber (source rectangle) untuk proses blit.

```
inc(xx,FABCWidth[c].abcA);
```

Kita tambahkan koordinat x penulisan dengan field abcA karakter yang sedang diproses sebelum mulai melakukan proses penggambaran sprite karakter.

```
dr:=Rect(xx,yy,xx+sr.Right-sr.Left,yy+sr.Bottom-sr.Top);
```

Setelah itu kita hitung rectangle tujuan (destination rectangle). Ukuran lebar dan tinggi rectangle ini sama persis dengan rectangle sumbe, namun posisi kiri atasnya terletak pada koordinat xx,yy.

```
if FGraphicEngine.MyBackSurface.IsLost=DDERR_SURFACELOST then
  FGraphicEngine.MyBackSurface._Restore;
```

Perlu dilakukan pengecekan terlebih dahulu untuk mengetahui apakah back surface dan font surface tidak hilang. Jika hilang kita lakukan proses restorasi surface.

```
if FFontSurface.IsLost=DDERR_SURFACELOST then
begin
  hr:=FFontSurface._Restore;
  if Failed(hr) then ReloadSurface;
```

Jika proses restorasi gagal maka kita load ulang data font sprite dari memori sistem ke surface dengan memanggil rutin *ReloadSurface*.

```
end;
FGraphicEngine.MyBackSurface.Blt(@dr,FFontSurface,@sr,DDBLT_WAIT or
DDBLT_KEYSRC,nil);
```

Kita gambar sprite karakter ke back buffer. Proses penggambaran ini tanpa menggunakan efek blit. Para pembaca dipersilahkan memodifikasi bagian ini jika ingin menambahkan efek blit ke teks seperti rotasi atau mirror.

```
inc(xx,FabcBPlusC[c]);
```

Tambahkan koordinat x dengan nilai abcB+abcC karakter yang sedang diproses. Setelah proses ini maka, kita telah siap menulis karakter berikutnya. Proses looping ini dikerjakan lagi sampai semua karakter dalam teks selesai diproses.

```
 end;
end;
```

Di bawah ini adalah rutin ReloadSurface yang berfungsi untuk mengkopikan data sprite font yang ada di memori sistem ke surface.

```
procedure TFontEngine2.ReloadSurface;
var awidth,aheight:cardinal;
  abmp,abmpTexture,abmpfont:TBitmap;
  hr:HResult;
```

```
  dc:HDC;
begin
 abmp:=TBitmap.Create;
 abmpFont:=TBitmap.Create;
```

Kita ciptakan bitmap sementara yang akan menampung sprite font tanpa tekstur dan sprite font yang diberi tekstur.

```
 try
  FFontStream.Seek(0,soFromBeginning);
  abmpFont.LoadFromStream(FFontStream);
  abmpFont.PixelFormat:=pf24Bit;
```

Pindah posisi penunjuk stream font ke awal stream. Tujuannya agar seluruh stream benar-benar dikopi ke bitmap font. Kita baca stream dan menyimpannya ke bitmap dengan metode LoadFromStream kelas TBitmap.Format pixel bitmap ini kita ubah ke 24 bit.

```
  awidth:=abmpFont.Width;
  aheight:=abmpFont.Height;
  abmp.Width:=aWidth;
  abmp.Height:=aHeight;
```

Bitmap yang akan kita beri tekstur kita ubah dimensinya agar sama dengan bitmapfont tanpa tekstur.

```
  abmp.Canvas.CopyMode:=cmSrcCopy;
  abmp.Canvas.Draw(0,0,abmpFont);
```

Bitmap font tanpa tekstur kita gambar ke bitmap ini. Properti CopyMode Canvas bitmap ini kita ubah menjadi cmSrcCopy. Setelah proses ini, abmp akan berisi data yang sama dengan abmpFont.

```
  if FUseTexture Then
  begin
```

Jika properti UseTexture sama dengan true maka kita perlu menyiapkan sebuah bitmap lagi yang akan menampung tekstu.

```
    abmpTexture:=TBitmap.Create;
    try
     FFontTexture.Seek(0,soFromBeginning);
     abmpTexture.LoadFromStream(FFontTexture);
     abmpTexture.PixelFormat:=pf24Bit;
```

Stream tesktur kita kopi ke bitmap texture. Format pixel bitmap ini kita samakan dengan format pixel bitmap font.

```
     abmp.Canvas.CopyMode:=cmSrcAnd;
     abmp.Canvas.Draw(0,0,abmpTexture);
```

Properti CopyMode kita ubah menjadi cmSrcAnd. Dengan cara ini maka pada proses pengkopian bitmap akan dikerjakan operasi AND antara bitmap font dengan bitmap tekstur. Bitmap font sebelumnya hanya berisi warna hitam dan warna putih. Operasi AND antara warna hitam (warna 0) dengan warna apa pun akan menghasilkan warna hitam., sedangkan operasi AND warna putih dengan warna apapun hasilnya warna itu sendiri.

```
    finally
     abmpTexture.Free;
    end;
  end;
  hr:=FFontSurface.GetDC(dc);
```

Dapatkan handle device context surface.

```
  if succeeded(hr) then
  begin
   BitBlt(dc,0,0,abmp.Width,abmp.Height,
        aBmp.Canvas.Handle,0,0,SRCCopy);
```

Kopikan bitmap ke surface.

```
    FFontSurface.ReleaseDC(dc);
```

Bebaskan handle device context.

```
  end else raise EFontEngine2Error.Create('Error'+DDErrorString(hr));


  FSurfaceWidth:=aWidth;
  FSurfaceHeight:=aHeight;
```

Dimensi sprite kita simpan di FSurfaceWidth dan FSurfaceHeight

```
 finally
  abmp.Free;
  abmpFont.Free;
 end;
```

Bitmap-bitmap sementara kita bebaskan.

```
end;
```

3.6.10 Mengubah Properti Color.

Untuk menulis properti Color kita panggil metode SetColor. Setelah properti Color bernilai *Value*, maka agar perubahan ini terlihat, kita perlu memanggil rutin yang bertugas menciptakan ulang font yakni *RecreateFont*. Penjelasan tentang rutin RecreateFont akan dijelaskan berikutnya. Sebelum kita

memenggil RecreateFont, kita perlu mengisi nilai ChangeColor menjadi true untuk memberitahukan RecreateFont bahwa yang ingin kita ubah adalah warna font.

```
procedure TFontEngine2.SetColor(const Value: TColor);
begin
  FColor := Value;
  ChangeColor:=true;
  RecreateFont(nil);
  ChangeColor:=false;
end;
```

3.6.11 Mengubah Properti Font.

Untuk menulis properti Font kita menggunakan metode SetFont. Kita lakukan pengecekan apakah Value tidak sama dengan nil. Kemudian kita panggil RecreateFont untuk menciptakan ulang font.

```
procedure TFontEngine2.SetFont(const Value: TFont);
begin
  if value<>nil then
  begin
    RecreateFont(value);
  end;
end;


procedure TFontEngine2.SetTexture(const value: TBitmap);
var abmp:TBitmap;
begin
   abmp:=TBitmap.Create;
  try
    abmp.Width:=FFontStreamWidth;
    abmp.Height:=FFontStreamHeight;
    if (value.Width<>FFontStreamWidth) or
       (value.Height<>FFontStreamHeight) then
    begin
      if FTextureStyle=tsStretch then
        abmp.Canvas.StretchDraw(Rect(0,0,FFontStreamWidth,FFontStreamHeight),value);
    end else abmp.Assign(value);
    FFontTexture.Clear;
```

```
    abmp.SaveToStream(FFontTexture);
  finally
    abmp.Free;
  end;
end;

procedure TFontEngine2.SetTextureStyle(const Value: TTextureStyle);
begin
  FTextureStyle := Value;
end;

procedure TFontEngine2.SetUseTexture(const Value: boolean);
begin
  FUseTexture := Value;
  RecreateFont(nil);
end;
```

3.6.12 Finalisasi.

Finalisasi, seperti biasa dikerjakan oleh destruktor *Destroy*. Proses yang dilakukan adalah dengan membebaskan memori font, stream-stream, dan surface yang telah digunakan.

```
destructor TFontEngine2.Destroy;
begin
  FreeFont;
  FreeFontStream;
  FreeFontTexture;
  FreeFontSurface;
  nilprivateVar;
  FGraphicEngine:=nil;
  inherited;
end;
```

3.7 Kelas TAnimation.

3.7.1 Pendahuluan.

Kelas ini dibuat untuk menangani proses animasi. Kelas ini memiliki objek list internal yang mencatat frame-frame apa yang ada dan berapa lama frame ini akan ditampilkan dalam satuan mili detik. Cara kerjanya adalah dengan membandingkan waktu yang telah dilalui sejak frame mulai ditampilkan dengan delay frame tersebut. Jika waktu yang dilalui telah melebihi delay maka frame saat ini sudah saatnya diperbarui dengan frame berikutnya.

TAnimation ini hanya bertugas memberitahukan frame yang harus ditampilkan kepada kelas TSpriteEngine. Bagaimana frame-frame ini ditampilkan diserahkan kepada TSpriteEngine dan bukan merupakan taggung jawab TAnimation.

Kelas TAnimation juga tidak peduli apakah daftar frame yang disimpan dalam listnya merupakan frame-frame animasi yang berhubungan dengan suatu objek sprite engine. Pemrogram yang harus bertanggung jawab untuk hal ini.

3.7.2 Properti.

Kelas ini memiliki beberapa properti penting. Properti-properti tersebut adalah:

- *AnimationFrame* bertipe TanimationFrame. Properti ini berguna untuk mendapatkan nomer frame yang harus ditampilkan dalam proses animasi.
- *Count* bertipe integer mncatat data jumlah frame yang disimpan oleh objek instance kelas TAnimation.
- *DeleteOnFinish* bertipe boolean. Properti ini belum digunakan untuk engine versi ini. Properti ini penulis maksudkan untuk mencatat status apakah objek instance kelas TAnimation ini akan dihapus dari daftar animasi (TanimationList) jika frame-frame animasinya telah selesai dimainkan.
- *Playing* bertipe boolean. Properti ini mencatat status apakah frame-frame masih ada yang sedang dimainkan.
- *OnStarted* bertipe TAnimationEvent digunakan untuk memberitahukan saat animasi mulai dimainkan.
- *OnFinished* bertipe TanimationEvent digunakan untuk memberitahukan saat animasi selesai dimainkan.
- *OnFrameChanged* digunakan untuk memberitahukan jika sebuah frame telah di-update dengan frame baru.

3.7.3 Inisialisasi.

Proses inisialisasi seperti pada kelas lain ditangani oleh konstruktor Create. Proses yang dikerjakan di konstruktor Create adalah melakukan inisialisasi variabel-variabel internal kelas TAnimation.

constructor TAnimation.Create;

```
begin
  LastTick:=0;
  Ticks:=0;
  FrameIndex:=0;
  NoFrameList:=TList.Create;
  FOnFrameChanged:=nil;
  FOnFinished:=nil;
  FOnStarted:=nil;
  FDeleteOnFinish:=false;
  FSpriteFile:='';
  FPlaying:=false;
end;
```

3.7.4 Format File Data Animasi.

Seperti halnya kelas-kelas lain, kelas ini juga memiliki kemampuan membaca data frame-frame dari file. Nama file data sprite adalah file yang dibaca oleh kelas TSpriteEngine. No.Frame adalah nomer frame yang harus ditampilkan. No.Frame erat kaitannya dengan properti FrameNow milik kelas TSpriteEngine. Delay adalah lamanya suatu No.Frame ditampilkan. Posisi X,Y,Z adalah posisi xyz relatif terhadap posisi sprite pada frame sebelumnya. Misal pada frame sebelumnya posisi sprite berada pada 100,100 sedangkan posisi x frame saat ini adalah 10 dan posisi y=-5 maka posisi dimana sprite harus diletakkan adalah (100+10,100-10).

Berikut ini adalah format data animasi yang kita gunakan untuk kelas TAnimation.

<ID Pengenal='ANIM'>

<Nama file data sprite>

<Jumlah frame=N>

<No.Frame Ke-0>

<Delay Ke-0>

<Posisi X Ke-0>

<Posisi Y Ke-0>

<Posisi Z Ke-0>

<No.Frame Ke-1>

<Delay Ke-1>

<Posisi X Ke-1>

<Posisi Y Ke-1>

<Posisi Z Ke-1>

…..dst

<No.Frame Ke-(N-1)>

<Delay Ke-(N-1)>

<Posisi X Ke-(N-1)>

<Posisi Y Ke-(N-1)>

<Posisi Z Ke-(N-1)>

Contoh file data animasi:

ANIM

C:\My Documents\My Pictures\sprite.txt

3

3

10

0

0

0

4

10

0

0

0

5

10

0

0

0

3.7.5 Membaca File Data Animasi.

Untuk keperluan membaca file data animasi, kita akan melengkapi kelas TAnimation dengan metode LoadFromFile yang berguna untuk membaca data dari file. Prosesnya cukup sederhana.

```
procedure TAnimation.LoadFromFile(const filename: string);

var i,noFrame,delay,framectr:integer;
```

```
  aPos:TPosition;
  str:string;
  FT:TextFile;
```

Kita siapkan variabel sementara

```
begin
 if FileExists(Filename) then
 begin
  try
   AssignFile(FT,Filename);
   Reset(FT);
   Readln(FT,str);
   if str='ANIM' then
```

Jika file yang diinputkan lewat paramater ada, buka dan baca file tersebut dan test apakah berisi data file animasi milik kita.

```
    begin
     Readln(FT,str);
```

Jika ya, baca baris berikutnya yang berisi nama file data sprite.

```
     if FileExists(str) then
     begin
      FSpriteFile:=str;
     end else
     begin
      Raise EAnimationError.Create('Sprite file:'+FSpriteFile+' not found.');
      exit;
     end;
```

Lakukan tes apakah file data sprite tersebut ada. Jika tidak ada timbulkan eksepsi.

```
     Readln(FT,str);
     if StrToIntRaise(str,
       Filename+' contains invalid format.Unable to read number of frame.',
       framectr)=false then exit;
```

Baca baris berikutnya yang berisi data jumlah frame dalam file. Kita lakukan konversi data ini ke nilai integer. Jika proses konversi gagal maka keluar dari prosedur dan timbulkan eksepsi.

```
     Clear;
```

Jika berhasil kita bersihkan isi NoFrameList dengan memeanggil metode Clear.

Kita lakukan proses proses pembacaan looping sebanyak jumlah frame yang ada. Tiap kali data berhasil dibaca kita konversi ke nilai integernya. Urutan pembacaannya adalah No.frame, Delay, posisi koordinat X, Y dan Z. Pada akhir loop lita tambahkan data-data tersebut ke NoFrameList.

```
for i:=0 to frameCtr-1 do
  begin
   Readln(FT,str);
   if StrToIntRaise(str,
     Filename+' contains invalid format.Unable to read frame No.'+IntTostr(i),
     Noframe)=false then exit;
   Readln(FT,str);
   if StrToIntRaise(str,
     Filename+' contains invalid format.Unable to read delay No.'+IntTostr(i),
     Delay)=false then exit;
   Readln(FT,str);
   if StrToIntRaise(str,
     Filename+' contains invalid format.Unable to read pos x No.'+IntTostr(i),
     aPos.x)=false then exit;
   Readln(FT,str);
   if StrToIntRaise(str,
     Filename+' contains invalid format.Unable to read pos y No.'+IntTostr(i),
     aPos.y)=false then exit;
   Readln(FT,str);
   if StrToIntRaise(str,
     Filename+' contains invalid format.Unable to read pos z No.'+IntTostr(i),
     aPos.z)=false then exit;
   Add(NoFrame,aPos.X,apos.Y,aPos.Z,delay);
  end;
 end else
  Raise EAnimationError.Create(Filename+' contains invalid format.');
finally
 CloseFile(FT);
```

Tutup file yang telah dibuka jika telah selesai atau terjadi eksepsi.

```
 end;
end else
 Raise EAnimationError.Create(Filename+' not found.');
```

```
end;
```

3.7.6 Proses Mendapatkan Frame Yang Harus Ditampilkan.

Untuk mendapatkan frame yang harus ditampilkan kita harus membaca properti *AnimationFrame*. Data properti ini berasal dari nilai yang dikembalikan metode privat *GetAnimationFrame*.

```
function TAnimation.GetAnimationFrame: TAnimationFrame;
begin
  if (FrameIndex=0) and (Assigned(FOnStarted)) then
  begin
    FPlaying:=true;
    FOnStarted(self);
```

Jika FrameIndex=0, ini berarti proses animasi baru dimulai, oleh karena itu kita jalankan event *onStarted* jika rutin yang menangani event ini ada. Selain itu kita juga mengubah status properti *Playing* menjadi true.

```
  end;
  Result:=PAnimationFrame(NoFrameList.Items[FrameIndex])^;
```

Kita baca frame animasi saat ini dengan mengindeks data yang ada pada list berdasarkan *FrameIndex*.

```
  Ticks:=GetTickCount;
```

Kita ambil informasi tick count Windows. Tick count adalah waktu yang telah dilampaui sejak Windows pertama kali distart dalam satuan milidetik.

```
  if (Ticks-LastTick)>PAnimationFrame(NoFrameList.Items[FrameIndex]).Delay then
  begin
```

Jika waktu saat ini dikurangi waktu saat frame diperbarui sebelumnya melebihi delay maka sudah saatnya kita harus memperbarui frame yang akan ditampilkan berikutnya.

```
    LastTick:=Ticks;
    Inc(FrameIndex);
    if Assigned(FOnFrameChanged) then
      FOnFrameChanged(self);
```

Waktu saat proses update ini kita catat di LastTick. FrameIndex kita tambahkan satu untuk menampilkan frame berikutnya. Kita juga memanggil rutin yang akan menangani event *OnFrameChanged* jika rutin tersebut ada.

```
    if FrameIndex=NoFrameList.Count then
    begin
```

Jika FrameIndex sama dengan jumlah frame yang ada dalam list, hal ini berarti proses animasi telah mencapai akhir, oleh karena itu kita kembalikan FrameIndex ke nilai nol untuk mencegah terjadinya eksepsi karena kita mencoba mengakses frame yang berada diluar jangkauan list. Kita juga perlu mengubah status properti Playing menjadi false guna memberitahukan bahwa proses memainkan animasi telah selesai serta menjalankan event handler yang menangani kejadian *OnFinished*.

```
      FrameIndex:=0;
      FPlaying:=false;
      if Assigned(FOnFinished) then
        FOnFinished(self);
    end;
  end;
end;
```

3.7.7 Proses Mendapatkan Frame Berdasarkan Indeks.

Untuk keperluan ini kita harus membaca properti *AnimationFrames*. Properti ini berupa array bertipe TAnimationFrame. Properti ini berbeda dengan properti AnimationFrame pada proses mendapatkan data frame animasi. Pada AnimationFrame dilakukan proses pengecekan apakah sudah saatnya frame harus diupdate. Pada properti AnimationFrames data frame animasi yang dikembalikan adalah data yang ditunjuk oleh indeks. Nilai yang diberikan properti ini adalah nilai hasil pemanggilan fungsi *GetAnimationFrames*.

```
function TAnimation.GetAnimationFrames(index: integer): TAnimationFrame;
var anim:TAnimationFrame;
begin
  anim.Frame:=0;
  anim.Position:=setPosition(0,0,0);
  anim.Delay:=0;
```

Inisialisasi variabel sementara *anim* dengan nol.

```
  if (index>=0) and (index<NoFrameList.Count) then
  begin
    anim:=PAnimationFrame(NoFrameList.Items[index])^;
```

Jika index masih dalam batas yang diijinkan yaitu antara dari 0 hingga NoFrameList.Count-1 maka data frame animasi yang ditunjuk oleh indeks kita simpan di variabel anim.

```
  end;
  Result:=anim;
```

Kita berikan data yang ada pada anim sebagai nilai yang dikembalikan fungsi.

end;

3.7.8 Menambahkan Frame Animasi ke List.

Proses menambahkan frame animasi ke NoFrameList dikerjakan oleh metode *Add*. Metode ini menerima parameter input terdiri atas *NoFrame*, *Delay*, dan posisi *X*, *Y*, *Z* relatif terhadap frame sebelumnya.

```
procedure TAnimation.Add(const NoFrame, X, Y,Z, Delay: integer);
var item:PAnimationFrame;
```

Kita siapkan variabel sementara bertipe *PAnimationFrame*. PAnimationFrame adalah pointer yang menunjuk ke tipe TAnimationFrame.

```
begin
  new(item);
  item.Frame:=NoFrame;
  item.Position:=SetPosition(x,y,z);
  item.Delay:=Delay;
```

Kita lakukan pemesanan memori untuk frame animasi, dan kita isikan field-field variabel sementara item dengan paramater-parameter input.

```
  NoFrameList.Add(item);
```

Kita tambahkan item ke list.

```
end;
```

3.7.9 Menghapus Sebuah Frame Animasi.

Untuk menghapus sebuah frame animasi kita menggunakan metode *Delete*. Metode ini akan menghapus sebuah data frame animasi yang ditunjuk oleh sebuah indeks. Selain menghapus entry dalam list, metode ini juga membebaskan memori yang telah digunakan untuk menyimpan data frame animasi.

```
procedure TAnimation.Delete(const i: integer);
var item:PAnimationFrame;
begin
  item:=PAnimationFrame(NoFrameList.items[i]);
  dispose(item);
  NoFrameList.Delete(i);
end;
```

3.7.10 Menghapus Seluruh Frame Animasi.

Metode *Clear* adalah metode yang fungsinya untuk menghapus semua frame animasi yanga ada dala list. Metode ini juga membebaskan semua memori yang digunakan untuk menyimpan data frame.

```
procedure TAnimation.Clear;
var i:integer;
   item:PAnimationFrame;
```

Kita siapkan dua variabel sementara.

```
begin
 for i:=NoFrameList.Count-1 downto 0 do
 begin
```

Lakukan proses looping sebanyak jumlah data frame dalam list. Proses looping ini kita mulai dari dari indeks terbesar sampai 0 untuk mencegah kita mengakses data frame animasi yang telah dihapus.

```
   item:=PAnimationFrame(NoFrameList.Items[i]);
   dispose(item);
   item:=nil;
```

Kita bebaskan memori yang dipakai oleh data frame yang ditunjuk oleh item.

```
   NoFrameList.Delete(i);
```

Kemudian kita hapus entry indeks ini dari list.

```
 end;
end;
```

3.7.11 Finalisasi.

Saat proses finalisasi dikerjakan, kita buang isi daftar nomer frame dan kita bebaskan memori yang dipakai NoFrameList.

```
destructor TAnimation.Destroy;
begin
 Clear;
 NoFrameList.Free;
 NoFrameList:=nil;
 inherited;
end;
```

3.8 Kelas TAnimationList.

3.8.1 Pendahuluan.

Kelas ini diciptakan untuk menampung sejumlah kelas TAnimation yang akan ditampilkan pada saat program dijalankan. Kelas ini merupakan turunan kelas TList, kita akan menambahkan beberapa properti dan metode baru pada kelas TAnimationList yang tidak ada pada kelas TList.

3.8.2 Properti.

Karena kelas ini dirancang untuk menampung daftar animasi yang harus ditampilkan, maka kelas ini perlu kita lengkapi dengan properti berikut:

- *Items* bertipe array TAnimation. Untuk mengubah atau membaca isi daftar animasi.

3.8.3 Inisialisasi.

Kelas ini tidak memiliki konstruktor. Konstruktor yang digunakan untuk melakukan inisialisasi adalah konstruktor Create milik kelas pendahulunya yaitu TList.

3.8.4 Menambahkan Animasi.

Untuk menambahkan animasi digunakan fungsi *Add*. Fungsi ini menambahkan animasi kedalam daftar. Metode ini memanggil fungsi Add milik kelas TList, perbedaannya adalah pada parameternya. Fungsi Add milik kelas TAnimationList memiliki parameter bertipe TAnimation, sedangkan Add milik Tlist parameternya bertipe pointer.

```
function TAnimationList.Add(Item: TAnimation): integer;
begin
  result:=Inherited Add(Pointer(item));
end;
```

3.8.5 Menghapus Animasi.

Proses menghapus animasi dari daftar dilakukan oleh prosedur *Delete*. Sama seperti metode Add, Delete juga memanggil prosedur Delete milik pendahulunya. Prosedur Delete milik kelas TAnimationList, parameternya bertipe TAnimation, sedangkan Delete milik TList parameternya bertipe integer.

Agar Delete milik kelas TAnimationList bisa menggunakan Delete milik Tlist, maka terlebih dahulu kita harus mendapatkan indeks dimana *item* berada. Untuk keperluan tersebut kita memanggil fungsi *isAnimationInList*. Fungsi ini bertugas untuk melakukan pengecekan apakah item telah ada dalam daftar. Jika ada maka *indx* akan berisi indeks dimana item berada, selanjutnya kita panggil metode Delete milik TList untuk menghapus animasi. Delete hanya menghapus entri dalam daftar animasi namun tidak membebaskan memori yang digunakan objek item

```
procedure TAnimationList.Delete(item: TAnimation);
```

```
var indx:integer;
begin
  if isAnimationInList(item,indx) then
  begin
    inherited Delete(indx);
  end;
end;
```

3.8.6 Mencari Indeks Animasi.

Kadang kala kita memerlukan informasi apakah sebuah objek animasi telah ada dalam daftar animasi. Situasi seperti ini kita hadapi ketika melakukan proses penghapusan entri dari daftar animasi. Untuk itu kita perlu melengkapi kelas ini dengan metode untuk mencari indeks animasi.

Metode ini menggunakan fungsi privat GetItems yang bertugas untuk mendapatkan objek animasi berdasarkan indeks. Proses pencariannya sederhana yaitu dengan melakukan looping sebanyak jumlah animasi dan membandingkan objek animasi yang diperoleh dari GetItems dengan objek animasi yang diinputkan dari parameter. Jika sama kita kita isi *animindex* dengan posisi indeks serta mengembalikan nilai fungsi sama dengan true, selanjutnya kita hentikan pencarian dan keluar dari rutin.

```
function TAnimationList.isAnimationInList(anim: TAnimation;var animIndex:integer): boolean;
var i:integer;
    item:TAnimation;
begin
  result:=false;
  for i:=0 to Count-1 do
  begin
    item:=GetItems(i);
    if item=anim then
    begin
      animIndex:=i;
      result:=true;
      exit;
    end;
  end;
end;
```

3.8.7 Mendapatkan Animasi Berdasarkan Indeks.

Untuk mendapatkan objek animasi yang ditunjuk suatu indeks digunakan fungsi GetItems. Fungsi ini mengembalikan nilai properti *Items* milik TList yang ditunjuk oleh *index* dan melakukan typecast pointer ke TAnimation. Fungsi ini digunakan pada saat membaca properti Items milik kelas TAnimationList

```
function TAnimationList.GetItems(Index: integer): TAnimation;
begin
  Result:=TAnimation(Inherited Items[index]);
end;
```

3.8.8 Mengubah Animasi Berdasarkan Indeks.

Proses ini adalah kebalikan dari proses diatas yaitu untuk mengubah objek animasi yang ditunjuk oleh indeks dengan objek animasi yang diinputkan dari parameter.

```
procedure TAnimationList.SetItems(Index: integer; const Value: TAnimation);
begin
  Inherited Items[index]:=Pointer(Value);
end;
```

3.8.9 Mengganti Animasi.

Jika kita ingin mengganti suatu objek animasi dengan objek animasi lain atau mengganti suatu animasi yang ada pada suatu daftar animasi dengan animasi lain kita menggunakan metode *Replace*. Metode Replace ini adalah metode yang di overload.

Ada dua Replace yang pertama adalah seperti berikut. Parameternya adalah objek animasi lama yang akan diganti dan objek animasi baru yang akan mengganti. Prosesnya sederhana yaitu dengan mencari indeks objek animasi lama. Jika objek animasi lama ada dalam daftar maka kita sisipkan objek animasi baru didepan objek animasi lama, selanjutnya objek animasi lama kita hapus. Kita kembalikan nilai fungsi sama dengan true yang menyatakan bahwa kita telah sukses mengganti objek animasi lama dengan objek animasi baru.

```
function TAnimationList.Replace(oldItem, newItem: TAnimation): boolean;
var i:integer;
begin
  result:=false;
  if isAnimationInList(olditem,i) then
  begin
    inherited insert(i,Pointer(newitem));
    inherited Delete(i+1);
```

```
   result:=true;
  end;
end;
```

Replace yang kedua digunakan untuk mengganti objek animasi yang sama dengan objek animasi yang disimpan dalam *AnimList*. Jika ada yang sama maka objek animasi yang lama kita ganti dengan objek animasi yang baru.

```
function TAnimationList.Replace(animList: TAnimationList;
  newItem: TAnimation): boolean;
var i,j:integer;
    item:TAnimation;
begin
  result:=false;
  for i:=0 to Count-1 do
  begin
    item:=GetItems(i);
    for j:=0 to animList.Count-1 do
    begin
      if item=animlist.Items[j] then
      begin
        inherited Insert(i,newitem);
        inherited Delete(i+1);
        result:=true;
        exit;
      end;
    end;
  end;
end;
```

3.8.10 Finalisasi.

Seperti halnya proses inisialisasi, proses finalisasi dikerjakan oleh destruktor Destroy milik pendahulunya.

3.9 Kelas TPrimitive.

3.9.1 Pendahuluan.

Kelas ini sebenarnya jarang kita pergunakan, penulis menyusun kelas ini hanya sekedar untuk menjelaskan proses surface locking yang telah kita bahas sebelumnya. Kelas ini merupakan kelas dasar bagi seperti objek pixel, garis, rectangle, lingkaran, elips dan lain-lain. TPrimitive memiliki sebuah metode abstrak yaitu *Draw* yang digunakan untuk melakukan proses rendering, oleh karena itu jangan menciptakan instance kelas ini secara langsung, untuk menggunakan kelas ini ciptakan instance dari kelas turunannya yang telah mengimplementasikan metode Draw, contohnya kelas TPixel yang nanti akan kita bahas.

3.9.2 Properti.

Properti yang dimiliki oleh kelas ini adalah sebagai berikut:

- Position bertipe TPosition, digunakan untuk menyimpan posisi dimana objek harus digambar.
- Color bertipe integer, menyimpan data warna yang digunakan.

3.9.3 Inisialisasi.

Inisialisasi dekerjakan oleh konstruktor Create. Parameternya adalah objek bertipe TGraphicEngine.

```
constructor TPrimitive.Create(AGraphicEngine: TGraphicEngine);
begin
  ParentGraphicEngine:=AGraphicEngine;
  FColor:=$ffffffff;
  FPosition:=uDirectDraw.SetPosition(0,0,0);
  ZeroMemory(@SurfDesc,SizeOf(TDDSurfaceDesc));
  SurfDesc.dwSize:=SizeOf(TDDSurfaceDesc);

  ZeroMemory(@PixelFormat,SizeOf(TDDPixelFormat));
```

Inisialisasi variabel-variabel internal milik kelas TPrimitive

```
  if(ParentGraphicEngine<>nil) and
    (ParentGraphicEngine.BackSurface<>nil)then
  begin
    PixelFormat.dwSize:=Sizeof(TDDPixelFormat);
    ParentGraphicEngine.BackSurface.GetPixelFormat(PixelFormat);
  end;
end;
```

3.9.4 Persiapan Sebelum Rendering.

Karena proses rendering yang dikerjakan oleh kelas TPrimitive dan kelas-kelas turunannya adalah menulis surface secara langsung, maka sebelum proses rendering sesungguhnya dikerjakan, kita perlu melakukan proses persiapan guna mendapatkan pointer ke surface yang akan kita tulis yaitu dengan mengunci surface yang akan kita gunakan.

```
procedure TPrimitive.BeginDraw;
var hr:HResult;
begin
  if (ParentGraphicEngine<>nil) and
     (ParentGraphicEngine.BackSurface<>nil) then
  begin
```

Jika ParentGraphicEngine dan BackSurface menunjuk pada pointer yang valid maka kita kerjakan proses surface locking.

```
    hr:=ParentGraphicEngine.BackSurface.Lock(nil,SurfDesc,DDLOCK_SURFACEMEMORYPTR or
DDLOCK_WAIT,0);
```

Kita kunci seluruh back surface dengan mengisi pointer rectangle sama dengan nil dan memerintahkan DirectDraw agar terus mencoba mengunci back surface sampai back surface berhasil dikunci. Jika proses berhasil maka field *lpSurface* milik SurfDesc akan menyimpan alamat pointer ke surface back surface. Jika gagal kita timbulkan eksepsi EPrimitiveError

```
    if Failed(hr) then
    begin
      raise EPrimitiveError.Create(DDErrorString(hr));
      exit;
    end;
  end;
end;
```

3.9.5 Proses Rendering.

Proses rendering dikerjakan oleh metode Draw. Seperti yang telah disebutkan di atas, metode ini adalah metode abstrak, oleh karena itu implementasi metode Draw ini diserahkan kepada kelas-kelas turunan TPrimitive.

3.9.6 Akhir Proses Rendering.

Untuk mengakhiri proses rendering kita perlu membebaskan surface yang telah kita kunci agar bisa diakses oleh proses lain. Jika kita lupa membebaskan penguncian maka surface kita tidak dapat digambar ke front buffer oleh DirectDraw.

```
procedure TPrimitive.EndDraw;
var hr:HResult;
begin
  if (ParentGraphicEngine<>nil) and
     (ParentGraphicEngine.BackSurface<>nil) then
  begin
    hr:=ParentGraphicEngine.BackSurface.Unlock(nil);
```

Kita bebaskan back surface dengan memanggil fungsi Unlock. Parameternya adalah nilai nil karena kita mengunci seluruh back surface. Jika gagal kita timbulkan eksepsi.

```
    if Failed(hr) then
    begin
      raise EPrimitiveError.Create(DDErrorString(hr));
      exit;
    end;
  end;
end;
```

3.9.7 Finalisasi.

Finalisasi dikerjakan oleh destruktor *Destroy*. Untuk saat ini Destroy tidak melakukan apa-apa selain memanggil Destroy milik kelas TObject. Finalisasi yang aman dikerjakan dengan Free.

3.10 Kelas TPixel.

3.10.1 Pendahuluan.

Kelas ini adalah turunan TPrimitive. TPixel hanya mengimplementasikan metode Draw milik TPrimitive, yang lainnya sama persis dengan TPrimitive. TPixel digunakan untuk menggambar sebuah pixel ke back surface.

3.10.2 Proses Rendering.

Berikut ini adalah metode Draw yang bertanggung jawab melakukan proses rendering. Sebelum memanggil Draw kita harus memanggil BeginDraw dan memanggil EndDraw sesudahnya.

```
procedure TPixel.Draw;
var offsets,pixelwidth:integer;
begin
```

```
inherited;
case PixelFormat.dwFlags of
  DDPF_RGB:begin
          case PixelFormat.dwRGBBitCount of
           15,16:pixelwidth:=2;
           24:pixelwidth:=3;
           32:pixelwidth:=4;
          end;
        end;
  DDPF_PALETTEINDEXED8:begin
            pixelwidth:=1;
                end;
end;
```

Kita tentukan ukuran data warna yang harus kita tulis back surface berdasarkan data yang disimpan dalam variabel PixelFormat.

```
offsets:=Integer(SurfDesc.lpSurface)+FPosition.Y*SurfDesc.lPitch+FPosition.X*pixelwidth;
```

Kita hitung alamat memori di mana data warna harus kita letakkan.

```
move(FColor,Pointer(offsets)^,pixelwidth);
```

Isi warna ke alamat memori yang ditunjuk oleh offset sebanyak pixelwidth.

```
end;
```

3.11 Kelas TFillRect.

3.11.1 Pendahuluan.

Kelas ini digunakan untuk mengambar rectangle yang diisi warna. Penulis sengaja tidak menurunkan dari kelas TPrimitive karena penulis ingin menjelaskan cara menggambar pixel dengan menggunakan fungsi Blt dengan tambahan efek color fill. Kelas ini nanti akan kita gunakan untuk menampilkan menu sorot pada game yang akan kita buat pada akhir tutorial ini.

3.11.2 Properti.

Properti kelas ini ada dua yaitu:

- *Color* bertipe integer yang menyimpan data warna.
- *Rect* bertipe TRect yang digunakan untuk menentukan ukuran rectangle.
- *Surface* dimana rectangle akan digambar bertipe IDirectDrawSurface

3.11.3 Inisialisasi.

Inisialisasi dilakukan oleh konstruktor Create. Prosesnya adalah dengan melakukan inisialisasi variabel-variabel internal TFillRect

```
constructor TFillRect.Create;
begin
  ZeroMemory(@bltfx,SizeOf(TDDBltFX));
  FColor:=0;
  ZeroMemory(@FRect,SizeOf(TRect));
  FSurface:=nil;
end;
```

3.11.4 Proses Rendering.

Proses rendering dikerjakan metode Draw. Kita siapkan variabel bltFX dengan mengisi field dwSize dan dwFillColor. Kedua field in yang diperlukan untuk melakukan blit dengan efek color fill. Kemudian kita panggil fungsi Blt milik Fsurface dengan flag tambahan DDBLT_COLORFILL guna melakukan proses blit color fill.

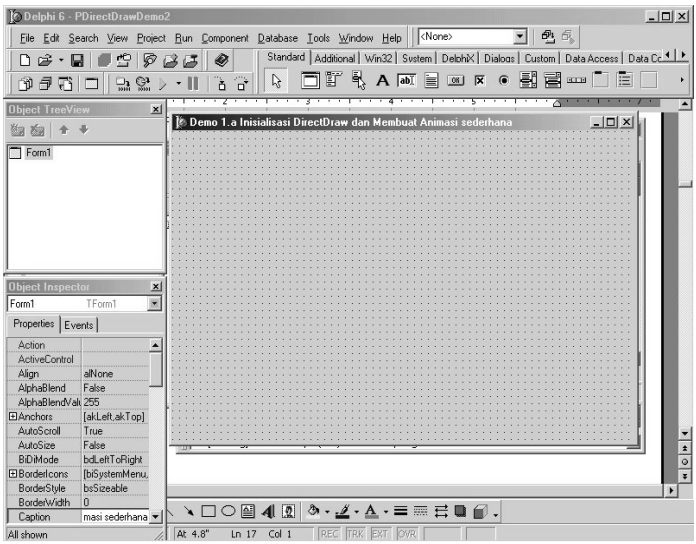
```
procedure TFillRect.Draw;
begin
 if FSurface<>nil then
 begin
  ZeroMemory(@bltfx,SizeOf(TDDBltFX));
  bltFX.dwSize:=SizeOf(TDDBltFX);
  bltfx.dwFillColor:=FColor;
  FSurface.Blt(@FRect,nil,nil,DDBLT_WAIT or DDBLT_COLORFILL,@bltfx);
 end;
end;
```

3.11.5 Finalisasi.

Finalisasi dilakukan oleh destruktor Destroy. Pada dasarnya Destroy tidak melakukan apa-apa selain memanggil destruktor pendahulunya.

Bab 4
Demo Program 1

4.1. Listing Program 1.a Inisialisasi DirectDraw Mode Full Screen dan Loading File Bitmap ke Surface.



Gambar 4.1 Tampilan IDE Demo Program 1.a

File project demo program 1.a terdapat pada direktori Pemrograman Game Dengan DirectX\Bab 4\Demo 2\PDirectDrawDemo2.dpr. Program dalam format executablenya Pemrograman Game Dengan DirectX\DirectDraw\Demo 2\PDirectDrawDemo2.exe. Berikut ini adalah listing Unit1.pas

```
{----------------------------------------}
{ Demo I Inisialisasi DirectDraw }
{----------------------------------------}
{File:uDirectDrawDemo2.pas    }
{Code:Zamrony P Juhara        }
{ Tekan ESC untuk keluar      }
{----------------------------------------}
```

```
{      (C) 2002 Zamrony P Juhara }
{-----------------------------------------}
unit uDirectDrawDemo2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,uDirectDraw;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
  private
    MyGraphicEngine:TGraphicEngine;
    MyBackgroundEngine:TBackgroundEngine;
    MySpriteEngine:TSpriteEngine;
    LastTicks,DelayTicks,FrameCtr:integer;
    { Private declarations }
  public
    procedure AppOnIdle(sender:TObject;var done:boolean);
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.dfm}

procedure TForm1.AppOnIdle(sender: TObject; var done: boolean);
var ticks:integer;
begin
  done:=false;
  Ticks:=GetTickCount;
  if Ticks-LastTicks>DelayTicks then
  begin
    lastTicks:=Ticks;
    MyBackgroundEngine.Show;
    MySpriteEngine.FrameNow:=FrameCtr;
    Inc(FrameCtr);
    if FrameCtr=MySpriteEngine.Count then
      FrameCtr:=0;
    MySpriteEngine.Show;
    MyGraphicEngine.Show;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
var param:TGraphicEngineParam;
begin
  DelayTicks:=10;
  LastTicks:=0;
  FrameCtr:=0;
  param.Handle:=Handle;
  param.Width:=640;
  param.Height:=480;
  param.BitPerPixel:=16;
  param.BackBufferCount:=1;
  param.FullScreen:=true;
  param.AllowReboot:=true;
  MyGraphicEngine:=TGraphicEngine.Create(param);
  MyBackgroundEngine:=TBackgroundEngine.Create(MyGraphicEngine);
```

```
  MyBackgroundEngine.LoadFromFile('back.bmp');
  MySpriteEngine:=TSpriteEngine.Create(MyGraphicEngine);
  MySpriteEngine.LoadFromFile('spr.txt');
  Application.OnIdle:=AppOnIdle;
  ShowCursor(false);
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  MySpriteEngine.Free;
  MyBackgroundEngine.Free;
  MyGraphicEngine.Free;
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
    VK_ESCAPE,
    VK_RETURN,
    VK_SPACE:begin
               ShowCursor(true);
               Close;
             end;
  end;
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
var poss:TPosition;
begin
  poss.X:=X;
  poss.Y:=Y;
  MySpriteEngine.Position:=poss;
end;
```

```
end.
```

4.2 Penjelasan Program 1.a.

4.2.1 Deklarasi.

Pada bagian deklarasi kita siapkan objek *MyGraphicEngine*, *MyBackgroundEngine* dan *MySpriteEngine*. Masing-masing bertugas menangani grafik, background dan animasi sprite. Kita juga menyiapkan beberapa variabel yang akan kita perlukan untuk melakukan animasi, yaitu:*LastTicks*, *DelayTicks* dan *FrameCounter*. LastTicks akan digunakan untuk menyimpan waktu saat ini, DelayTicks menyimpan delay animasi dan FrameCounter untuk pencacah frame animasi.

Kita juga mendeklarasikan metode privat guna menangani event OnIdle milik TApplication, yaitu *AppOnIdle*. OnIdle timbul saat aplikasi tidak lagi memproses pesan (message), event ini yang akan kita gunakan untuk memproses animasi kita.

4.2.2 Inisialisasi.

```
procedure TForm1.FormCreate(Sender: TObject);
var param:TGraphicEngineParam;
begin
  DelayTicks:=10;
  LastTicks:=0;
  FrameCtr:=0;
  param.Handle:=Handle;
  param.Width:=640;
  param.Height:=480;
  param.BitPerPixel:=16;
  param.BackBufferCount:=1;
  param.FullScreen:=true;
  param.AllowReboot:=true;
  MyGraphicEngine:=TGraphicEngine.Create(param);
  MyBackgroundEngine:=TBackgroundEngine.Create(MyGraphicEngine);
  MyBackgroundEngine.LoadFromFile('back.bmp');
  MySpriteEngine:=TSpriteEngine.Create(MyGraphicEngine);
  MySpriteEngine.LoadFromFile('spr.txt');
  Application.OnIdle:=AppOnIdle;
```

```
  ShowCursor(false);
end;
```

Saat memasuki prosedur FormCreate kita inisialisasi DelayTicks dengan nilai 10. DelayTicks adalah variabel yang akan kita gunakan mengurangi kecepatan animasi agar tidak terlalu cepat, sehingga tampilan animasi akan terlihat bagus. DelayTicks kita isi dengan 10 mili detik. Variabel LastTick akan kita gunakan untuk menyimpan informasi waktu terakhir, sedangkan FrameCtr adalah pencacah nomor frame yang akan ditampilkan.

Sebelum menciptakan objek MyGraphicEngine, MyBackground dan MySpriteEngine, terlebih dahulu kita siapkan variabel sementara Param bertipe TGraphicEngineParam yang akan kita isi dengan deskripsi paramater graphic engine yang kita inginkan. Aplikasi demo 1.a akan menggunakan resolusi 640x480 16 bit mode full screen dengan jumlah back buffer satu serta tombol kombinasi CTRL+ALT+DEL untuk reboot tetap berfungsi. Field Handle variable Param kita isi dengan handle form utama kita.

Setelah deskripsi graphic engine berisi data yang valid maka kita ciptakan MyGraphicEngine dengan Param sebagai parameter input. MyGraphicEngine harus diciptakan terlebih dahulu sebelum menciptakan objek-objek lain.

Selanjutnya kita ciptakan objek MyBackgroundEngine yang berfungsi menangani background scrolling dengan parameter input konstruktor Create kita isi MyGraphicEngine. Sebenarnya fungsi scrolling yang merupakan fungsi utama kelas TBackgroundEngine belum dipakai di contoh program ini. Penulis menciptakan MyBackgroundEngine sekedar untuk gambar latar. Setelah MyBackgroundEngine berhasil di inisialisasi, MyBackgroundEngine kita isi dengan gambar bitmap dari file Back.bmp. Sampai pada tahap ini maka MyBackground telah siap digunakan.

Objek MySpriteEngine inisialisasinya mirip dengan MyBackgroundEngine karena parameter input konstruktor Create milik MySpriteEngine sama dengan MyBackgroundEngine. Objek MySpriteEngine tidak harus diinisialisasi sesudah MyBackgroundEngine bisa sebaliknya, tapi baik MySpriteEngine maupun MyBackgroundEngine harus diciptakan sesudah MyGraphicEngine berhasil diinisialisasi. Selanjutnya MySpriteEngine kita isi dengan data sprite yang dibaca dari file Spr.txt. File ini berisi data sprite donat berwarna merah.

Properti OnIdle objek Application kita isi dengan rutin tempat kita melakukan proses rendering animasi yaitu AppOnIdle. Event OnIdle akan terjadi saat antrian pesan sudah habis sehingga Windows tidak lagi melakukan apa-apa.

Pada baris terakhir kita sembunyikan kursor mouse karena kita akan menggunakan sprite kita sendiri sebagai kursor.

4.2.3 Event OnKeyDown Form.

Event OnKeyDown milik form utama akan kita isi dengan rutin untuk menutup aplikasi. Jika tombol keyboard yang ditekan adalah tombol Escape atau

Enter atau Spasi maka kursor mouse yang sebelumnya disembunyikan ditampilkan lagi, kemudian metode Close milik form utama dipanggil untuk menutup aplikasi.

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
    VK_ESCAPE,
    VK_RETURN,
    VK_SPACE:begin
               ShowCursor(true);
               Close;
             end;
  end;
end;
```

4.2.4 Event OnMouseMove Form.

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
var poss:TPosition;
begin
  poss.X:=X;
  poss.Y:=Y;
  MySpriteEngine.Position:=poss;
end;
```

Karena kita akan menggunakan sprite donat sebagai kursor mouse maka kita ubah posisi sprite setiap kali ada gerakan penunjuk mouse, dengan cara mengisi properti Position milik MySpriteEngine dengan posisi penunjuk mouse saat ini. Untuk mengubah properti Position maka kita siapkan variabel sementara Poss bertipe TPosition. Kemudian Poss kita isi dengan posisi X dan Y mouse pointer, setelah itu kita kopikan isi Poss ke properti Position objek MySpriteEngine. Anda tidak bisa langsung mengisi field X dan Y properti Position objek MySpriteEngine, contoh: pernyataan MySpriteEngine.Position.X:=X akan menimbulkan kesalahan saat proses kompilasi.

Cara lain mengisi properti Position milik objek MySpriteEngine adalah dengan fungsi SetPosition.

4.2.5 Proses Rendering Animasi.

Proses rendering akan dikerjakan di prosedur AppOnIdle.

```
procedure TForm1.AppOnIdle(sender: TObject; var done: boolean);
var ticks:integer;
begin
  done:=false;
  Ticks:=GetTickCount;
  if Ticks-LastTicks>DelayTicks then
  begin
    lastTicks:=Ticks;
    MyBackgroundEngine.Show;
    MySpriteEngine.FrameNow:=FrameCtr;
    Inc(FrameCtr);
    if FrameCtr=MySpriteEngine.Count then
      FrameCtr:=0;
    MySpriteEngine.Show;
    MyGraphicEngine.Show;
  end;
end;
```

Saat memasuki prosedur AppOnIdle, kita isi Done dengan false dengan tujuan agar event OnIdle terjadi lagi, nilai defaultnya adalah true. Variabel Ticks kemudian kita isi dengan nilai yang dikembalikan oleh fungsi GetTickCount. Fungsi GetTickCount akan mengembalikan informasi waktu yang telah ditempuh Windows sejak pertama kali distart dalam satuan mili detik.

Ticks kemudian dikurangi dengan LastTicks untuk mendapatkan waktu sejak proses rendering sebelumnya. Nilai ini kemudian dibandingkan dengan DelayTicks, jika waktu ini telah melebihi DelayTicks maka sudah waktunya memperbarui tampilan layar.

Simpan nilai Ticks ke LastTicks untuk proses rendering berikutnya. Lakukan penggambaran backround ke back buffer dengan memanggil metode Show milik objek MyBackgroundEngine. Proses ini akan menimpa (overwrite) seluruh tampilan layar sebelumnya.

Isikan properti FrameNow objek MySpriteEngine dengan nomor frame yang akan ditampilkan. Tambah FrameCtr dengan satu untuk proses rendering berikutnya, jika lebih besar dari jumlah total frame yang ada di MySpriteEngine maka kembalikan FrameCtr ke frame awal. Lakukan proses penggambaran sprite ke back buffer dengan memanggil metode Show milik MySpriteEngine.

Karena seluruh proses rendering back buffer telah selesai maka tampilkan back buffer ke layar agar perubahannya terlihat dengan metode Show objek MyGraphicEngine.

4.2.6 Finalisasi.

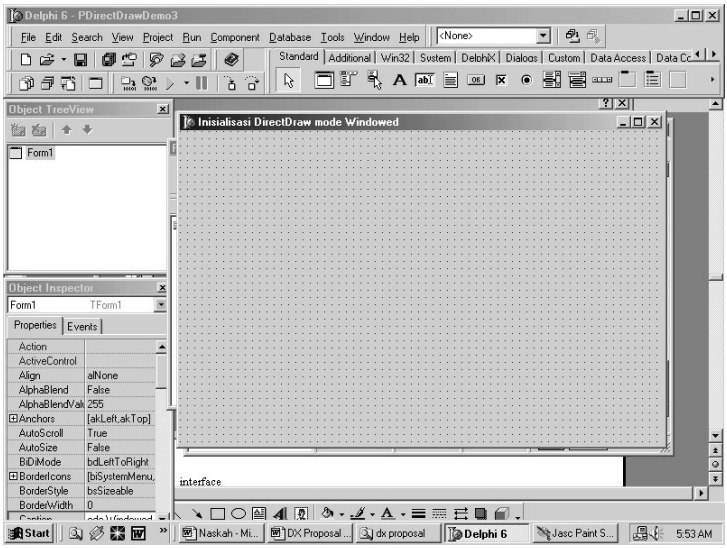
```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  MySpriteEngine.Free;
  MyBackgroundEngine.Free;
  MyGraphicEngine.Free;
end;
```

Pada proses finalisasi kita bebaskan memori objek-objek yang kita pakai dengan memanggil metode Free masing-masing objek.

Dibawah ini adalah screenshot tampilan program 1.a ketika dijalankan. Program 1.a telah menggunakan proses sprite clipping, sehingga jika gambar donat melebihi layar maka gambar otomatis akan dipotong oleh DirectDraw.

Gambar 4.2 Screenshot Demo Program 1.a

4.3 Listing Program 1.b Inisialisasi DirectDraw Mode Windowed.

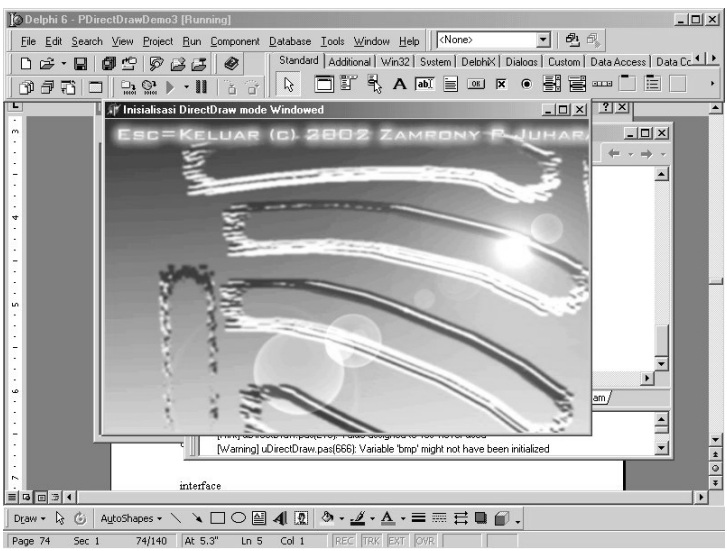


Gambar 4.3 Tampilan IDE Demo Program 1.b

```
{-------------------------------------}
{ Inisialisasi DirectDraw Windowed  }
{  (c) 2002 Zamrony P Juhara       }
{-------------------------------------}
{File:uDirectDrawDemo3.Pas         }
{Code:Zamrony P Juhara             }
{Tgl: Oktober 2002                 }
{-------------------------------------}
unit uDirectDrawDemo3;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,uDirectDraw;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
```

```
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    GraphicEngine:TGraphicEngine;
    Background:TBackgroundEngine;
    Procedure AppOnIdle(Sender:TObject;var done:boolean);
    { Private declarations }
  public

    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AppOnIdle(Sender: TObject; var done: boolean);
begin
  done:=false;
  Background.Show;
  GraphicEngine.Show;
end;

procedure TForm1.FormCreate(Sender: TObject);
var param:TGraphicEngineParam;
begin
  fillchar(param,sizeOf(TGraphicEngineParam),0);
  param.Handle:=Handle;
  param.Width:=Width;
  param.Height:=Height;
  param.FullScreen:=false;
  GraphicEngine:=TGraphicEngine.Create(param);
```

```
  Background:=TBackgroundEngine.Create(GraphicEngine);
  Background.LoadFromFile('back.bmp');
  Application.OnIdle:=AppOnIdle;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  Background.Free;
  GraphicEngine.Free;
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
   VK_ESCAPE:close;
  end;
end;

end.
```



Gambar 4.4 Screenshot Demo Program 1.b

Penulis sengaja tidak menjelaskan isi kode programnya karena isinya hampir sama dengan program 1.a. Perbedaannya terletak pada saat menciptakan GraphicEngine, dimana field *FullScreen* milik *Param* kita isi dengan nilai false.

Untuk contoh-contoh lain pembaca dapat melihatnya di dalam CD-ROM yang disertakan bersama tutorial ini. Programnya hampir sama dengan contoh program diatas, dan menurut cukup mudah sehingga pembaca diharapkan dapat memahami sendiri maksud tiap baris kode dalam program-program tersebut.

Bab 5
Menggunakan DirectSound.

5.1 Pendahuluan.

Untuk memainkan file suara khususnya file WAV, Windows menyediakan berbagai alternatif, yakni dengan Multimedia Control Interface (MCI), Wavemix.dll, fungsi PlaySound atau dengan DirectSound.

Cara termudah adalah dengan fungsi PlaySound. Deklarasi fungsi ini terdapat di file MMSystem.pas dan dideklarasikan sebagai berikut:

function PlaySound(pzsSound:PChar;hMod:Cardinal;fdwSound:cardinal):boolean;

Parameter *pzsSound* adalah nama file WAV yang akan dimainkan atau jika fdwSound berisi konstanta SND_MEMORY maka pzsSound harus menunjuk ke alamat data file WAV yang ada di memori. *hMod* diisi sama dengan nol kecuali jika *fdwSound* berisi konstanta SND_RESOURCE. Untuk memainkan file WAV berulang-ulang fdwSound diisi SND_LOOP. Untuk informasi lebih lanjut bagaimana mengunakan PlaySound pembaca dipersilahkan membaca file help Windows SDK yang tersedia bersama Delphi.

Kekuatan PlaySound terletak pada kemudahan penggunaannya. Jika kita hanya ingin memainkan file WAV pendek untuk efek suara dan tidak membutuhkan mixing maka gunakan PlaySound. Namun jika kita ingin menggunakan beberapa file WAV yang dimainkan secara bersamaan, kita harus meninggalkan PlaySound.

Kekurangan PlaySound terletak pada tidak tersedianya kemampuan mixing suara. File WAV yang sebelumnya sedang dimainkan dengan PlaySound akan dihentikan jika kita memanggil PlaySound untuk memainkan file WAV lain. Dengan PlaySound kita tidak bisa memainkan musik latar bersamaan dengan efek-efek suara lain seperti suara ledakan dan lain-lain.

Cara berikutnya adalah dengan WaveMix.dll. Dengan menggunakan fungsi-fungsi yang ada pada file ini kita dapat memainkan beberapa file WAV secara bersamaan. Kekurangannya adalah jumlah maksimum file WAV yang dapat dimainkan adalah 8 file WAV. Cara menggunakan Wavemix.dll juga sedikit lebih rumit dibandingkan dengan PlaySound.

Cara ketiga adalah dengan MCI. Pada Delphi MCI driver dienkapsulasi menjadi objek TMediaPlayer. Fungsi-fungsi TMediaPlayer sangat lengkap meliputi audio dan video. TMediaPlayer digunakan jika kita ingin mengontrol memainkan atau merekam file audio/video dari perangkat CD_ROM player , MIDI sequencer, video player/recorder.

Jika kita hanya ingin memainkan file-file WAV, MIDI atau MP3 pengunaan TMediaPlayer merupakan pemborosan memori. Selain itu MCI tidak dirancang untuk game, melainkan untuk mengakses perangkat multimedia.

Cara keempat adalah dengan DirectSound. Dengan DirectSound file WAV yang dapat dimainkan bersamaan tidak dibatasi, selama memori komputer kita masih sanggup menampung semua data file WAV. DirectSound menyediakan bermacam-macam efek suara yang dapat kita gunakan seperti chorus, reverb, distorsi dan lain-lain. DirectSound juga mendukung pemrosesan suara 3D. Dengan suara 3D, game yang kita buat dapat menghasilkan efek surround sehingga game menjadi mengasyikkan untuk dimainkan.

5.2 Inisialisasi DirectSound.

Interface *IDirectSound* adalah interface utama DirectSound. Setelah IDirectSound diciptakan maka IDirectSound dapat digunakan untuk menciptakan sound buffer, mengatur level kooperatif dan lain-lain. Jika kita ingin menggunakan kelebihan DirectSound versi terbaru (versi 8.1), maka kita harus menggunakan interface IDirectSound8. Untuk saat ini kita hanya akan memfokuskan pembahasan pada interface IDirectSound agar program-program yang kita buat kompatibel dengan versi DirectSound sebelumnya.

Untuk menciptakan IDirectSound digunakan fungsi *DirectSoundCreate*,

Function DirectSoundCreate(lpGUID:PGUID;ppDS:IDirectSound; pUnkOuter:Iuknown):Hresult;

lpGUID

Alamat Globally Unique Identifier (GUID) yang mengidentifikasi perangkat sound card. *lpGUID* diisi dengan nil jika kita ingin menggunakan perangkat default atau diisi dengan harga yang dikembalikan oleh fungsi *DirectSoundEnumerate*. Untuk saat ini pembahasan akan difokuskan pada perangkat default.

ppDS

Variabel yang akan digunakan menampung pointer ke interface IDirectSound.

pUnkOuter

Variabel yang menampung alamat interface lain yang mengontrol IDirectSound (aggregasi COM). Harus diisi nil karena aggregasi belum didukung.

Keberhasilan pemanggilan fungsi DirectSoundCreate dapat dites dengan Succeeded atau Failed.

5.3 Mengatur Level Kooperatif.

Setelah objek DirectSound berhasil diciptakan, program aplikasi harus segera mengatur level kooperatif yang diinginkan sebelum menggunakan fungsi-fungsi lain objek DirectSound.

Level kooperatif diset dengan memanggil fungsi anggota IDirectSound *SetCoopeartiveLevel*,

Function SetCooperativeLevel(handle:HWND;dwLevel:cardinal):HResult;

Handle,

Handel window aplikasi.

dwLevel,

Level kooperatif yang diinginkan. Harga level kooperatif adalah seperti yang terteara pada tabel berikut.

Tabel Level Kooperatif

DDSCL_EXCLUSIVE	Untuk versi 8 keatas, harga ini menghasilkan efek yang sama dengan DDSCL_PRIORITY. Untuk versi sebelumnya harga ini menyebabkan DirectSound mengatur level aplikasi menjadi level eksklusif. Level eksklusif menyebabkan aplikasi kita adalah satu-satunya yang bisa didengar suara keluarannya ketika aplikasi kita menerima input fokus.
DDSCL_NORMAL	Level normal adalah level yang mengijinkan multitasking dan berbagi resource, dengan demikian aplikasi lain yang mengunakan perangkat keras sound card tetap bisa didengar walaupun tidak menerima input fokus. Pada level normal suara keluaran dibatasi hanya berformat 8 bit.
DDSCL_PRIORITY	Level prioritas. Pada level ini aplikasi bisa mengubah format primary buffer (primary buffer akan dijelaskan pada sub bab berikutnya) dengan memanggil fungsi anggota IDirectSound *SetFormat* dan *Compact*. Fungsi Compact yang digunakan untuk memindahkan blok memori sound card yang tidak terpakai jika ada, menjadi blok memori yang berurutan sehingga diperoleh blok memori tidak terpakai paling besar.
DDSCL_WRITEPRIMARY	Pada level ini aplikasi diberi akses langsung untuk menulis data pada

	primary buffer. Semua hal yang berkaitan dengan proses mixing dan memainkan buffer dilakukan oleh aplikasi. Karena level ini relatif rumit maka kita tidak akan menggunakan level ini.

5.4 Menciptakan Sound Buffer.

DirectSound buffer adalah objek yang digunakan untuk mengontrol aliran data suara dari suatu sumber ke suatu tujuan. Sumber ini bisa perangkat sintesizer, buffer lain, file atau resource sedangkan tujuan biasanya adalah primary buffer. Ada dua jenis sound buffer, yakni:primary buffer dan secondary buffer.

Untuk menciptakan sound buffer digunakan fungsi anggota IDirectSound *CreateSoundBuffer*,

```
FunctionCreateSoundBuffer(const lpDSBufferDesc:TDSBufferDesc; out lpIDirectSoundBuffer;
pUnkOuter:IUnknown):Hresult;
```

lpDSBufferDesc

Deskripsi sound buffer yang diinginkan, bertipe *TDSBufferDesc*. Sebelum memanggil CreateSoundBuffer struktur ini harus disiapkan terlebih dahulu.

lpIDirectSoundBuffer

Variabel yang menerima pointer ke interface IDirectSoundBuffer.

pUnkOuter

Aggregasi COM. Belum didukung DirectSound sehingga harus diisi nil.

Struktur TDSBufferDesc

Deklarasi struktur ini adalah sebagai berikut:

```
TDSBufferDesc_DX6 = packed record
  dwSize       : DWORD;
  dwFlags      : DWORD;
  dwBufferBytes : DWORD;
  dwReserved   : DWORD;
  lpwfxFormat  : PWaveFormatEx;
end;

TDSBufferDesc1 = TDSBufferDesc_DX6;
```

```
  PDSBufferDesc1 = ^TDSBufferDesc1;
  PCDSBufferDesc1 = PDSBufferDesc1;

  TDSBufferDesc_DX7 = packed record
    dwSize         : DWORD;
    dwFlags        : DWORD;
    dwBufferBytes  : DWORD;
    dwReserved     : DWORD;
    lpwfxFormat    : PWaveFormatEx;
    guid3DAlgorithm : TGUID;
  end;

  TDSBufferDesc_DX8 = packed record
    dwSize         : DWORD;
    dwFlags        : DWORD;
    dwBufferBytes  : DWORD;
    dwReserved     : DWORD;
    lpwfxFormat    : PWaveFormatEx;
    guid3DAlgorithm : TGUID;
  end;

{$IFDEF DIRECTX6}
  TDSBufferDesc = TDSBufferDesc_DX6;
{$ELSE}
  {$IFDEF DIRECTX7}
    TDSBufferDesc = TDSBufferDesc_DX7;
  {$ELSE}
    TDSBufferDesc = TDSBufferDesc_DX8;
  {$ENDIF}
{$ENDIF}
```

TDSBufferDesc digunakan untuk mendeskripsikan karakteristik sound buffer yang akan diciptakan. Untuk versi 7 ke atas TDSBufferDesc mendapat tambahan field baru yakni guid3DAlgorithm. Penjelasan lengkap tiap field sebagai berikut:

dwSize

Ukuran struktur dalam byte. Field ini harus diisi sebelum struktur digunakan.

dwFlags

Karakteristik yang ingin disertakan ketika menciptakan sound buffer. Tabel berikut menjelaskan konstanta-konstanta yang valid untuk field ini.

Tabel Konstanta dwFlags

DSBCAPS_CTRL3D	3D format. Flag ini tidak bisa dikombinasi dengan flag DSBCAPS_CTRLPAN serta tidak bisa digunakan pada buffer stereo.
DSBCAPS_CTRLFREQUENCY	Buffer memiliki kemampuan untuk mengontrol frekuensi. Flag ini tidak bisa di kombinasi dengan DSBCAPS_CTRLFX.
DSBCAPS_CTRLFX	Buffer mendukung pemrosesan efek suara. Flag ini tidak bisa dikombinasi dengan DSBCAPS_CTRLFREQUENCY dan hanya bisa untuk buffer berformat 8 bit atau 16 bit dengan jumlah channel tidak lebih dari dua (buffer stereo). Buffer juga harus cukup besar untuk menampung data sebesar DSBSIZE_FX_MIN.
DSBCAPS_CTRLPAN	Memiliki kemampuan mengontrol pan (balance). Flag ini tidak bisa dikombinasi dengan DSBCAPS_CTRL3D.
DSBCAPS_CTRLPOSITIONNOTIFY	Memiliki kemampuan untuk memberitahukan posisi.
DSBCAPS_CTRLVOLUME	Buffer memiliki kemampuan untuk mengatur volume.
DSBCAPS_GETCURRENTPOSITION2	Untuk mendapatkan posisi akurat saat ini.
DSBCAPS_GLOBALFOCUS	Buffer adalah sound buffer global. Buffer yang sedang dimainkan tetap dapat didengar walau aplikasi tidak lagi menerima input fokus dan pengguna beralih ke aplikasi lain (bahkan aplikasi yang menggunakan DirectSound). Namun hal ini tidak berlaku jika

	aplikasi lain yang menggunakan DirectSound tersebut pada level kooperatif DDSCL_WRITEPRIMARY.
DSBCAPS_LOCDEFER	Buffer dapat menerima resource saat sedang dimainkan.
DSBCAPS_LOCHARDWARE	Buffer menggunakan mixing secara hardware. Jika hardware mixing tidak didukung atau memori sound card tidak tersedia maka CreateSoundBuffer akan gagal. Aplikasi juga harus memastikan channel untuk mixing tersedia untuk buffer ini.
DSBCAPS_LOCSOFTWARE	Buffer menggunakan mixing secara software dan berada pada memori sistem walaupun mixing secara hardware tersedia dan memori sound card cukup.
DSBCAPS_MUTE3DATMAXDISTANCE	Buffer yang sedang dimainkan akan dihentikan bila memlebihi jarak maksimum (untuk suara 3D)
DSBCAPS_PRIMARYBUFFER	Buffer adalah primary buffer. Jika flag ini tidak digunakan buffer yang diciptakan adalah secondary buffer. Flag ini tidak dapat dikombinasi dengan DSBCAPS_CTRLFX. (Primary buffer dan secondary buffer akan dijelaskan pada sub.bab berikutnya).
DSBCAPS_STATIC	Buffer diletakkan pada memori sound card jika memori tersedia, jika tidak tersedia buffer akan diciptakan di memori sistem. Flag ini tidak dapat dikombinasi dengan DSBCAPS_CTRLFX.
DSBCAPS_STICKYFOCUS	Jika pengguna beralih ke aplikasi lain, buffer yang sedang dimainkan, yang menggunakan flag ini tetap dapat didengar sedangkan buffer lain yang tidak menggunakan flag ini tidak akan terdengar.

dwBufferBytes

Ukuran buffer dalam byte. Field ini harus berisi nol jika buffer yang ingin kita ciptakan adalah primary buffer. Untuk secondary buffer bisa diisi harga antara DSBSIZE_MIN hingga DSBSIZE_MAX.

dwReserved

Cadangan. Diisi sama dengan nol.

lpwfxFormat

Struktur bertipe PWaveFormatEx atau PWaveFormatExtensible yang menjelaskan format waveform buffer. Diisi sama dengan nol jika ingin menciptakan primary buffer. Selanjutnya format primary buffer dapat diubah dengan fungsi anggota IDirectSoundBuffer *SetFormat*.

guid3DAlgorithm

Jika tidak digunakan flag DSBCAPS_CTRL3D, maka field ini harus diisi dengan GUID_NULL. Field ini hanya dapat diterapkan pada mixing secara software. Untuk saat ini kita tidak menggunakannya.

Ketika level kooperatif berhasil diset maka kita harus menciptakan paling tidak satu secondary sound buffer untuk menyimpan dan memainkan sebuah suara.

5.4.1 Primary Buffer.

Primary buffer adalah buffer dimana data audio dicampur (mixing) dengan data audio lain yang telah ada di buffer tersebut untuk kemudian dikirim ke perangkat keras untuk dimainkan.

Sebenarnya ketika objek DirectSound diciptakan. DirectSound otomatis menciptakan primary buffer sehingga kita tidak perlu lagi menciptakan primary buffer lain. Namun menciptakan primary buffer baru tidak dilarang.

Untuk menciptakan primary buffer field struktur TDSBufferDesc harus diisi DSBCAPS_PRIMARYBUFFER.

Contoh listing berikut menjelaskan bagaimana menciptakan primary buffer,

```
Procedure CreatePrimary(out primDirectSound:IDirectSoundBuffer;
                const cooperativelevel:cardinal);
type EsoundError=class(Exception);
Var buffdesc:TDSBufferDesc;
    Sizeofstruc:integer;
    Hr:Hresult;
   Pcm:TWaveFormatEx;
```

```
begin
    sizeofstruc:=SizeOf(TDSBufferDesc);
    fillchar(buffdesc,sizeOfstruc,0);
    buffDesc.dwSize:=sizeOfstruc;
    buffDesc.dwFlags:=DSBCAPS_PRIMARYBUFFER;
    buffdesc.dwBufferBytes:=0;
    hr:=MyDirectSound.CreateSoundBuffer(buffdesc,PrimDirectSound,nil);
    if Failed(hr) then raise ESoundError.Create('Unable to create primary sound buffer');

    FillChar(PCM, SizeOf(TWaveFormatEx), 0);
    with PCM do
    begin
      wFormatTag:=WAVE_FORMAT_PCM;  //format PCM
      nChannels:=2;      //buffer stereo
      nSamplesPerSec:=44100; frekuensi sample 44,1KHz
      wBitsPerSample:=16;  //buffer 16 bit
      nBlockAlign:=(wBitsPerSample div 8) * nChannels;
      cbSize:=0;
      nAvgBytesPerSec:=nSamplesPerSec * nBlockAlign;
    end;
    if CooperativeLevel<>DSSCL_NORMAL then  //pada level kooperatif
       hr:=PrimDirectSound.SetFormat(@PCM); //normal kita tidak di
end;                                  //ijinkan mengubah format
                                      //primary buffer.
```

5.4.2 Secondary Buffer.

Primary buffer hanya bisa diciptakan satu untuk tiap aplikasi. Hal ini berbeda dengan secondary buffer. Secondary buffer bisa dibuat berapa pun yang kita inginkan selama memori mencukupi.

Secondary buffer adalah buffer yang berisi data-data audio yang siap untuk dimainkan. Umumnya programmer berinteraksi dengan secondary buffer untuk memainkan efek suara karena prosesnya lebih sederhana. Dengan berinteraksi secara langsung dengan secondary buffer proses mixing dan mengirim data-data di primary buffer ke sound card akan ditangani DirectSound secara internal.

Contoh berikut adalah adalah salah satu alternatif menciptakan secondary buffer. Tiap buffer yang diciptakan tidak dengan flag DSBCAPS_PRIMARYBUFFER akan dianggap sebagai secondary buffer.

```
function CreateSecondaryBuffer(out lpdsb: IDirectSoundBuffer): Boolean;
type EsoundError=class(Exception);
var dsbdesc  : TDSBUFFERDESC;
   PCM      : TWaveFormatEx;
   hr:HResult;
begin
 Result:=True;
 FillChar(dsbdesc, SizeOf(TDSBUFFERDESC), 0);
 FillChar(PCM, SizeOf(TWaveFormatEx), 0);
 with PCM do
 begin
  wFormatTag:=WAVE_FORMAT_PCM;
  nChannels:=2;
  nSamplesPerSec:=44100;
  wBitsPerSample:=16;
  nBlockAlign:=(wBitsPerSample * nChannels) div 8;
  nAvgBytesPerSec:=nSamplesPerSec * nBlockAlign;
  cbSize:=0;
 end;

 dsbdesc.dwSize:=SizeOf(TDSBUFFERDESC);
dsbdesc.dwFlags:=DSBCAPS_STATIC or DSBCAPS_GETCURRENTPOSITION2 or
              DSBCAPS_GLOBALFOCUS;
 dsbdesc.dwBufferBytes:=Time * PCM.nAvgBytesPerSec;
 dsbdesc.lpwfxFormat:=@PCM;
 hr:=MyDirectSound.CreateSoundBuffer(dsbdesc, lpdsb, nil);
 if Failed(hr) then
 begin
   Result:=False;
   raise ESoundError.Create('Unable to create secondary sound buffer');
 end;
end;
```

5.5 Menulis Data Audio ke Buffer.

Agar data file WAV dapat dimainkan maka data tersebut harus ditulis ke buffer. Untuk menulis data tersebut terlebih dahulu kita harus mendapatkan alamat secondary buffer, yakni dengan memanggil fungsi anggota IDirectSoundBuffer *Lock*. Fungsi ini menyiapkan seluruh atau sebagian buffer untuk ditulis serta mengembalikan pointer dimana data dapat ditulis.

```
Function Lock(dwOffset:Dword;dwBytes:Dword;
    Var ppvAudioPointer1:Pointer;
    Var lpdwAudioBytes1:Dword;
    Var ppvAudioPointer2:Pointer;
    Var lpdwAudioBytes2:Dword;dwFlags:Dword):HResult;
```

dwOffset

Nilai yang menyatakan posisi offset, dalam satuan byte, dihitung dari awal buffer hingga posisi dimana penguncian dimulai. Contoh: jika ingin mengunci buffer mulai posisi awal buffer maka parameter ini diisi 0. Parameter ini diabaikan jika paramater dwFlags bernilai DSBLOCK_FROMWRITECURSOR.

dwBytes

Ukuran dalam byte, bagian buffer yang akan dikunci.

ppvAudioPointer1

Pointer ke bagian pertama buffer yang terkunci.

lpdwAudioBytes1

Ukuran dari blok *ppvAudioPointer1*.

ppvAudioPointer2

Pointer ke bagian kedua buffer yang terkunci. Jika nilai yang dikembalikan nil maka ppvAudioPointer1 menunjuk seluruh buffer terkunci.

lpdwAudioBytes2

Ukuran dari blok *ppvAudioPointer2*. Jika ppvAudioPointer2 bernilai nil maka parameter ini bernilai 0.

dwFlags

Flag yang mengatur bagaimana proses penguncian dilakukan.

Tabel berikut menjelaskan konstanta yang valid untuk parameter ini.

Tabel Konstanta Flag Proses Penguncian (Lock).

DSBLOCK_FROMWRITECURSOR	Mulai penguncian dari posisi kursor penulisan. Parameter dwOffset diabaikan.
DSBLOCK_ENTIREBUFFER	Mengunci seluruh buffer.

	Parameter dwBytes diabaikan.

Setelah data telah ditulis maka buffer yang terkunci harus dibebaskan dari proses penguncian dengan memanggil fungsi *Unlock*.

Function Unlock(ppvAudioPointer1:Pointer;

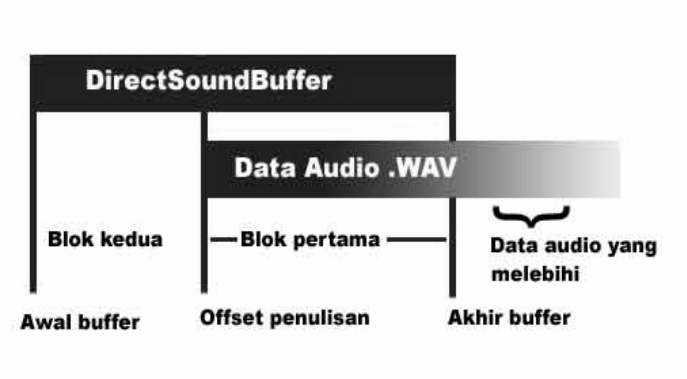
lpdwAudioBytes1:Dword;

ppvAudioPointer2:Pointer;

lpdwAudioBytes2:Dword):HResult;

ppvAudioPointer1, *lpdwAudioBytes1*, *ppvAudioPointer2* dan *lpdwAudioPointer2* adalah nilai-nilai yang dikembalikan oleh fungsi Lock.

Fungsi Lock mengembalikan dua pointer karena sifat DirectSoundBuffer yang sirkular. Jika ukuran data audio+offset penulisan tidak melebihi akhir buffer maka ppvAudioPointer2 berisi nil, jika melebihi maka ppvAudioPointer2 berisi alamat awal buffer. Data audio yang melebihi akan ditulis pada awal buffer ini.



Gambar 5.1 Skema penulisan data pada DirectSoundBuffer.

Kedua pointer ini hanya bisa ditulis (write-only) sehingga programmer seharusnya tidak membaca isi data yang ditunjuk oleh pointer ini karena data mungkin tidak valid. Contoh, jika sebuah buffer berada di memori sound card maka pointer yang dikembalikan fungsi Lock bisa berupa pointer ke suatu buffer sementara di memori sistem, setelah proses unlock maka data pada buffer sementara ditransfer ke memori sound card.

Fungsi Lock mengembalikan status keberhasilan proses. Salah satu error yang mungkin timbul adalah bahwa buffer telah hilang. Untuk mengembalikan buffer yang hilang digunakan fungsi *Restore*.

Setelah sound buffer di bebaskan dari penguncian, maka buffer ini siap dimainkan. Seandainya kita lupa membebaskan buffer dari proses penguncian pada bebrapa sound card suara yang dimainkan tidak akan terdengar.

5.6 Memainkan Buffer.

Untuk memainkan buffer yang telah berisi data audio kita menggunakan fungsi anggota IDirectSound *Play*.

Function Play(dwReserved1:Dword;dwPriority:Dword; dwFlags:Dword):HResult;

dwReserved1

Cadangan, harus bernilai 0.

dwPriority

Prioritas, digunakan oleh voice manager untuk mengatur hardware mixing. Prioritas terendah adalah 0 dan tertinggi 0FFFFFFFh. Jika buffer saat diciptakan tidak menggunakan flag DSBCAPS_LOCDEFER maka parameter ini harus diisi 0.

dwFlags

Flag yang menentukan bagaimana buffer dimainkan. Salah satu konstanta yang valid untuk parameter ini adalah DSBPLAY_LOOPING yang digunakan untuk memainkan buffer secara berulang-ulang. Jika kita ingin memainkan tidak berulang-ulang parameter ini kita isi dengan 0.

Fungsi Play akan gagal bila buffer hilang sehingga kita harus mengembalikan buffer tersebut dengan memanggil fungsi Restore.

5.7 Menghentikan Buffer yang Sedang Dimainkan.

Untuk menghentikan buffer yang sedang dimainkan kita menggunakan fungsi *Stop*. Fungsi ini adalah fungsi anggota IDirectSoundBuffer, fungsi ini tidak memerlukan parameter.

Function Stop:HResult;

5.8 Volume, Pan (Balance) dan Frekuensi.

5.8.1 Mengatur Volume.

Jika buffer diciptakan dengan flag DSBCAPS_CTRLVOLUME maka kita bisa mengubah-ubah volume buffer dengan fungsi *SetVolume* serta mendapatkan informasi volume buffer dengan *GetVolume*.

Function SetVolume(lVolume:integer):HResult;

Function GetVolume(var lVolume:integer):HResult;

lVolume

Volume yang diinginkan

Pada SetVolume, parameter ini berisi nilai volume yang diminta, sedangkan pada GetVolume, berisi nilai volume buffer saat ini. Parameter volume dinyatakan dalam satuan seperseratus desibel (dB). Jika kita ingin mengubah menurunkan volume menjadi 3 dB maka kita mengisi parameter ini dengan nilai -300. Harga maksimum yang diperbolehkan adalah sebesar DSBVOLUME_MAX dan minimum DSBVOLUME_MIN. Untuk saat ini DSBVOLUME_MAX didefiniskan sama dengan 0 dan DSBVOLUME_MIN sama dengan –10000.

5.8.2 Mengatur Pan (Balance).

Jika buffer diciptakan dengan flag DSBCAPS_CTRLPAN maka kita dapat mengatur volume relatif (balans) speaker kiri dan speaker kanan (jika komputer kita memiliki dua speaker). Fungsi anggota IDirectSoundBuffer *SetPan* dan *GetPan* digunakan untuk keperluan ini. SetPan digunakan untuk mengubah volume relatif, sedangkan GetPan digunakan untuk mendapatkan informasi volume relatif saat ini.

Function SetPan(lPan:integer):HResult;

Function GetPan(var lPan:integer):HResult;

lPan

Volume relatif speaker kiri dan speaker kanan, diukur dalam seperseratus desibel (dB). Nilai minimumnya adalah DSBPAN_LEFT dan maksimum DSBPAN_RIGHT. Saat ini DSBPAN_LEFT didefinisikan sebesar –10000 dan DSBPAN_RIGHT sama dengan 10000. Nilai defaultnya adalah DSBPAN_CENTER (sama dengan 0) yang berarti volume kiri dan kanan sama besar. Nilai lPan=-2173 berarti volume kiri berada pada volume maksimum dan volume kanan diturunkan sebanyak 21,73 dB. Nilai lPan=890 berarti volume kanan berada pada volume maksimum dan volume kiri diturunkan sebanyak 8,9 dB.

5.8.3 Mengatur Frekuensi.

Buffer yang diciptakan dengan flag DSBCAPS_CTRLFREQUENCY dapat diatur frekuensinya dengan *SetFrequency* dan *GetFrequency*.

Function SetFrequency(dwFrequency:Dword):HResult;

Function GetFrequency(var dwFrequency:Dword):HResult;

dwFrequency

Frekuensi, diukur dalam Hertz (Hz). Nilai parameter ini berkisar antara DSBFREQUENCY_MIN hingga DSBFREQUENCY_MAX. Untuk saat ini DSBFREQUENCY_MIN adalah 100 dan DSBFREQUENCY_MAX sama dengan 100000. Jika frekuensi diisi DSBFREQUNCY_ORIGINAL maka frekuensi dikembalikan ke nilai frekuensi default format buffer.

5.9 Finalisasi DirectSound.

Setelah aplikasi tidak membutuhkan DirectSound, kita harus membebaskan memori yang digunakan. Interface IDirectSound dan semua interface IDirectSoundBuffer yang telah diciptakan kita bebaskan dengan mengisinya sama dengan nil.

Bab 6
Membuat Unit uDirectSound.Pas

6.1 Pendahuluan.

Seperti halnya pada DirectDraw, untuk mempermudah kita menggunakan DirectSound, kita akan membuat enkapsulasinya. Ada dua kelas yang akan kita buat untuk keperluan tersebut yaitu *TWaveFile* dan *TSoundEffect*. TWaveFile berfungsi untuk proses pembacaan file WAV, sedangkan TSoundEffect berfungsi untuk menangani proses memainkan suara.

6.2 Listing Unit uDirectSound.

```
unit uDirectSound;
interface
uses windows,sysutils,MMSystem,classes,DirectSound;

const MaxVolume      =DSBVOLUME_MAX;
      MinVolume      =DSBVOLUME_MIN;
      LeftPan        =DSBPAN_LEFT;
      RightPan       =DSBPAN_RIGHT;
      CenterPan      =DSBPAN_CENTER;
      MaxFrequency   =DSBFREQUENCY_MAX;
      MinFrequency   =DSBFREQUENCY_MIN;
      InitialFrequency=DSBFREQUENCY_ORIGINAL;

type ESoundError=Class(Exception);
     EWaveError=class(Exception);
    TSoundBufferEx=Record
         Active:boolean;
         Buffer:IDirectSoundBuffer;
         Size:integer;
     end;
     TSoundBufferExs=array of TSoundBufferEx;
```

```
TSoundCaps=record
   scStatic:boolean;
   sc3D:boolean;
   scCtrlVolume:boolean;
   scCtrlPan:boolean;
   scCtrlFrequency:boolean;
   scCtrlDefault:boolean;
   scGlobalFocus:boolean;
   scStickyFocus:boolean;
 end;
TCooperativeLevel=(cplNormal,cplExclusive,cplPriority);
TSoundParam=record
    Handle:HWND;
    CooperativeLevel:TCooperativeLevel;
    Caps:TSoundCaps;
    PrimaryBufferCaps:TSoundCaps;
    Filename:string;
end;
TWaveFile=class
 private
   FFileSize: integer;
   FDataSize: integer;
      FWaveData: Pointer;
      FSampleData: Pointer;
      FFilename: TFilename;
      FWaveFormatEx: TWaveFormatEx;
 public
   constructor Create(const filename:string);
   destructor Destroy;override;
   procedure Free;
   property WaveData:Pointer read FWaveData;
   property SampleData:Pointer read FSampleData;
 published
   property Filename:TFilename read FFilename;
   property WaveFormatEx:TWaveFormatEx read FWaveFormatEx;
```

```
    property FileSize:integer read FFileSize;
    property DataSize:integer read FDataSize;
  end;
TSoundEffect=class
    private
      MySoundParam:TSoundParam;
      FLoaded:Boolean;
      FEffectNow: integer;
      FCount: Integer;
      FWAVList:TStringList;
      FFilename: String;
      FSoundFile:string;
      FNoSound: Boolean;
      MyDirectSound:IDirectSound;
      PrimDirectSound:IDirectSoundBuffer;
      BackDirectSound:TSoundBufferExs;
      procedure SetEffectNow(const Value: integer);
      procedure SetFilename(const Value: String);
      procedure SetNoSound(const Value: Boolean);
      function GetActive(index: integer): boolean;
      procedure SetActive(index: integer; const Value: boolean);
      function GetBuffer(index: integer): pointer;
      function GetSize(index: integer): integer;
      function GetWAVList(index: integer): string;
      function CreateSecondaryBuffer(out lpdsb: IDirectSoundBuffer;
                        SamplesPerSec : Integer;
                        Bits          : Word;
                        Stereo        : Boolean;
                        Time          : Integer;
                        isStatic      : Boolean) : Boolean; overload;
      function CreateSecondaryBuffer(out lpdsb: IDirectSoundBuffer;
                            waveformat:TWaveFormatEx;
                            const size:cardinal;param:TSoundParam) : Boolean;overload;
     function WriteDataToBuffer(lpdsb: IDirectSoundBuffer;
                    OffSet: DWord; var SoundData;
```

```
                    SoundBytes: DWord) : Boolean;
    public
      constructor Create(SoundParam:TSoundParam);
      destructor  Destroy;override;
      procedure   Free;
      procedure   Play;
      procedure   PlayLoop;
      procedure   Stop;
      procedure   SetVolume(const vol:integer);
      function    GetVolume:integer;
      procedure   SetPan(const pan:integer);
      function    GetPan:integer;
      procedure   SetFrequency(const freq:cardinal);
      function    GetFrequency:cardinal;
      procedure   Reload(const filename:string);overload;
      procedure   Reload(list:TStringList);overload;
      property    WAVList[index:integer]:string read GetWAVList;
      property    Active[index:integer]:boolean read GetActive write SetActive;
      property    Buffer[index:integer]:pointer read GetBuffer;
      property    Size[index:integer]:integer read GetSize;
      property    NoSound:Boolean read FNoSound write SetNoSound;
      property    EffectNow:integer read FEffectNow write SetEffectNow;
      property    Count:Integer read FCount;
      property    SoundFile:String read FSoundFile;
      property    Filename:String read FFilename write SetFilename;
    end;

implementation
{ TSoundEffect }

//format file
//SOUNDFX    //tag
//4         //jumlah file wav
//mysound1WAV //string nama file
//1          //1=active 0=tdk actice
```

```
//mysound2WAV
//1
//mysound3WAV
//1
//mysound4WAV
//1

constructor TSoundEffect.Create(SoundParam: TSoundParam);
var Mytxt:TextFile;
    str:string;
    ctr,sizeofstruc:integer;
    buffdesc:TDSBufferDesc;
    PCM:TWaveFormatEx;
    hr:HResult;
    myWave:TWaveFile;
    coopLevel:cardinal;
begin
  MySoundParam:=SoundParam;
  FCount:=0;
  FEffectNow:=0;
  FLoaded:=false;
  FSoundFile:='';
  FFilename:='';
  FNoSound:=False;
  FWAVList:=TStringList.Create;
  If FileExists(MySoundParam.Filename) then
  Begin
    if Succeeded(DirectSoundCreate(nil,MyDirectSound,nil)) then
    begin
      case MySoundParam.CooperativeLevel of
        cplNormal:coopLevel:=DSSCL_NORMAL;
        cplExclusive:coopLevel:=DSSCL_EXCLUSIVE;
        cplPriority:coopLevel:=DSSCL_PRIORITY;
      end;
      hr:=MyDirectSound.SetCooperativeLevel(MySoundParam.Handle,coopLevel);
```

```
if Failed(hr) then raise ESoundError.Create('Unable to set cooperative level');
sizeofstruc:=SizeOf(TDSBufferDesc);
fillchar(buffdesc,sizeOfstruc,0);
buffDesc.dwSize:=sizeOfstruc;

buffDesc.dwFlags:=DSBCAPS_PRIMARYBUFFER;
if (MySoundParam.PrimaryBufferCaps.sc3D) and
   (not MySoundParam.PrimaryBufferCaps.scCtrlPan) then
  buffDesc.dwFlags:=buffDesc.dwFlags or DSBCAPS_CTRL3D else
if ((MySoundParam.PrimaryBufferCaps.scCtrlPan)) or
   ((MySoundParam.PrimaryBufferCaps.sc3D) and
    (MySoundParam.PrimaryBufferCaps.scCtrlPan))then
  buffDesc.dwFlags:=buffDesc.dwFlags or DSBCAPS_CTRLPAN;

buffdesc.dwBufferBytes:=0;
hr:=MyDirectSound.CreateSoundBuffer(buffdesc,PrimDirectSound,nil);
if Failed(hr) then raise ESoundError.Create('Unable to create primary sound buffer');

FillChar(PCM, SizeOf(TWaveFormatEx), 0);
with PCM do
begin
  wFormatTag:=WAVE_FORMAT_PCM;
  nChannels:=2;
  nSamplesPerSec:=44100;
  wBitsPerSample:=16;
  nBlockAlign:=(wBitsPerSample div 8) * nChannels;
  cbSize:=0;
  nAvgBytesPerSec:=nSamplesPerSec * nBlockAlign;
end;
if MySoundParam.CooperativeLevel<>cplNormal then
   hr:=PrimDirectSound.SetFormat(@PCM);
if Failed(hr) then raise ESoundError.Create('Unable to set primary buffer format');
if (MySoundParam.Filename<>'') and (FileExists(MySoundParam.Filename)) then
begin
  AssignFile(MyTxt,MySoundParam.Filename);
```

```
Reset(MyTxt);
readln(MyTxt,str);
if str='SOUNDFX' then
begin
  readln(MyTxt,str);
  FCount:=StrToInt(str);
  SetLength(BackDirectSound,FCount);
  for ctr:=0 to FCount-1 do
  begin
    BackDirectSound[ctr].Buffer:=nil;
    BackDirectSound[ctr].Active:=false;
    BackDirectSound[ctr].Size:=0;
  end;
  for ctr:=0 to FCount-1 do
  begin
    readln(MyTxt,str);
    if FileExists(str) then
    begin
      try
        myWave:=TWaveFile.Create(str);
        CreateSecondaryBuffer(BackDirectSound[ctr].Buffer,
                        myWaveWAVeFormatEx,myWave.DataSize,MySoundParam);
        WriteDataToBuffer(BackDirectSound[ctr].Buffer,0,MyWave.SampleData^,MyWave.DataSize);
        BackDirectSound[ctr].Size:=myWave.FileSize;
      finally
        myWave.Free;
      end;
    end else str:='';
    FWAVList.Add(str);
    readln(MyTxt,str);
    BackDirectSound[ctr].Active:=(str='1');
  end;
end;
CloseFile(MyTxt);
FSoundFile:=MySoundParam.Filename;
```

```
      FLoaded:=true;
    end;
  end else raise ESoundError.Create('Unable to create DirectSound Object');
 end else raise ESoundError.Create('File:'+MySoundParam.Filename+' not found.');
end;

destructor TSoundEffect.Destroy;
var i:integer;
begin
 Stop;
 if FLoaded then
 begin
 begin
  FWAVList.Clear;
  FWAVList.Free;
  for i:=0 to FCount-1 do
  begin
   BackDirectSound[i].Buffer:=nil;
   BackDirectSound[i].Active:=false;
   BackDirectSound[i].Size:=0;
  end;
  PrimDirectSound:=nil;
  MyDirectSound:=nil;
 end;
 end;
 inherited;
end;

procedure TSoundEffect.Free;
begin
 if self<>nil then
 begin
  destroy;
  self:=nil;
 end;
```

```
end;

function TSoundEffect.GetActive(index: integer): boolean;
begin
  Result:=false;
  if (index>=0) and (index<FCount) then
    Result:=BackDirectSound[index].Active;
end;

function TSoundEffect.GetBuffer(index: integer): pointer;
begin
  Result:=nil;
  if (index>=0) and (index<FCount) then
   Result:=Pointer(BackDirectSound[index].Buffer);
end;

function TSoundEffect.GetSize(index: integer): integer;
begin
  Result:=0;
  if (index>=0) and (index<FCount) then
      Result:=BackDirectSound[index].Size;
end;

function TSoundEffect.GetWAVList(index: integer): string;
begin
  Result:='';
  if (index>=0) and (index<FCount) then
   Result:=FWAVList[index];
end;


procedure TSoundEffect.Play;
begin
  if (FLoaded) and (not FNoSound) and
    (BackDirectSound[FEffectNow].Active) then
```

```
  begin
    BackDirectSound[FEffectNow].Buffer.SetCurrentPosition(0);
    BackDirectSound[FEffectNow].Buffer.Play(0,0,0);
  end;
end;

procedure TSoundEffect.PlayLoop;
begin
   if (FLoaded) and (not FNoSound) and
      (BackDirectSound[FEffectNow].Active) then
       BackDirectSound[FEffectNow].Buffer.Play(0,0,DSBPLAY_LOOPING);
end;


procedure TSoundEffect.SetActive(index: integer; const Value: boolean);
begin
  if (index>=0) and (index<FCount) then
  BackDirectSound[index].Active:=Value;
end;

procedure TSoundEffect.SetEffectNow(const Value: integer);
begin
  if (Value>=0) and (Value<FCount) then
  FEffectNow := Value;
end;

procedure TSoundEffect.SetFilename(const Value: String);
begin
  if FileExists(Value) then
  FFilename := Value;
end;

procedure TSoundEffect.SetNoSound(const Value: Boolean);
begin
  FNoSound := Value;
```

```
end;

procedure TSoundEffect.Stop;
begin
  if BackDirectSound[FEffectNow].Buffer<>nil then
     BackDirectSound[FEffectNow].Buffer.Stop;
end;

function TSoundEffect.WriteDataToBuffer(lpdsb: IDirectSoundBuffer;
                        OffSet: DWord; var SoundData;
                        SoundBytes: DWord) : Boolean;
var AudioPtr1, AudioPtr2    : Pointer;
    AudioBytes1,AudioBytes2 : DWord;
    h : HResult;
    Temp : Pointer;
begin
  Result:=True;
  H:=lpdsb.Lock(OffSet, SoundBytes, AudioPtr1, AudioBytes1,
                              AudioPtr2, AudioBytes2, 0);
  if H = DSERR_BUFFERLOST then
  begin
   lpdsb.Restore;
   if lpdsb.Lock(OffSet, SoundBytes, AudioPtr1, AudioBytes1,
              AudioPtr2, AudioBytes2, 0) <> DS_OK then Result:=False;
  end else if H <> DS_OK then Result:=False;
  Temp:=@SoundData;
  Move(Temp^, AudioPtr1^, AudioBytes1);
  if AudioPtr2 <> nil then
  begin
   Temp:=@SoundData;
   Inc(Integer(Temp), AudioBytes1);
   Move(Temp^, AudioPtr2^, AudioBytes2);
  end;
  if lpdsb.UnLock(AudioPtr1, AudioBytes1, AudioPtr2, AudioBytes2) <> DS_OK
   then Result:=False;
```

```
end;

function TSoundEffect.CreateSecondaryBuffer(out lpdsb: IDirectSoundBuffer;
                            SamplesPerSec : Integer;
                            Bits          : Word;
                            Stereo        : Boolean;
                            Time          : Integer;
                            isStatic      : Boolean) : Boolean;
var dsbdesc  : TDSBUFFERDESC;
   PCM      : TWaveFormatEx;
   hr:HResult;
begin
 Result:=True;
 FillChar(dsbdesc, SizeOf(TDSBUFFERDESC), 0);
 FillChar(PCM, SizeOf(TWaveFormatEx), 0);
 with PCM do
 begin
  wFormatTag:=WAVE_FORMAT_PCM;
  if Stereo then nChannels:=2 else nChannels:=1;
  nSamplesPerSec:=SamplesPerSec;
  wBitsPerSample:=Bits;
  nBlockAlign:=(wBitsPerSample * nChannels) div 8;
  nAvgBytesPerSec:=nSamplesPerSec * nBlockAlign;
  cbSize:=0;
 end;

 dsbdesc.dwSize:=SizeOf(TDSBUFFERDESC);
 dsbdesc.dwFlags:=DSBCAPS_STATIC or DSBCAPS_GETCURRENTPOSITION2 or
              DSBCAPS_GLOBALFOCUS;
 dsbdesc.dwBufferBytes:=Time * PCM.nAvgBytesPerSec;
 dsbdesc.lpwfxFormat:=@PCM;
 hr:=MyDirectSound.CreateSoundBuffer(dsbdesc, lpdsb, nil);
 if Failed(hr) then
 begin
   Result:=False;
```

```
   raise ESoundError.Create('Unable to create secondary sound buffer');
 end;
end;

procedure TSoundEffect.Reload(const filename: string);
var myTxt:TextFile;
    ctr:integer;
    str:string;
    myWave:TWaveFile;
begin
  if (MySoundParam.Filename<>'') and (FileExists(MySoundParam.Filename)) then
  begin
    AssignFile(MyTxt,MySoundParam.Filename);
    Reset(MyTxt);
    readln(MyTxt,str);
    if str='SOUNDFX' then
    begin
      readln(MyTxt,str);
      FCount:=StrToInt(str);
      SetLength(BackDirectSound,FCount);
      for ctr:=0 to FCount-1 do
      begin
        BackDirectSound[ctr].Buffer:=nil;
        BackDirectSound[ctr].Active:=false;
        BackDirectSound[ctr].Size:=0;
      end;
      for ctr:=0 to FCount-1 do
      begin
        readln(MyTxt,str);
        if FileExists(str) then
        begin
          try
            myWave:=TWaveFile.Create(str);
            CreateSecondaryBuffer(BackDirectSound[ctr].Buffer,
                          myWaveWAVeFormatEx,myWave.DataSize,MySoundParam);
```

```
          writeDataToBuffer(BackDirectSound[ctr].Buffer,0,MyWave.SampleData^,MyWave.DataSize);
          BackDirectSound[ctr].Size:=myWave.FileSize;
        finally
          myWave.Free;
        end;
      end else str:='';
      FWAVList.Add(str);
      readln(MyTxt,str);
      BackDirectSound[ctr].Active:=(str='1');
    end;
  end;
  CloseFile(MyTxt);
  FSoundFile:=MySoundParam.Filename;
  FLoaded:=true;
 end;
end;

function TSoundEffect.CreateSecondaryBuffer(out lpdsb: IDirectSoundBuffer;
  waveformat: TWaveFormatEx;const size:cardinal;param:TSoundParam): Boolean;
var dsbdesc  : TDSBUFFERDESC;
  hr:HResult;
  sz:integer;
begin
 sz:=SizeOf(TDSBUFFERDESC);
 fillchar(dsbdesc,Sz,0);
 dsbdesc.dwSize:=sz;
 dsbdesc.dwFlags:=DSBCAPS_STATIC;

 if param.Caps.scStatic  then
 dsbdesc.dwFlags:=DSBCAPS_STATIC;
 if param.Caps.scGlobalFocus  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_GLOBALFOCUS;
 if param.Caps.scCtrlPan  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRLPAN;
 if param.Caps.scCtrlFrequency  then
```

```
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRLFREQUENCY;
 if param.Caps.scCtrlVolume  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRLVOLUME;
 if param.Caps.scCtrlDefault  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRLDEFAULT;
 if param.Caps.scStickyFocus  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_STICKYFOCUS;
 if (param.Caps.sc3D) and
    (not param.Caps.scCtrlPan) then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRL3D;

 dsbdesc.dwBufferBytes:=size;
 dsbdesc.lpwfxFormat:=@WaveFormat;
 hr:=MyDirectSound.CreateSoundBuffer(dsbdesc, lpdsb, nil);
 if Failed(hr) then
 begin
   Result:=False;
   raise ESoundError.Create('Unable to create secondary sound buffer');
 end;
end;
procedure TSoundEffect.Reload(list: TStringList);
var ctr:integer;
    str:string;
    myWave:TWaveFile;
begin
  if list<>nil then
  begin
    str:=list.Strings[0];
    if str='SOUNDFX' then
    begin
      str:=list.Strings[1];
      FCount:=StrToInt(str);
      SetLength(BackDirectSound,FCount);
      for ctr:=0 to FCount-1 do
      begin
```

```
      BackDirectSound[ctr].Buffer:=nil;
      BackDirectSound[ctr].Active:=false;
      BackDirectSound[ctr].Size:=0;
    end;
    for ctr:=0 to FCount-1 do
    begin
      str:=list.Strings[2*ctr+2];
      if FileExists(str) then
      begin
        try
          myWave:=TWaveFile.Create(str);
          CreateSecondaryBuffer(BackDirectSound[ctr].Buffer,
                        myWaveWAVeFormatEx,myWave.DataSize,MySoundParam);
          writeDataToBuffer(BackDirectSound[ctr].Buffer,0,MyWave.SampleData^,MyWave.DataSize);
          BackDirectSound[ctr].Size:=myWave.FileSize;
        finally
          myWave.Free;
        end;
      end else str:='';
      FWAVList.Add(str);
      str:=list.Strings[2*ctr+3];
      BackDirectSound[ctr].Active:=(str='1');
    end;
  end;
  FSoundFile:=MySoundParam.Filename;
  FLoaded:=true;
 end;
end;

function TSoundEffect.GetVolume: integer;
var res:integer;
   hr:HResult;
begin
 res:=0;
 hr:=BackDirectSound[FEffectNow].Buffer.GetVolume(res);
```

```
  if Failed(hr) then raise ESoundError.Create('Unable to get volume');
  Result:=res;
end;

procedure TSoundEffect.SetVolume(const vol: integer);
var hr:HResult;
begin
  hr:=BackDirectSound[FEffectNow].Buffer.SetVolume(vol);
  if Failed(hr) then raise ESoundError.Create('Unable to set volume');
end;

function TSoundEffect.GetFrequency: cardinal;
var res:cardinal;
    hr:HResult;
begin
  res:=0;
  hr:=BackDirectSound[FEffectNow].Buffer.GetFrequency(res);
  if Failed(hr) then raise ESoundError.Create('Unable to get frequency');
  Result:=res;
end;

function TSoundEffect.GetPan: integer;
var res:integer;
    hr:HResult;
begin
  res:=0;
  hr:=BackDirectSound[FEffectNow].Buffer.GetPan(res);
  if Failed(hr) then raise ESoundError.Create('Unable to get pan');
  Result:=res;
end;

procedure TSoundEffect.SetFrequency(const freq: cardinal);
var hr:HResult;
begin
  hr:=BackDirectSound[FEffectNow].Buffer.SetFrequency(freq);
```

```
  if Failed(hr) then raise ESoundError.Create('Unable to set frequency');
end;

procedure TSoundEffect.SetPan(const pan: integer);
var hr:HResult;
begin
  hr:=BackDirectSound[FEffectNow].Buffer.SetPan(pan);
  if Failed(hr) then raise ESoundError.Create('Unable to set pan');
end;

{ TWaveFile }

constructor TWaveFile.Create(const filename: string);
var fStream:TFileStream;
    poss:integer;
    chunk,dwType:string[4];
    dwLength,pdwend:integer;
begin
 try
  try
    fStream:=TFileStream.Create(filename,fmOpenRead);
    FFileSize:=fStream.Size;
    GetMem(FWaveData,FFileSize);
    FFilename:=filename;
    fStream.ReadBuffer(FWaveData^,FFileSize);

    fStream.Seek(0,soFromBeginning);
    SetLength(Chunk, 4);
    SetLength(dwType, 4);
    fStream.ReadBuffer(chunk[1],4);
    fStream.ReadBuffer(dwLength,4);
    fStream.ReadBuffer(dwType[1],4);
    if chunk='RIFF' then
    begin
      If dwType='WAVE' then
```

```
    begin
      poss:=fStream.Position;
      pdwEnd:=poss+dwLength-4;
      while poss<pdwEnd-1 do
      begin
        fStream.ReadBuffer(dwType[1],4);
        fStream.ReadBuffer(dwLength,4);
        if dwType='fmt 'then
        begin
          if dwLength<sizeOf(TWaveFormat) then
            raise EWaveError.Create(filename+'contains invalid wave format');
          fStream.ReadBuffer(FWaveFormatEx,dwLength);
        end;
        if dwType='data' then
        begin
          FDataSize:=dwLength;
          GetMem(FSampleData, FDataSize);
          Fstream.ReadBuffer(FSampleData^, FDataSize);
        end;
        poss:=fStream.Position;
      end;
    end else raise EWaveError.Create(filename+' is not a wave file.');
  end else raise EWaveError.Create(filename+' is not a wave RIFF format file.');
 finally
  fStream.Free;
 end;
except
  raise EWaveError.Create('Unable to open '+filename);
end;
end;

destructor TWaveFile.Destroy;
begin
 if FSampleData<>nil then
   FreeMem(FSampleData,FDataSize);
```

```
  if WaveData<>nil then
    FreeMem(FWaveData,FFileSize);
  inherited;
end;

procedure TWaveFile.Free;
begin
  if self<>nil then Destroy;
end;
end.
```

6.3 Format WAV File.

Sebelum kita menuju sub bab memainkan file WAV dengan DirectSound, kita harus mengetahui bagaimana file WAV disimpan.

File WAV terdiri atas bagian header dan bagian data. Pada bagian data disimpan format sampel suara dan data sampel-sampel suara. Sebuah sampel adalah bilangan yang menyatakan rata-rata amplitudo yang direkam sepanjang suatu interval tertentu.

Format sample suara berupa format WAVEFORMATEX. Pada Delphi format ini dideklarasikan sebagai tipe TWaveFormatEx di file MMSystem.pas.

```
  PWaveFormatEx = ^TWaveFormatEx;
  {$EXTERNALSYM tWAVEFORMATEX}
  tWAVEFORMATEX = packed record
    wFormatTag: Word;
{ tipe format WAVE_FORMAT_PCM=1=Pulse Code Modulation Format PCM }
    nChannels: Word;          {jumlah chanel 1=mono 2=stereo }
    nSamplesPerSec: DWORD;  { frekuensi sampling }
    nAvgBytesPerSec: DWORD;
{nAvgBytesPerSec  = perkiraan ukuran buffer }
{nAvgBytesPerSec  = nChannels*nSamplesPerSec*wBitsPerSample div 8 }
    nBlockAlign: Word;
{ ukuran data block, nBlockAlign=nChannels*wBitsPerSample div 8}
    wBitsPerSample: Word;   { jumlah bit tiap sample}
    cbSize: Word;
{untuk wFormatTag=WAVE_FORMAT_PCM cbSize tidak dipakai }
  end;
```

File WAV menggunakan tag-tag untuk identifikasi bagian header dan data dan berstruktur RIFF (Resource Interchange File Format). Format WAV adalah sebagai berikut:

RIFF File Format

Struktur dasar RIFF disebut chunk yang disusun seperti berikut:

<*rID*><*rLength*><*rData*>

<*rID*> = String 'RIFF' (4 byte) yang merupakan identifikasi struktur RIFF.

<*rLength*> = Panjang data yang mengikuti (4 byte).

<*rData*> = Data RIFF (rLength byte).

Blok <*rData*> bisa berisi format RIFF yang berbeda. Untuk file WAV isi <rData> adalah sebagai berikut:

Definisi Blok WAVEForm

<*rData*>=<*wID*><*Format Chunk*><*Data Chunk*>

di mana

<*wID*> = String 'WAVE' (4 byte) sebagai identifikasi file WAV

Definisi Format Chunk

<*Format Chunk*>=<*fID*><*fLength*><*WaveFormatEx*>

<*fID*> = String 'fmt ' (4 byte) sebagai identifikasi blok format.

(Catatan: string 'fmt ' mengandung karakter spasi).

<*fLength*> = Panjang data wave format yang mengikuti (4 byte).

<*WaveFormatEx*> = informasi wave format (TWaveFormatEx).

Definisi Data Chunk

Data chunk berisi data audio file WAV. Format data tergantung nilai <*WaveFormatEx.wFormatTag*> yang disimpan di format chunk.

<*Data Chunk*>=<*dID*><*dLength*><*dData*>

di mana:

<*dID*> = String 'data' (4 byte).

<*dLength*> = Panjang data yang mengikuti (4 byte).

<*dData*> = data audio WAV sesungguhnya sebanyak dLength byte.

6.4 Membuat Kelas TWaveFile.

Kelas TWaveFile adalah kelas yang akan kita gunakan untuk membaca file WAV dan memisahkan blok data dan blok format file WAV sehingga memudahkan kita untuk menyusun fungsi-fungsi memainkan file WAV. Kelas ini akan kita turunkan dari objek TComponent.

Kelas TWaveFile memiliki properti-properti (published) berikut:

- *WaveFormat* yang bertipe TWaveFormatEx
- *WaveDat*a yang bertipe pointer yang digunakan untuk menampung seluruh isi file WAV.
- *SampleData* yang bertipe pointer digunakan menyimpan data audio WAV
- *DataSize* bertipe integer yang digunakan menyimpan informasi ukuran SampleData dalam byte.
- *FileSize* bertipe integer digunakan untuk menyimpan ukuran file WAV seluruhnya.
- *FileName* bertipe string untuk menyimpan nama file yang telah dibaca.

Objek TWaveFile juga dilengkapi dengan metode-metode sebagai berikut:

- Constructor Create untuk inisialisasi objek dan membaca file WAV (publik).
- Destructor Destroy untuk finalisasi objek (publik).
- Procedure Free untuk finalisasi objek yang aman (publik).

Berikut ini adalah implementasi objek TWaveFile. Pada file *uDirectSound.Pas* juga dideklarasikan objek TSoundEffect yang berfungsi untuk memainkan efek suara. Objek TSoundEffect akan dibahas pada Sub Bab 6.5.

6.4.1 Inisialisasi.

Inisialisasi dikerjakan oleh konstructor Create. Pada inisialisasi, objek TWaveFile diciptakan, sekaligus membaca file WAV yang diinputkan oleh parameter *Filename*. Untuk itu kita deklarasikan objek *fStream* bertipe TFileStream yang akan kita gunakan membaca file. Jangan lupa untuk menambahkan unit *Classes* pada klausa uses karena kelas TFileStream dideklarasikan di unit tersebut. *Poss*, *dwLength*, *pdwEnd* adalah variabel yang akan mencatat posisi, panjang data dan posisi akhir data. *Chunk* dan *dwType* digunakan untuk menyimpan informasi tag file WAV .

```
constructor TWaveFile.Create(const filename: string);

var fStream:TFileStream;

   poss:integer;

   chunk,dwType:string[4];

   dwLength,pdwend:integer;

begin
```

Masukkan dalam blok *try…except* untuk menangkap setiap eksepsi yang timbul. Jika ada eksepsi buat eksepsi EWaveError. Selanjutnya buka file dalam mode baca

dan inisialisasi FFileSize, FWaveData dan FFilename. Variabel FfileSize,FwaveData, Ffilename berturut-turut menyimpan ukuran file WAV, seluruh data file WAV dan nama file WAV yang sedang dibuka.

```
try
 try
  fStream:=TFileStream.Create(filename,fmOpenRead);
  FFileSize:=fStream.Size;
  GetMem(FWaveData,FFileSize);
  FFilename:=filename;
  fStream.ReadBuffer(FWaveData^,FFileSize);
```

Setelah seluruh data dibaca maka posisi penunjuk fStream akan berada diakhir file oleh karena itu kita harus mengembalikan ke posisi awal file. Cara ini sebenarnya tidak efisien karena proses pembacaan dilakukan dua kali. Cara yang lebih baik adalah dengan membaca FWaveData.

```
  fStream.Seek(0,soFromBeginning);
  SetLength(Chunk, 4);
  SetLength(dwType, 4);
  fStream.ReadBuffer(chunk[1],4);
  fStream.ReadBuffer(dwLength,4);
  fStream.ReadBuffer(dwType[1],4);
```

Inisialisasi Chunk dan dwType. Baca 12 byte pertama file WAV yang berisi string ‘RIFF’, panjang data RIFF dan tipe data RIFF (untuk file WAV berisi string ‘WAVE’). Jika bukan ‘RIFF’ atau ‘WAVE’ timbulkan eksepsi EWaveError.

```
  if chunk='RIFF' then
  begin
   If dwType='WAVE' then
   begin
    poss:=fStream.Position;
    pdwEnd:=poss+dwLength-4;
```

Hitung posisi akhir data pdwEnd dimana pdwEnd=Poss+dwLength-4. dwLength harus dikurangi empat karena kita telah membaca tag ‘WAVE’ ke dwType. Tag ‘WAVE’ merupakan bagian data RIFF.

```
    while poss<pdwEnd-1 do
    begin
```

Lakukan proses looping selama Poss belum mencapai akhir file. Baca 8 byte data yang ditunjuk oleh penunjuk fStream.

```
fStream.ReadBuffer(dwType[1],4);
fStream.ReadBuffer(dwLength,4);
if dwType='fmt 'then
begin
```

Jika blok data yang mengikuti adalah blok format lakukan pengecekan terhadap dwLength. Nila dwLength tidak boleh kurang dari ukuran struktur TWaveFormat. Jika hal ini terjadi file WAV tidak valid sehingga perlu ditimbulkan eksepsi EWaveError. Jika tidak maka baca data sebanyak dwLength ke FWaveFormatEx.

```
 if dwLength<sizeOf(TWaveFormat) then
   raise EWaveError.Create(filename+'contains invalid wave format');
 fStream.ReadBuffer(FWaveFormatEx,dwLength);
end;
if dwType='data' then
begin
```

Jika blok yang mengikuti adalah blok data, copy isi dwLength ke FDataSize dan lakukan inisialisasi FSampleData sebanyak FDataSize. Selanjutnya membaca file sebanyak FDataSize dan menyimpannya di FSampleData.

```
 FDataSize:=dwLength;
 GetMem(FSampleData, FDataSize);
 Fstream.ReadBuffer(FSampleData^, FDataSize);
end;
poss:=fStream.Position;
```

Perbarui isi Poss karena posisi penunjuk fStream telah berubah.

```
  end;
 end else raise EWaveError.Create(filename+' is not a wave file.');
end else raise EWaveError.Create(filename+' is not a wave RIFF format file.');
finally
 fStream.Free;
```

Bebaskan objek fStream karena sudah tidak diperlukan.

```
end;
except
 raise EWaveError.Create('Unable to open '+filename);
end;
end;
```

Setelah constructor Create sukses dijalankan maka property-property *SampleData*, *DataSize* dan lain-lain telah siap diakses.

6.4.2 Finalisasi.

Setelah objek TWaveFile tidak diperlukan lagi maka memori yang dipakai harus dibebaskan. Demikian pula memori yang dipakai oleh data-data privat objek TWaveFile.

```
destructor TWaveFile.Destroy;
begin
  if FSampleData<>nil then
    FreeMem(FSampleData,FDataSize);
  if WaveData<>nil then
    FreeMem(FWaveData,FFileSize);
  inherited;
end;
```

6.5 Kelas TSoundEffect.

Kelas TSoundEffect adalah kelas yang kita gunakan untuk menangani proses memainkan efek suara dalam game yang akan kita buat.

Daftar nama file WAV yang digunakan sebagai efek suara disimpan dalam suatu file teks yang disusun dalam format tersendiri.

6.5.1 Format File Input.

Format penyimpanannya adalah sebagai berikut:

```
Tag pengenal='SOUNDFX'
Jumlah  file wav yang mengikuti (n)
Nama  file wav ke-1
Status aktif wav ke-1, aktif='1' tidak aktif='0'
Nama  file wav ke-2
Status aktif wav ke-2, aktif='1' tidak aktif='0'
…dst
Nama  file wav ke-n
Ststus aktif file wav ke-n,aktif='1' tidak aktif='0'
```

Contoh :

```
SOUNDFX
4
c:\My Documents\MyWav1WAV
1
```

```
c:\My Documents\MyWav2WAV
1
c:\My Documents\MyWav2WAV
1
c:\My Documents\MyWav2WAV
1
```

6.5.2 Struktur Data.

Tiap-tiap file WAV yang dibaca akan disimpan dalam sound buffer terpisah. Sound buffer bertipe *TSoundBufferEx*.

```
type   TSoundBufferEx=Record
       Active:boolean;
       Buffer:IDirectSoundBuffer;
       Size:integer;
     end;
```

Active

Status buffer. Jika active=true maka buffer dimainkan jika sebaliknya maka buffer ini tidak akan dimainkan.

Buffer

Sound buffer

Size

Ukuran file WAV yang telah dibaca.

Kelas TSoundEffect dirancang untuk memainkan beberapa file WAV. Oleh karena itu kita membutuhkan variabel yang menampung data file-file WAV tersebut. Kita akan menggunakan array dinamis untuk mengimplementasikan maksud tersebut.

```
type  TSoundBufferExs=array of TSoundBufferEx;
```

TSoundEffect akan menangani secara internal proses pembacaan file WAV dan menuliskan data ke sound buffer .

TSoundParam berisi deskripsi sound buffer, level kooperatif, nama file input dan lain-lain. Struktur data ini digunakan saat inisialisasi kelas TSoundEffect.

Struktur TSoundParam:

```
TSoundParam=record
    Handle:HWND;
    CooperativeLevel:TCooperativeLevel;
    Caps:TSoundCaps;
```

```
    PrimaryBufferCaps:TSoundCaps;
    Filename:string;
  end;
```

Handle

Handle window utama aplikasi.

CooperativeLevel

Level kooperatif yang diinginkan, nilai-nilai yang valid adalah *cplNormal*, *cplExclusive*, *cplPriority*.

Caps

Kapabilitas secondary buffer yang kita inginkan, bertipe *TsoundCaps*.

PrimaryBufferCaps

Kapabilitas primary buffer yang diinginkan, bertipe TSoundCaps.

Filename

Nama file teks yang berisi daftar file-file WAV yang akan digunakan sebagai efek suara.

Struktur TSoundCaps

```
  TSoundCaps=record
    scStatic:boolean;
    sc3D:boolean;
    scCtrlVolume:boolean;
    scCtrlPan:boolean;
    scCtrlFrequency:boolean;
    scCtrlDefault:boolean;
    scGlobalFocus:boolean;
    scStickyFocus:boolean;
   end;
```

TSoundEffect memiliki properti-properti berikut:

EffectNow bertipe integer, digunakan untuk mengindeks efek yang akan dimainkan. Bersifat dapat dibaca dan ditulis. Untuk menulis, nilai yang diinputkan harus valid sehingga perlu dilakukan pengecekan terlebih dahulu.

Count bertipe integer, mencatat jumlah file WAV yang telah dibaca dan ditulis kesound buffer. Properti ini bersifat read-only.

Active bertipe Boolean, digunakan untuk mengaktifkan atau menonaktifkan efek suara yang ditunjuk oleh suatu nomer indeks.

Buffer bertipe pointer, mengembalikan alamat sound buffer yang ditunuk oleh suatu nomer indeks.

Size bertipe integer, mengembalikan ukuran file WAV yang telah dibaca ke sound buffer yang ditunjuk oleh suatu indeks.

SoundFile bertipe string, mengembalikan nama file yang berisi daaftar file-file WAV.

NoSound bertipe Boolean digunakan untuk menonaktifkan semua suara.

TSoundEffect memiliki beberapa metode yang digunakan untuk berbagai keperluan.

6.5.3 Inisialisasi.

Inisialisasi objek TSoundEffect ditangani oleh konstruktor *Create*. Konstruktor Create akan melakukan proses pembacaan file WAV, menulis data ke secondary buffer dan mengisi data-data awal secara internal. Pembacaan file WAV akan memanfaatkan objek TWaveFile yang telah kita buat sebelumnya. Setelah konstruktor selesai menjalankan prosesnya maka secondary buffer siap dimainkan.

```
constructor Create(SoundParam:TSoundParam);
```

Sebelum memanggil konstruktor ini maka kita harus menyiapkan struktur data bertipe TSoundParam.

Berikut ini adalah penjelasan apa yang dikerjakan oleh konstruktor Create

```
constructor TSoundEffect.Create(SoundParam: TSoundParam);
var Mytxt:TextFile;
   str:string;
   ctr,sizeofstruc:integer;
   buffdesc:TDSBufferDesc;
   PCM:TWaveFormatEx;
   hr:HResult;
   myWave:TWaveFile;
   coopLevel:cardinal;
```

Kita siapkan variabel sementara untuk menampung berbagai data yang diperlukan. *MyTxt* digunakan untuk menyimpan informasi file teks yang akan dibaca. *Str* digunakan untuk menampung hasil pembacaan file teks yang ditunjuk oleh MyTxt. *Ctr* digunakan untuk proses looping dengan *For…To…Do*. *SizeOfStruc* adalah variable yang menampung ukuran struktur TDSBufferDesc. *BuffDesc* menampung informasi deskripsi sound buffer. *MyWave* adalah objek yang akan kita gunakan membaca file WAV. *PCM* menampung infomasi format WAV dan coopLevel untuk menampung informasi level kooperatif.

```
begin
 MySoundParam:=SoundParam;
 FCount:=0;
 FEffectNow:=0;
 FLoaded:=false;
 FSoundFile:='';
 FFilename:='';
 FNoSound:=False;
 FWAVList:=TStringList.Create;
```

Simpan SoundParam ke MySoundParam karena informasi yang ada di SoundParam masih akan kita perlukan. Proses inisialisasi variable *FCount* diisi nol karena belum ada file WAV yang dibaca. *FEffectNow* adalah variabel yang mencatat nomer indeks saat ini, kita isi nol. *FFilename* adalah variabel yang akan menampung nama file WAV. Untuk saat ini variabel ini belum begitu penting mengingat FFilename penulis rencanakan untuk pengembangan ke depan. FSoundFile menampung nama file yang berisi daftar file WAV. FNoSound menampung status apakah seluruh sound buffer bisa dimainkan atau tidak.. Asumsikan bahwa semua sound buffer bisa dimainkan sehingga kita isi dengan false. Terakhir adalah menginisialisasi FWAVList. FWAVList digunakan menyimpan daftar nama-nama file WAV yang telah dibaca.

```
 If FileExists(MySoundParam.Filename) then
```

Apakah file yang ditunjuk oleh MySoundParam.Filename ada? Jika ya masuk ke blok berikut:

```
  Begin
  if Succeeded(DirectSoundCreate(nil,MyDirectSound,nil)) then
```

Tes apakah DirectSound berhasil diinisialisasi.

```
  begin
   case MySoundParam.CooperativeLevel of
    cplNormal:coopLevel:=DSSCL_NORMAL;
    cplExclusive:coopLevel:=DSSCL_EXCLUSIVE;
    cplPriority:coopLevel:=DSSCL_PRIORITY;
   end;
   hr:=MyDirectSound.SetCooperativeLevel(MySoundParam.Handle,coopLevel);
   if Failed(hr) then raise ESoundError.Create('Unable to set cooperative level');
```

Isi informasi level kooperatif yang sesuai dengan yang ada pada MySoundParam.CooperativeLevel. Atur level kooperatif, jika gagal timbulkan eksepsi ESoundError.

```
   sizeofstruc:=SizeOf(TDSBufferDesc);
```

```
fillchar(buffdesc,sizeOfstruc,0);
buffDesc.dwSize:=sizeOfstruc;
```

Inisialisasi BuffDesc dengan nol serta isikan ukuran struktur TDSBufferDesc ke field dwSize. Dua langkah tersebut sangat penting untuk dilakukan karena DirectSound menginginkan field-field BuffDesc yang tidak terpakai diisi dengan nol dan field dwSize berisi ukuran struktur. Jika kita lupa melakukan hal ini kemungkinan besar timbul error yang menyatakan bahwa parameter tidak valid (DSERR_INVALIDPARAM)

```
buffDesc.dwFlags:=DSBCAPS_PRIMARYBUFFER;
```

Sound yang akan dibuat adalah primary sound buffer

```
if (MySoundParam.PrimaryBufferCaps.sc3D) and
  (not MySoundParam.PrimaryBufferCaps.scCtrlPan) then
 buffDesc.dwFlags:=buffDesc.dwFlags or DSBCAPS_CTRL3D else
if ((MySoundParam.PrimaryBufferCaps.scCtrlPan)) or
  ((MySoundParam.PrimaryBufferCaps.sc3D) and
   (MySoundParam.PrimaryBufferCaps.scCtrlPan))then
 buffDesc.dwFlags:=buffDesc.dwFlags or DSBCAPS_CTRLPAN;
```

Isikan flag kapabilitas yang ingin digunakan.

```
buffdesc.dwBufferBytes:=0;
```

Isikan ukuran buffer. Untuk primary sound buffer field dwBufferBytes harus diisi nol.

```
 hr:=MyDirectSound.CreateSoundBuffer(buffdesc,PrimDirectSound,nil);
if Failed(hr) then raise ESoundError.Create('Unable to create primary sound buffer');
```

Ciptakan primary sound buffer, jika gagal timbulkan eksepsi ESoundError

```
FillChar(PCM, SizeOf(TWaveFormatEx), 0);
with PCM do
begin
 wFormatTag:=WAVE_FORMAT_PCM;
 nChannels:=2;
 nSamplesPerSec:=44100;
 wBitsPerSample:=16;
 nBlockAlign:=(wBitsPerSample div 8) * nChannels;
 cbSize:=0;
 nAvgBytesPerSec:=nSamplesPerSec * nBlockAlign;
end;
```

Isikan format primary sound buffer. Format sound buffer yang kita gunakan adalah format PCM sehingga wFormatTag kita isi dengan WAVE_FORMAT_PCM karena DirectSound hanya mendukung format ini. Sound buffer stereo (nChannels=2), frekuensi sampling 44100 Hz dan bit tiap sampel 16.

```
if MySoundParam.CooperativeLevel<>cplNormal then
  hr:=PrimDirectSound.SetFormat(@PCM);
```

Ubah format jika level kooperatif adalah cplPriority atau cplExclusive. Jika level cplNormal, DirectSound tidak mengijinkan kita mengubah format primary sound buffer.

```
if Failed(hr) then raise ESoundError.Create('Unable to set primary buffer format');
```

Jika gagal timbulkan eksepsi ESoundError.

```
if (MySoundParam.Filename<>'') and (FileExists(MySoundParam.Filename)) then
begin
 AssignFile(MyTxt,MySoundParam.Filename);
 Reset(MyTxt);
 readln(MyTxt,str);
```

Buka dan baca string pertama pada file yang berisi daftar file WAV.

```
 if str='SOUNDFX' then
 begin
  readln(MyTxt,str);
  FCount:=StrToInt(str);
  SetLength(BackDirectSound,FCount);
  for ctr:=0 to FCount-1 do
  begin
   BackDirectSound[ctr].Buffer:=nil;
   BackDirectSound[ctr].Active:=false;
   BackDirectSound[ctr].Size:=0;
  end;
  for ctr:=0 to FCount-1 do
  begin
   readln(MyTxt,str);
   if FileExists(str) then
   begin
    try
     myWave:=TWaveFile.Create(str);
     CreateSecondaryBuffer(BackDirectSound[ctr].Buffer,
```

```
                    myWaveWAVeFormatEx,myWave.DataSize,MySoundParam);
          WriteDataToBuffer(BackDirectSound[ctr].Buffer,0,MyWave.SampleData^,MyWave.DataSize);
          BackDirectSound[ctr].Size:=myWave.FileSize;
        finally
          myWave.Free;
        end;
      end else str:='';
      FWAVList.Add(str);
      readln(MyTxt,str);
      BackDirectSound[ctr].Active:=(str='1');
    end;
  end;
  CloseFile(MyTxt);
  FSoundFile:=MySoundParam.Filename;
  FLoaded:=true;
 end;
 end else raise ESoundError.Create('Unable to create DirectSound Object');
 end else raise ESoundError.Create('File:'+MySoundParam.Filename+' not found.');
end;
```

6.5.4 Menciptakan Secondary Sound Buffer.

Pada kelas TSoundEffect terdapat fungsi *CreateSecondaryBuffer* yang di *overload* yang digunakan untuk menciptakan secondary sound buffer. Fungsi CreateSecondaryBuffer yang pertama digunakan bila kita memiliki informasi frekuensi sampling (*SamplePerSec*),

```
function TSoundEffect.CreateSecondaryBuffer(out lpdsb: IDirectSoundBuffer;
                    SamplesPerSec : Integer;
                    Bits        : Word;
                    Stereo      : Boolean;
                    Time        : Integer;
                    isStatic    : Boolean) : Boolean;
var dsbdesc  : TDSBUFFERDESC;
  PCM     : TWaveFormatEx;
  hr:HResult;
begin
 Result:=True;
```

```
FillChar(dsbdesc, SizeOf(TDSBUFFERDESC), 0);
FillChar(PCM, SizeOf(TWaveFormatEx), 0);
with PCM do
begin
 wFormatTag:=WAVE_FORMAT_PCM;
 if Stereo then nChannels:=2 else nChannels:=1;
 nSamplesPerSec:=SamplesPerSec;
 wBitsPerSample:=Bits;
 nBlockAlign:=(wBitsPerSample * nChannels) div 8;
 nAvgBytesPerSec:=nSamplesPerSec * nBlockAlign;
 cbSize:=0;
end;

dsbdesc.dwSize:=SizeOf(TDSBUFFERDESC);
dsbdesc.dwFlags:=DSBCAPS_STATIC or DSBCAPS_GETCURRENTPOSITION2 or
          DSBCAPS_GLOBALFOCUS;
dsbdesc.dwBufferBytes:=Time * PCM.nAvgBytesPerSec;
dsbdesc.lpwfxFormat:=@PCM;
hr:=MyDirectSound.CreateSoundBuffer(dsbdesc, lpdsb, nil);
if Failed(hr) then
begin
  Result:=False;
  raise ESoundError.Create('Unable to create secondary sound buffer');
end;
end;
```

Sedangkan fungsi CreateSecondaryBuffer digunakan bila kita memiliki informasi format wave yang akan dsimpan dalam buffer. Fungsi yang kedua ini sangat berhubungan erat dengan kelas TWaveFile untuk proses inisialisasi TSoundEffect.

Jika anda menggunakan fungsi CreateSecondaryBuffer yang pertama, apabila format wave yang anda inginkan tidak sama dengan format wave yang disimpan dalam file WAV, kemungkinan besar suara yang dihasilkan akan terdengar aneh.

Untuk menghindari hal ini, gunakan saja fungsi yang kedua dengan format wave yang diinputkan berasal dari properti WaveFormat kelas TWaveFile. Dengan cara ini dijamin suara yang dihasilkan terdengar bagus.

```
function TSoundEffect.CreateSecondaryBuffer(out lpdsb: IDirectSoundBuffer;
```

```
  waveformat: TWaveFormatEx;const size:cardinal;param:TSoundParam): Boolean;
var dsbdesc  : TDSBUFFERDESC;
    hr:HResult;
    sz:integer;
begin
 sz:=SizeOf(TDSBUFFERDESC);
 fillchar(dsbdesc,Sz,0);
 dsbdesc.dwSize:=sz;
 dsbdesc.dwFlags:=DSBCAPS_STATIC;

 if param.Caps.scStatic  then
 dsbdesc.dwFlags:=DSBCAPS_STATIC;
 if param.Caps.scGlobalFocus  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_GLOBALFOCUS;
 if param.Caps.scCtrlPan  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRLPAN;
 if param.Caps.scCtrlFrequency  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRLFREQUENCY;
 if param.Caps.scCtrlVolume  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRLVOLUME;
 if param.Caps.scCtrlDefault  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRLDEFAULT;
 if param.Caps.scStickyFocus  then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_STICKYFOCUS;
 if (param.Caps.sc3D) and
    (not param.Caps.scCtrlPan) then
 dsbdesc.dwFlags:=dsbdesc.dwFlags or DSBCAPS_CTRL3D;

 dsbdesc.dwBufferBytes:=size;
 dsbdesc.lpwfxFormat:=@WaveFormat;
 hr:=MyDirectSound.CreateSoundBuffer(dsbdesc, lpdsb, nil);
 if Failed(hr) then
 begin
   Result:=False;
   raise ESoundError.Create('Unable to create secondary sound buffer');
```

```
 end;
end;
```

6.5.4 Menulis Data WAV ke Sound Buffer.

```
function TSoundEffect.WriteDataToBuffer(lpdsb: IDirectSoundBuffer;
                    OffSet: DWord; var SoundData;
                    SoundBytes: DWord) : Boolean;
var AudioPtr1, AudioPtr2    : Pointer;
   AudioBytes1,AudioBytes2 : DWord;
   h : HResult;
   Temp : Pointer;
begin
  Result:=True;
  H:=lpdsb.Lock(OffSet, SoundBytes, AudioPtr1, AudioBytes1,
                        AudioPtr2, AudioBytes2, 0);
  if H = DSERR_BUFFERLOST then
  begin
   lpdsb.Restore;
   if lpdsb.Lock(OffSet, SoundBytes, AudioPtr1, AudioBytes1,
            AudioPtr2, AudioBytes2, 0) <> DS_OK then Result:=False;
  end else if H <> DS_OK then Result:=False;
  Temp:=@SoundData;
  Move(Temp^, AudioPtr1^, AudioBytes1);
  if AudioPtr2 <> nil then
  begin
   Temp:=@SoundData;
   Inc(Integer(Temp), AudioBytes1);
   Move(Temp^, AudioPtr2^, AudioBytes2);
  end;
  if lpdsb.UnLock(AudioPtr1, AudioBytes1, AudioPtr2, AudioBytes2) <> DS_OK
   then Result:=False;
end;
```

6.5.5 Proses Memainkan Buffer.

Untuk keperluan memainkan buffer, kita menambahkan metode *Play* dan *PlayLoop* ke objek TSoundEffect. PlayLoop digunakan untuk memainkan buffer

secara berulang-ulang. Play dan PlayLoop akan memainkan buffer yang bertipe TSoundBuffer yang ditunjuk oleh property EffectNow, hanya jika field Active buffer ini bernilai true.

```
procedure TSoundEffect.Play;
begin
  if (FLoaded) and (not FNoSound) and
     (BackDirectSound[FEffectNow].Active) then
  begin
```

Jika file input yang berisi daftar nama-nama file WAV berhasil dibaca dan FNoSound=true serta sound buffer yang ditunjuk oleh FEffectNow aktif maka

```
    BackDirectSound[FEffectNow].Buffer.SetCurrentPosition(0);
    BackDirectSound[FEffectNow].Buffer.Play(0,0,0);
```

Kembalikan posisi penunjuk ke awal sound buffer dan mainkan buffer yang ditunjuk oleh FEffectNow. Jika kita tidak mengembalikan penunjuk ke awal maka buffer akan dimainkan dimulai dari posisi penunjuk saat ini.

```
  end;
end;
```

Untuk proses memainkan buffer secara berulang-ulang caranya mirip dengan Play hanya terdapat perbedaan parameter yang diinputkan pada fungsi Play milik interface IDirectSoundBuffer Play yakni DSBPLAY_LOOPING.

```
procedure TSoundEffect.PlayLoop;
begin
  if (FLoaded) and (not FNoSound) and
     (BackDirectSound[FEffectNow].Active) then
      BackDirectSound[FEffectNow].Buffer.Play(0,0,DSBPLAY_LOOPING);
end;
```

6.5.6 Proses Menghentikan Buffer.

Buffer yang ditunjuk properti EffectNow yang sedang dimainkan dapat dihentikan dengan metode *Stop*.

6.5.7 Mengubah Volume.

TSoundEffect kita tambahkan kemampuan untuk mengubah volume. Untuk proses ini kita lengkapi objek TSoundEffect dengan metode *SetVolume* dan *GetVolume* yang masing-masing digunakan untuk mengatur dan mendapatkan nilai volume buffer yang ditunjuk oleh property EffectNow. Kedua metode ini akan menimbulkan eksepsi *ESoundError* bila buffer tidak diciptakan dengan kemampuan mengubah volume (DSBCAPS_CTRLVOLUME).

6.5.8 Mengubah Pan (Balance).

TSoundEffect kita tambahkan kemampuan untuk mengubah pan. Untuk proses ini kita lengkapi objek TSoundEffect dengan metode *SetPan* dan *GetPan* yang masing-masing digunakan untuk mengatur dan mendapatkan nilai pan buffer yang ditunjuk oleh property EffectNow. Kedua metode ini akan menimbulkan eksepsi *ESoundError* bila buffer tidak diciptakan dengan kemampuan mengubah pan (DSBCAPS_CTRLPAN).

6.5.9 Mengubah Frekuensi.

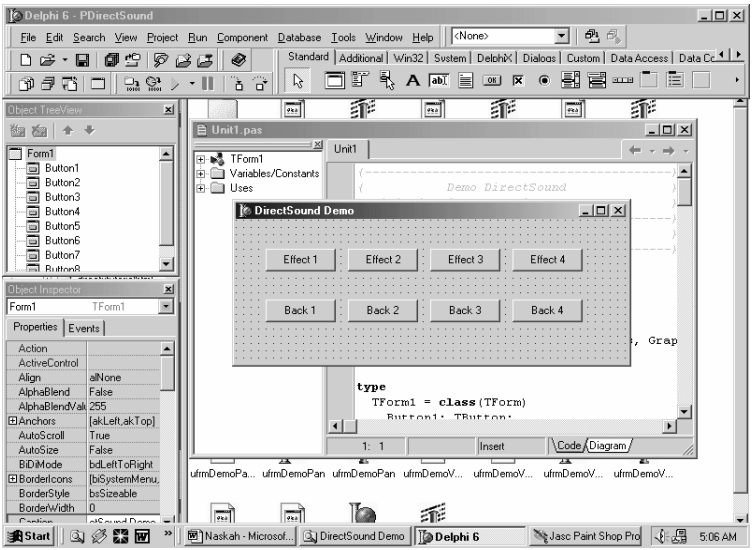
TSoundEffect juga kita tambahkan kemampuan untuk mengubah frekuensi. Untuk proses ini kita lengkapi objek TSoundEffect dengan metode *SetFrequency* dan *GetFrequency* yang masing-masing digunakan untuk mengatur dan mendapatkan frekuensi buffer yang ditunjuk oleh property EffectNow. Kedua metode ini akan menimbulkan eksepsi *ESoundError* bila buffer tidak diciptakan dengan kemampuan mengubah frekuensi (DSBCAPS_CTRLFREQUENCY).

6.5.10 Finalisasi.

Untuk membebaskan semua memori yang digunakan meliputi, membebaskan memori semua sound buffer, memori objek dan lain-lain. Untuk proses finalisasi kita tambahkan destruktor *Destroy* dan prosedur *Free*. Free digunakan untuk finalisasi yang aman.

Bab 7
Demo Program 2
Menambahkan Efek Suara

7.1 Listing Program 2.1 Inisialisasi DirectSound.



Gambar 7.1 Demo 2.1

File project demo 2.1 terdapat pada direktori Pemrograman Game Dengan DirectX\Bab 7\Demo 1\PDirectSound.dpr. Program dalam format executablenya di direktori yang sama dengan nama PDirectSound.exe. Berikut ini adalah listing Unit1.pas

```
{----------------------------------------}
{        Demo DirectSound              }
{Inisialisasi dan memainkan suara      }
{----------------------------------------}
{Copyright (c) 2002 Zamrony P Juhara    }
{----------------------------------------}
unit Unit1;
```

```
interface

uses
  Windows, Messages, SysUtils,Classes, Graphics, Controls, Forms,
  Dialogs,uDirectSound, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
  private
    SoundEffect:TSoundEffect;
    { Private declarations }
  public
    { Public declarations }
  end;

var
```

```
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
var soundParam:TSoundParam;
begin
  soundParam.Handle:=Handle;
  soundParam.CooperativeLevel:=cplExclusive;
  soundParam.Caps.scStatic:=true;
  soundParam.Caps.scCtrlVolume:=true;
  soundParam.Filename:='sndfx.txt';
  soundEffect:=TSoundEffect.Create(soundParam);
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  soundEffect.Free;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  soundeffect.EffectNow:=0;
  soundeffect.Play;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  soundeffect.EffectNow:=1;
  soundeffect.Play;
end;

procedure TForm1.Button3Click(Sender: TObject);
```

```
begin
  soundeffect.EffectNow:=2;
  soundeffect.Play;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  soundeffect.EffectNow:=3;
  soundeffect.Play;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
  soundeffect.EffectNow:=4;
  soundeffect.PlayLoop;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
  soundeffect.EffectNow:=5;
  soundeffect.PlayLoop;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  soundeffect.EffectNow:=6;
  soundeffect.PlayLoop;
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
  soundeffect.EffectNow:=7;
  soundeffect.PlayLoop;
end;
```

```
end.
```

7.2 Penjelasan Listing Program 2.1.

7.2.1 Inisialisasi.

```
procedure TForm1.FormCreate(Sender: TObject);
var soundParam:TSoundParam;
begin
  soundParam.Handle:=Handle;
  soundParam.CooperativeLevel:=cplExclusive;
  soundParam.Caps.scStatic:=true;
  soundParam.Caps.scCtrlVolume:=true;
  soundParam.Filename:='sndfx.txt';
  soundEffect:=TSoundEffect.Create(soundParam);
end;
```

Untuk inisialisasi kita siapkan variabel sementara SoundParam bertipe TSoundParam untuk menampung berbagai informasi yang diperlukan untuk proses inisialisasi objek SoundEffect. SoundParam.Handle kita isi dengan handle form utama kita. Level kooperatif yang kita pakai adalah level eksklusif dan semua secondary sound buffer adalah buffer statis dengan kemampuan untuk mengontrol volume. Field Filename diisi dengan file daftar WAV kita yaitu *sndfx.txt*

7.2.2 Event OnClick Button.

Tiap-tiap efek suara dimainkan dengan mengklik Button1-Button8. Button1 sampai Button 4 memainkan sound buffer tanpa looping dangan metode Play milik SoundEffect, sedangkan Button5- Button8 memainkan sound buffer secara looping dengan metode PlayLoop milik objek SoundEffect. Sebelum memanggil Play atau PlayLoop, properti EffectNow milik SoundEffect di isi dengan nomer efek suara yang diinginkan. Jika kita tidak mengisi properti ini maka play atau PlayLoop akan memainkan efek suara yang ditunjuk oleh EffectNow.

Yang perlu diperhatikan, efek suara pertama dalam daftar dimulai dari indeks nol dan efek terakhir di indeks dengan n-1 dimana n adalah jumlah efek suara. Jika EffectNow diisi dengan harga yang tidak valid seperti –1 atau n+1 maka harga yang diisikan akan diabaikan dan EffectNow akan berisi nilai valid terakhir yang diisikan.

7.2.3 Finalisasi.

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
```

```
  soundEffect.Free;
end;
```

Bebaskan memori objek SoundEffect dengan memanggil metode Free soundEffect.

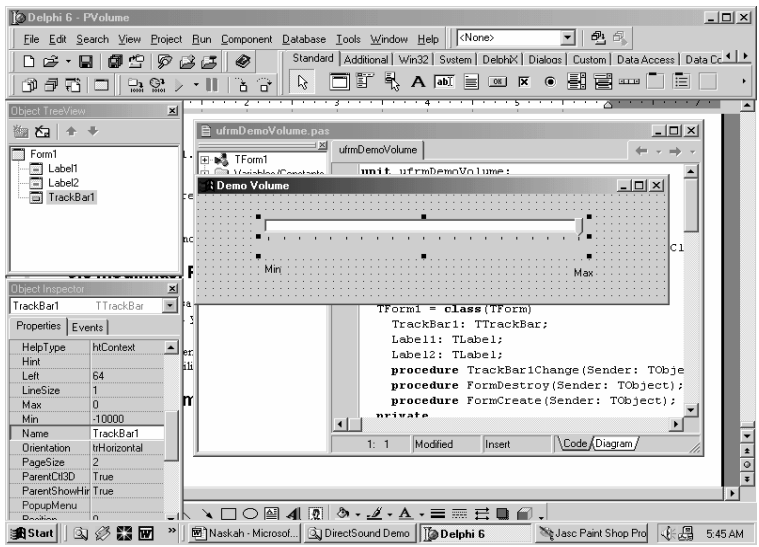
7.3 Modifikasi Program 2.1.

Modifikasi bisa dilakukan dengan menambahkan tombol baru yang berfungsi untuk menghentikan efek suara. Efek suara yang dimainkan secara looping pada program 2.1 akan terus dimainkan sampai program dihentikan.

Untuk menghentikan sebuah efek suara, isi nomer efek suara yang akan dihentikan, kemudian panggil metode *Stop* milik SoundEffect.

7.4 Listing Program 2.2 Mengatur Volume Suara.

File project demo 2.2 terdapat pada direktori Pemrograman Game Dengan DirectX\Bab 7\Demo 2\PVolume.dpr. Program dalam format executablenya di direktori yang sama dengan nama PVolume.exe. Berikut ini adalah listing UfrmDemoVolume.pas:



Gambar 7.2 Demo 2.2

```
unit ufrmDemoVolume;
```

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls,uDirectsound;

type
  TForm1 = class(TForm)
    TrackBar1: TTrackBar;
    Label1: TLabel;
    Label2: TLabel;
    procedure TrackBar1Change(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    sound:TSoundEffect;
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  sound.SetVolume(trackbar1.Position);
end;
procedure TForm1.FormDestroy(Sender: TObject);
begin
  sound.Free;
end;
procedure TForm1.FormCreate(Sender: TObject);
var soundparam:TSoundParam;
begin
```

```
  fillchar(soundparam,sizeof(TSoundParam),0);
  soundparam.Handle:=Handle;
  soundparam.CooperativeLevel:=cplNormal;
  soundparam.Caps.scCtrlVolume:=true;
  soundparam.PrimaryBufferCaps.scCtrlVolume:=true;
  soundparam.Filename:='sndfx.txt';
  sound:=TSoundEffect.Create(soundparam);
  sound.EffectNow:=0;
  sound.PlayLoop;
end;
end.
```

7.5 Penjelasan Listing Program 2.2.

7.5.1 Inisialisasi.

```
procedure TForm1.FormCreate(Sender: TObject);
var soundParam:TSoundParam;
begin
  soundParam.Handle:=Handle;
  soundParam.CooperativeLevel:=cplExclusive;
  soundParam.Caps.scStatic:=true;
  soundParam.Caps.scCtrlVolume:=true;
  soundParam.Filename:='sndfx.txt';
  soundEffect:=TSoundEffect.Create(soundParam);
  sound.EffectNow:=0;
  sound.PlayLoop;
end;
```

Proses inisialisasi hampir sama dengan demo 2.1. Perbedaannya terletak pada baris-baris terakhir metode FormCreate dimana kita langsung memainkan efek suara anjing menggonggong secara looping. Agar sound buffer dapat kita atur volumenya field scCtrlVolume kita isi dengan true. Jika kita lupa mengatur field ini akan timbul error DSERR_CONTROLUNAVAIL yang menyatakan bahwa kita mencoba mengontrol volume buffer yang tidak dilengkapi dengan kemampuan mengubah volume.

7.5.2 Event OnChange TrackBar1.

```
procedure TForm1.TrackBar1Change(Sender: TObject);
```

begin

sound.SetVolume(trackbar1.Position);

end;

Event ini kita gunakan karena kita akan mengubah volume efek suara no 0 (suara anjing menggonggong) tiap kali TrackBar1 posisinya berubah. Pada saat mendesain form jangan lupa untuk mengatur properti Max dan Min TrackBar1. Properti Min kita isi dengan nilai yang sama dengan DSBVOLUME_MIN yakni –10000 dan Max sama dengan DSBVOLUME_MAX yakni 0.

Tiap kali terjadi event onChange maka kita set volume dengan memanggil metode SetVolume milik objek Sound dengan parameter input posisi TrackBar1 saat ini.

7.5.3 Finalisasi.

procedure TForm1.FormDestroy(Sender: TObject);

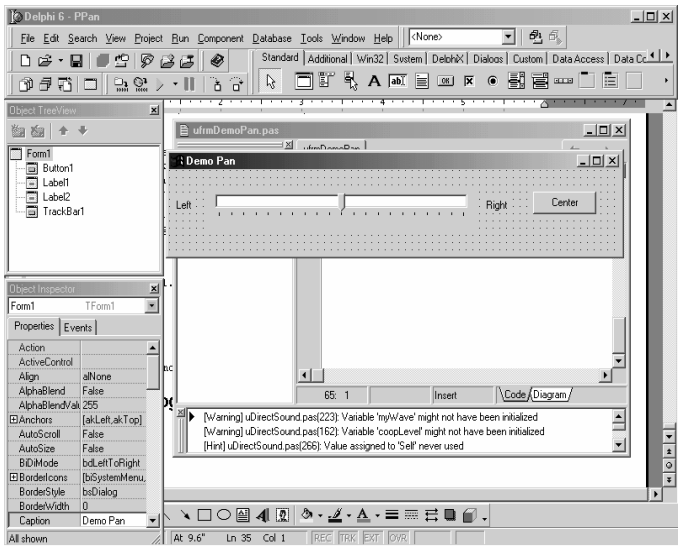
begin

sound.Free;

end;

Bebaskan memori objek sound karena sudah tidak diperlukan.

7.6 Listing Program 2.3 Mengatur Pan (Balance) Suara.



Gambar 7.3 Demo 2.3

File project demo 3.3 terdapat pada file Pemrograman Game Dengan DirectX\Bab 7\Demo 3\PPan.dpr dan executablenya bernama PPan.exe di direktori yang sama. Berikut ini adalah listing ufrmDemoPan.pas:

```
unit ufrmDemoPan;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls,uDirectSound, StdCtrls;
type
  TForm1 = class(TForm)
    TrackBar1: TTrackBar;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    Sound:TSoundEffect;
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
var soundparam:TSoundParam;
begin
  fillchar(soundparam,sizeof(TSoundParam),0);
  soundparam.Handle:=Handle;
  soundparam.CooperativeLevel:=cplNormal;
  soundparam.Caps.scCtrlPan:=true;
  soundparam.Caps.scStatic:=true;
```

```
  soundparam.Filename:='sndfx.txt';
  sound:=TSoundEffect.Create(soundparam);
  sound.EffectNow:=0;
  sound.PlayLoop;
end;
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  sound.SetPan(trackbar1.Position);
end;
procedure TForm1.FormDestroy(Sender: TObject);
begin
  sound.Free;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  sound.SetPan(CenterPan);
  Trackbar1.Position:=sound.GetPan;
end;
end.
```

7.7 Penjelasan Listing Program 2.3.

7.7.1 Inisialisasi.

```
procedure TForm1.FormCreate(Sender: TObject);
var soundparam:TSoundParam;
begin
  fillchar(soundparam,sizeof(TSoundParam),0);
  soundparam.Handle:=Handle;
  soundparam.CooperativeLevel:=cplNormal;
  soundparam.Caps.scCtrlPan:=true;
  soundparam.Caps.scStatic:=true;
  soundparam.Filename:='sndfx.txt';
  sound:=TSoundEffect.Create(soundparam);
  sound.EffectNow:=0;
  sound.PlayLoop;
```

```
end;
```

Proses inisialisasi demo 2.3 sama persis dengan proses inisialisasi demo 2.2, kecuali field scCtrlVolume pada demo 2.2 kita ganti dengan scCtrlPan. Field ini kita isi dengan true agar sound buffer bisa kita atur balancenya. Jika kita lupa mengatur field ini akan timbul error DSERR_CONTROLUNAVAIL yang menyatakan bahwa kita mencoba mengontrol pan buffer yang tidak dilengkapi dengan kemampuan mengubah pan.

7.7.2 Event OnChange TrackBar1.

```
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  sound.SetPan(trackbar1.Position);
end;
```

Event ini kita gunakan karena kita akan mengubah pan efek suara no 0 (suara anjing menggonggong) tiap kali TrackBar1 posisinya berubah. Pada saat mendesain form jangan lupa untuk mengatur properti Max dan Min TrackBar1. Properti Min kita isi dengan nilai yang sama dengan DSBPAN_MIN yakni –10000 dan Max sama dengan DSBPAN_MAX yakni 10000.

Tiap kali terjadi event onChange maka kita set pan dengan memanggil metode SetPan milik objek Sound dengan parameter input posisi TrackBar1 saat ini.

7.7.3 Event OnClick Button1.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  sound.SetPan(CenterPan);
  Trackbar1.Position:=sound.GetPan;
end;
```

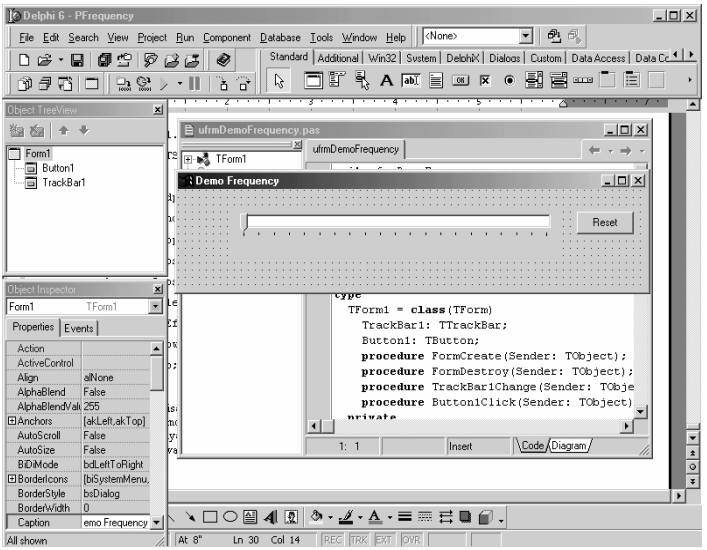
Button1 digunakan untuk mengembalikan pan buffer ke posisi center. Saat Button1 diklik SetPan dipanggil dengan menggunakan konstanta CenterPan sebagai parameter. Posisi pan yang baru didapatkan dengan memanggil fungsi GetPan. Hasil pemanggilan fungsi ini diisikan ke properti Position milik TrackBar1 untuk mengembalikan posisi Trackbar1 ke center.

7.7.4 Finalisasi.

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  sound.Free;
end;
```

Bebaskan memori yang dipakai objek Sound.

7.8 Listing Program 2.4 Mengatur Frekuensi Suara.



Gambar 7.4 Demo 2.4

File project demo 2.4 terdapat pada file Pemrograan Game Dengan DirectX\Bab 7\ Demo 1\PFrequency.dpr dan executablenya di direktori yang sama bernama PFrequency.exe. Berikut ini adalah listing ufrmDemoFrequency.pas:

```
unit ufrmDemoFrequency;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,uDirectSound, ComCtrls, StdCtrls;

type
  TForm1 = class(TForm)
    TrackBar1: TTrackBar;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
```

```
  private
    { Private declarations }
  public
    sound:TSoundEffect;
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
var soundparam:TSoundParam;
    initfreq:cardinal;
begin
  fillchar(soundparam,sizeof(TSoundParam),0);
  soundparam.Handle:=Handle;
  soundparam.CooperativeLevel:=cplNormal;
  soundparam.Caps.scCtrlFrequency:=true;
  soundparam.Filename:='sndfx.txt';
  sound:=TSoundEffect.Create(soundparam);
  sound.EffectNow:=0;
  sound.PlayLoop;
  initfreq:=sound.GetFrequency;
  trackbar1.Position:=initfreq;
end;
procedure TForm1.FormDestroy(Sender: TObject);
begin
  sound.Free;
end;
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  sound.SetFrequency(Trackbar1.Position);
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
```

```
  Sound.SetFrequency(InitialFrequency);
  trackbar1.Position:=sound.GetFrequency;
end;
end.
```

7.9 Penjelasan Listing Program 2.4.

7.9.1 Inisialisasi.

```
procedure TForm1.FormCreate(Sender: TObject);
var soundparam:TSoundParam;
    initfreq:cardinal;
begin
  fillchar(soundparam,sizeof(TSoundParam),0);
  soundparam.Handle:=Handle;
  soundparam.CooperativeLevel:=cplNormal;
  soundparam.Caps.scCtrlFrequency:=true;
  soundparam.Filename:='sndfx.txt';
  sound:=TSoundEffect.Create(soundparam);
  sound.EffectNow:=0;
  sound.PlayLoop;
  initfreq:=sound.GetFrequency;
  trackbar1.Position:=initfreq;
end;
```

Proses inisialisasi demo 2.4 hampir sama dengan proses inisialisasi demo 2.2 dan 2.3, kecuali field scCtrlVolume pada demo 2.2 atau scCtrlPan pada 2.3 kita ganti dengan scCtrlFrequency. Field ini kita isi dengan true agar sound buffer bisa kita atur frekuensinya. Jika kita lupa mengatur field ini akan timbul error DSERR_CONTROLUNAVAIL yang menyatakan bahwa kita mencoba mengontrol frekuensi buffer yang tidak dilengkapi dengan kemampuan mengubah frekuensi. Pada dua baris terakhir kita mengambil informasi frekuensi sampling default. Frekuensi sampling ini adalah frekuensi dari format WAV dan mengisikan ke properti Position TrackBar1.

7.9.2 Event OnChange TrackBar1.

```
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  sound.SetFrequency(Trackbar1.Position);
```

```
end;
```

Event ini kita gunakan karena kita akan mengubah frekuensi sampling efek suara no 0 (suara anjing menggonggong) tiap kali TrackBar1 posisinya berubah. Pada saat mendesain form jangan lupa untuk mengatur properti Max dan Min TrackBar1. Properti Min kita isi dengan nilai yang sama dengan DSBFREQUENCY_MIN yakni 100 dan Max sama dengan DSBFREQUENCY_MAX yakni 100000.

Tiap kali terjadi event onChange maka kita set frekuensi dengan memanggil metode SetFrequency milik objek Sound dengan parameter input posisi TrackBar1 saat ini.

7.9.3 Event OnClick Button1.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Sound.SetFrequency(InitialFrequency);
  trackbar1.Position:=sound.GetFrequency;
end;
```

Button1 digunakan untuk mengembalikan frekuensi sampling buffer ke frekuensi sampling default. Saat Button1 diklik SetFrequency dipanggil dengan menggunakan konstanta InitialFrequency sebagai parameter. Frekuensi yang baru, didapatkan dengan memanggil fungsi GetFrequency. Hasil pemanggilan fungsi ini diisikan ke properti Position milik TrackBar1 untuk mengembalikan posisi Trackbar1 ke posisi frekuensi default.

7.9.4 Finalisasi.

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  sound.Free;
end;
```

Bebaskan memori yang dipakai objek Sound.

Bab 8
Menggunakan DirectInput

8.1 Pendahuluan.

Jika anda mengamati game-game yang ada, terutama game bertipe fighting, kadang-kadang pemain harus menekan kombinasi beberapa tombol secara bersamaan agar karakter game melakukan suatu gerakan tertentu, misalnya melakukan sliding atau mengeluarkan suatu jurus.

Jika anda pernah membuat game menggunakan event WM_KEYDOWN untuk menangani input dari keyboard anda akan menemukan beberapa keterbatasan.

Keterbatasan ini adalah:

- Adanya delay setelah sebuah tombol ditekan pertama kali. Contohnya ketika anda mengetikkan suatu huruf. Huruf akan langsung tercetak dilayar ketika tombol keyboard kita tekan, setelah itu akan terjadi delay sekitar setengah detik dimana tidak ada huruf yang tercetak dilayar walaupun tombol keyboard masih ditekan. Setelah delay ini habis maka akan tercetak huruf terus menerus sampai tombol kita lepaskan. Hal ini diciptakan untuk mengurangi sensitifitas penekanan tombol keyboard agar kita tidak terganggu dengan penekanan tombol keyboard yang berlebihan. Untuk aplikasi game, hal seperti ini akan mengurangi keasikan bermain.
- Tidak bisa mendeteksi adanya tombol kombinasi.. Informasi tombol yang diberikan oleh event WM_KEYDOWN adalah informasi tombol yang sedang ditekan. Jika ada tombol lain yg ditekan sementara suatu tombol juga sedang ditekan, informasi tombol yang dikembalikan adalah tombol yang lebih dulu ditekan. Akibatnya kita tidak dapat mendeteksi tombol kombinasi.

Dulu programer game DOS memecahkan masalah ini dengan membelokkan hardware interupsi 9h yang dibangkitkan ketika sebuah tombol ditekan dan membaca port 21h untuk membaca scan code tombol yang sedang ditekan. Untuk aplikasi game yang berjalan di Windows, membaca port secara langsung sangat beresiko. Pada Windows 2000, Windows NT dan Windows XP, jika kita mengakses port secara langsung, misal dengan contoh penggalan program berikut:

```
Asm
   In al,20h
End;
```

Windows akan menimbulkan eksepsi dengan pesan “privileged instructions”. Pada Windows NT/2000/XP hanya device driver yang memiliki akses langsung ke perangkat keras. Jika proses pembacaan input dengan membaca

langsung port dikerjakan dalam looping utama game akibatnya program berjalan tidak seperti yang kita inginkan karena pesan “privileged instructions” akan muncul terus.

Dengan DirectInput, untuk menggunakan perangkat input sangat mudah, mulai dari keyboard, mouse hingga joystick force feedback.

8.2 Inisialisasi DirectInput.

Interface *IDirectInput* adalah interface utama DirectInput. Setelah IDirectInput diciptakan maka IDirectInput dapat digunakan untuk menciptakan input device, mengatur level kooperatif dan lain-lain. Jika kita ingin menggunakan kelebihan DirectInput versi terbaru (versi 8), maka kita harus menggunakan interface IDirectInput8. Untuk saat ini kita hanya akan memfokuskan pembahasan pada interface IDirectInput agar program-program yang kita buat kompatibel dengan versi DirectInput sebelumnya. Mungkin pada edisi berikutnya kita akan membahas DirectInput versi 8 dengan segala kelebihannya.

Untuk menciptakan IDirectInput digunakan fungsi *DirectInputCreate*,

Function DirectSoundCreate(instance:cardinal;dwVersion:cardinal;ppDI:IDirectInput; pUnkOuter:Iuknown):Hresult;

instance

Handle instance dari aplikasi.

dwVersion

Versi DirectInput

ppDI

Variabel yang akan digunakan menampung pointer ke interface IDirectInput.

pUnkOuter

Variabel yang menampung alamat interface lain yang mengontrol IDirectInput (aggregasi COM). Harus diisi nil karena aggregasi belum didukung.

Keberhasilan pemanggilan fungsi DirectInputCreate dapat dites dengan Succeeded atau Failed.

8.3 Menciptakan Device.

Setelah objek utama DirectInput berhasil diciptakan, selanjutnya kita ciptakan device untuk tiap-tiap input yang ingin kita gunakan. Interface *IDirectInputDevice* kita gunakan untuk keperluan ini.

Untuk menciptakan device kita gunakan fungsi anggota interface IDirectInput *CreateDevice*.

function CreateDevice(const rguid : TGUID; out lplpDirectInputDevice : IDirectInputDevice; pUnkOuter : IUnknown) : HResult; stdcall;

rguid

GUID dari perangkat input. GUID diperoleh dari proses enumerasi atau dengan variabel yang telah didefinisikan di unit DirectInput yakni *GUID_SysKeyboard* dan *GUID_SysMouse*. Karena kita masih dalam tahap perkenalan dengan DirectInput, penulis hanya akan membahas proses menciptakan device dengan variabel yang telah didefinisikan di Unit DirectInput. Untuk menciptakan device joystick kita perlu mendapatkan informasi GUID joystick yang terpasang di komputer. Oleh karena itu kita harus melakukan proses enumerisasi. Khusus untuk joystick pembaca jangan menggunakan *GUID_Joystick* karena GUID ini adalah GUID produk bukan GUID instance joystick.

lplpDirectInputDevice

Objek DirectInputDevice yang akan diciptakan.

pUnkOuter

Aggregasi COM, untuk saat ini tidak digunakan sehingga harus diisi nil.

Keberhasilan pemanggilan ini bisa kita ketahui dengan menggunakan fungsi Failed atau Succeeded

8.3.1 Keyboard Device.

Untuk menciptakan keyboard device bisa digunakan contoh rutin berikut:

```
procedure CreateKeyboardDevice(DI:IDirectInput;out keyboard:IDirectInputDevice);
begin
  DI.CreateDevice(GUID_SysKeyboard,keyboard,nil);
end;
```

8.3.2 Mouse Device.

Mirip dengan keyboard device, untuk menciptakan mouse device bisa digunakan contoh rutin berikut:

```
procedure CreateMouseDevice(DI:IDirectInput;out  mouse:IDirectInputDevice);
begin
  DI.CreateDevice(GUID_SysMouse,mouse,nil);
end;
```

8.3.3 Joystick Device.

Untuk menciptakan joystick kita harus melakukan proses enumerasi untuk mendapatkan GUID joystick yang terpasang di komputer. Untuk melakukan enumerasi kita menggunakan fungsi anggota interface IDirectInput *EnumDevices*

function EnumDevices(dwDevType : Cardinal; lpCallback : TDIEnumDevicesCallback; pvRef : Pointer; dwFlags : Cardinal) : HResult; stdcall;

Parameter-parameter:

dwDevType

Tipe device. Beberapa konstanta berikut adalah konstanta yang bisa digunakan untuk dwDevType

DIDEVTYPEJOYSTICK_UNKNOWN	Joystick sembarang
DIDEVTYPEJOYSTICK_TRADITIONAL	Joystick tradisional seperti yang ada pada mesin-mesin game arcade jaman dulu.
DIDEVTYPEJOYSTICK_FLIGHTSTICK	Joystick yang bentuknya mirip pengendali pada pesawat terbang. Joystick ini biasanya digunakan untuk game-game flight simulator.
DIDEVTYPEJOYSTICK_GAMEPAD	Joystick berbentuk gamepad seperti yang ada di mesin Playstation.
DIDEVTYPEJOYSTICK_RUDDER	Joystick yang memiliki kemampuan untuk melakukan rotasi pada sumbu z
DIDEVTYPEJOYSTICK_WHEEL	Joystick berbentuk stir mobil. Biasanya digunakan untuk game-game balap.
DIDEVTYPEJOYSTICK_HEADTRACKER	Joystick yang berbentuk mirip topi yang digunakan untuk melacak gerakan kepala.

Untuk DirectX versi 8 konstanta DIDEVTYPE** menjadi DI8DEVTYPE**. Ada beberapa tambahan konstanta baru pada versi 8, namun untuk sementara tidak akan kita bahas terlebih dahulu.

Selain konstanta-konstanta diatas kita juga dapat menggunakan konstanta berikut ini untuk mengenumerasi kelas device.

DIDEVTYPE_JOYSTICK	Joystick sembarang sama dengan DIDEVTYPEJOYSTICK_GAMEPAD

lpCallBack

Alamat rutin yang akan dipanggil bila sebuah joystick ditemukan (callback). Pada rutin inilah kita akan melakukan proses penciptaan device. Pembahasan tentang callback ini akan dijelaskan setelah ini.

pvRef

Pointer yang dilewatkan ke rutin callback tiap kali rutin callback dipanggil.

dwFlags

Flag yang mendeskripsikan scope proses enumerasi. Konstanta berikut ini valid untuk dwFlags

DIEDFL_ALLDEVICES	Proses enumerasi mencari semua device yang terinstall di komputer. Defaultnya adalah DIEDFL_ALLDEVICES
DIEDFL_ATTACHEDONLY	Proses enumerasi hanya mencari device yang sedang terpasang dan terinstall di komputer.
DIEDFL_FORCEFEEDBACK	Proses enumerasi hanya mencari device yang mendukung force feedback.
DIEDFL_INCLUDEALIASES	Proses enumerasi menyertakan device yang meruapak alias dari device lain.
DIEDFL_INCLUDEHIDDEN	Proses enumerasi menyertakan device yang merupakan device tersembunyi. Device tersembunyi adalah device fiktif yang dibuat oleh device driver sehingga dapat menghasilkan event mouse dan keyboard.
DIEDFL_INCLUDEPHANTOM	Proses enumerasi menyertakan device yang merupakan place holder bagi devcie yang nantinya mungkin ada.

Berikut ini adalah deklarasi tipe prosedur callback yang akan dipanggil tiap kali enumerasi berhasil menemukan suatu device.

TDIEnumDeviceObjectsCallback = function (var lpddoi : TDIDeviceObjectInstance; pvRef : Pointer) : Integer; stdcall;

Parameter-parameter

lpddoi

Objek instance device. Variabel ini adalah variabel yang menampung informasi device yang ditemukan. Variabel ini bertipe TDIDeviceObjectinstance. Field struktur ini yang nanti kita perlukan untuk menciptakan device adalah *guidInstance* bertipe TGUID.

pvRef

Pointer yang dilewatkan ke rutin callback tiap kali rutin callback dipanggil.

Setelah kita mengetahui cara melakukan enumerasi, maka berikut ini adalah contoh rutin untuk menciptakan joystick device untuk sembarang joystick yang ditemukan. Jika proses ini berhasil menemukan sebuah device, maka interface device ini diciptakan dan disimpan di JoyStickDev. Jika proses menciptakan device gagal, kita lanjutkan dengan mencari device yang lain, untuk itu nilai kembalian fungsi JoyCallBack kita isi DIENUM_CONTINUE. Jika berhasil maka proses enumerasi kita hentikan dengan mengembalikan nilai DIENUM_STOP. Fungsi dibawah mengasumsikan bahwa objek DirectInput DI bersifat global dan dapat diakses.

```
Function JoyCallBack((var lpddoi : TDIDeviceObjectInstance; pvRef : Pointer) : Integer;
Var hr:Hresult;
Begin
  Hr:= DI.CreateDevice(lpddoi.guidInstance,JoyStickDev,nil);
 If Failed(hr) then result:=DIENUM_CONTINUE else
Result:=DIENUM_STOP;
End;


procedure CreateMouseDevice(DI:IDirectInput;out  mouse:IDirectInputDevice);
begin
  DI.EnumDevices(DIDEVTYPEJOYSTICK_UNKNOWN,JoyCallBack,nil,DIEDFL_ATTACHEDONLY);
end;
```

8.4 Mengatur Level Kooperatif.

Seperti halnya komponen DirectDraw dan DirectSound, untuk menggunakan DirectInput dengan sukses, kita harus mengatur level kooperatif yang kita inginkan. Berbeda dengan DirectDraw dan DirectSound, pada DirectInput level kooperatif di atur oleh objek device bukan oleh objek utama DirectInput.

Untuk mengatur level kooperatif kita menggunakan fungsi *SetCooperativeLevel*,

```
function SetCooperativeLevel(hwnd : HWND; dwFlags : Cardinal) : HResult; stdcall;
```

hwnd

Handle form utama aplikasi

dwFlags

Flag yang mendeskripsikan level kooperatif yang diinginkan.

Tabel Level Kooperatif DirectInput Device

DISCL_BACKGROUND	Aplikasi meminta akses background. Dengan level kooperatif ini meskipun window aplikasi bukan window yang aktif aplikasi tetap dapat menerima input. Flag ini tidak dapat dikombinasi dengan DISCL_FOREGROUND.
DISCL_EXCLUSIVE	Aplikasi meminta akses ekslusif. Dengan level ini tidak ada objek instance device lain yang diijinkan mengakses device yang sedang diakses oleh aplikasi. Namun aplikasi lain yang berada pada level non eksklusif tetap dapat menggunakan input. Efek sampingnya adalah kursor akan hilang. Flag ini tidak dapat dikombinasi dengan DISCL_NONEXCLUSIVE.
DISCL_FOREGROUND	Aplikasi hanya dapat menerima input bila window aplikasi sedang aktif. Flag ini tidak dapat dikombinasi dengan DISCL_BACKGROUND.
DISCL_NONEXCLUSIVE	Instance device lain yang mengakses perangkat input masih dapat menerima input. Flag ini tidak dapat dikombinasi dengan DISCL_EXCLUSIVE.
DISCL_NOWINKEY	Menonaktifkan tombol Windows pada keyboard

8.5 Mengatur Format Data.

Sebelum aplikasi dapat membaca perangkat input. Aplikasi harus memberitahukan format data yang dibutuhkan kepada DirectInput.

Untuk mengatur format data digunakan fungsi anggota IDirectInputDevice *SetDataFormat*,

function SetDataFormat(var lpdf : TDIDataFormat) : HResult; stdcall;

lpdf

Format data perangkat input bertipe TDIDataFormat.

Struktur *TDIFormat* adalah sebagai berikut:

PDIDataFormat = ^TDIDataFormat;

```
TDIDataFormat = packed record
  dwSize     : Cardinal;
  dwObjSize  : Cardinal;
  dwFlags    : Cardinal;
  dwDataSize : Cardinal;
  dwNumObjs  : Cardinal;
  rgodf      : PDIObjectDataFormat;
end;
```

dwSize

Ukuran struktur TDIDataFormat dalam byte

dwObjSize

Ukuran struktur TDIObjectDataFormat dalam byte.

dwFlags

Flag yang mendeskripsikan atribut data lain

Tabel Flag untuk Format Data

DIDF_ABSAXIS	Aksis dalam mode absolute. Tidak dapat dikombinasi dengan DIDF_RELAXIS
DIDF_RELAXIS	Aksis dalam mode relatif. Tidak dapat dikombinasi dengan DIDF_ABSAXIS

dwDataSize

Ukuran paket data yang dikembalikan oleh perangkat input. Nilainya harus kelipatan 4 dan nilainya harus lebih besar dari data yang paling besar

dwNumObjs

Jumlah objek dalam array *rgodf*

rgdof

Alamat array bertipe *TDIObjectDataFormat*

Struktur TDIObjectDataFormat:

```
PDIObjectDataFormat = ^TDIObjectDataFormat;
TDIObjectDataFormat = packed record
  pguid   : PGUID;
```

```
  dwOfs   : Cardinal;
  dwType  : Cardinal;
  dwFlags : Cardinal;
end;
```

8.5 Meminta Akses Perangkat Input (Acquire).

Setelah format data diatur maka agar kita bisa menggunakan kita harus meminta ijin untuk menggunakan perangkat input tersebut. Proses ini disebut acquiring. Untuk keperluan ini kita menggunakan fungsi anggota interface IDirectInputDevice *Acquire*

```
function Acquire : HResult; stdcall;
```

Fungsi ini bisa kita panggil berulang-ulang sebanyak yang kita inginkan tanpa menyebabkan aplikasi hang.

8.6 Membaca Input.

Setelah kita memperoleh akses ke perangkat input maka sekarang kita dapat membaca input yang diberikan. Untuk membaca input digunakan fungsi anggota IDirectInputDevice *GetDeviceState*

```
function GetDeviceState(cbData : Cardinal; lpvData : Pointer) : HResult; stdcall;
```

cbData

Ukuran parameter lpvData dalam byte

lpvdata

Alamat buffer yang akan menerima status perangkat input

Jika pada saat mengatur format data kita mengatur format data mouse (dengan *c_dfDIMouse*) maka lpvData harus menunjuk ke buffer bertipe *TDIMouseState*.

Struktur TDIMouseState adalah sebagai berikut:

```
PDIMouseState = ^TDIMouseState;
TDIMouseState = packed record
  lX: Longint;
  lY: Longint;
  lZ: Longint;
  rgbButtons: Array [0..3] of Byte;
end;
```

Jika format data adalah format data untuk keyboard (*c_dfDIKeyboard*) maka lpvData menunjuk ke buffer bertipe *TDIKeyboardState* sedangkan untuk joystick (*c_dfDIJoystick*) menunjuk buffer bertipe *TDIJoyState*.

Struktur TDIKeyboardState dan TDIJoyState adalah sebagai berikut:

```
TDIKeyboardState = array[0..255] of Byte;
PDIJoyState = ^TDIJoyState;
TDIJoyState = packed record
  lX      : Longint;          (* x-axis position          *)
  lY      : Longint;          (* y-axis position          *)
  lZ      : Longint;          (* z-axis position          *)
  lRx     : Longint;          (* x-axis rotation          *)
  lRy     : Longint;          (* y-axis rotation          *)
  lRz     : Longint;          (* z-axis rotation          *)
  rglSlider  : array [0..1] of Longint;   (* extra axes positions      *)
  rgdwPOV    : array [0..3] of Cardinal;  (* POV directions            *)
  rgbButtons : array [0..31] of Byte;     (* 32 buttons               *)
end;
```

Sebenarnya masih ada struktur data yang lain yaitu TMouseState2 dan TJoyState2. Keduanya merupakan struktur data untuk mouse dengan 8 jumlah tombol dan joystick force feedback. Untuk saat ini belum kita bahas.

Jika fungsi GetDeviceState mengembalikan nilai DIERR_INPUTLOST, hal itu berarti kita telah kehilangan akses ke perangkat input sehingga kita harus melakukan proses acquire lagi sebelum membaca perangkat input.

Input bisa hilang bila aplikasi kita tidak lagi memperoleh input fokus karena user berpindah ka aplikasi yang lain.

8.7 Melepaskan Akses Perangkat Input (Unacquire).

Proses ini adalah kebalikan dari proses acquire. Fungsi ini biasanya kita panggil saat kita tidak lagi memerlukan perangkat input, misalnya saat finalisasi aplikasi.

```
function Unacquire : HResult; stdcall;
```

Meskipun proses acquire kita lakukan berkali-kali, untuk melepaskan akses perangkat input kita cukup melakukan proses unacquire sekali saja.

8.8 Finalisasi.

Untuk membebaskan memori yang telah dipakai caranya sama dengan cara membebaskan DirectDraw dan DirectSound yaitu dengan mengisikan nil ke semua objek device yang telah diciptakan , demikian pula objek DirectInput.

Bab 9
Membuat Unit uDirectInput.Pas

9.1 Pendahuluan.

Setelah kita mendapatkan cukup informasi bagaimana menggunakan DirectInput, selanjutnya kita akan meng-enkapsulasi DirectInput ini menjadi suatu objek yang ditujukan untuk menyederhanakan proses inisialisasi, membaca input dan proses finalisasi DirectInput.

9.2 Listing Unit uDirectInput.

```
unit uDirectInput;


interface
uses windows,classes,sysutils,DirectInput;
type TKeyboardBuffer=array[0..255]of byte;
    TMouseBuffer=TDIMouseState;
    TJoystickBuffer=TDIJoyState;
    TJoyStickType=record
       jstUnknown,jstTraditional,jstGamePad,jstFlightStick,
       jstWheel,jstHeadTracker,jstRudder:boolean;
     end;
    TCooperativeLevel=record
      Exclusive:boolean;
      Background:boolean;
      Foreground:boolean;
      NonExclusive:boolean;
      NoWinKey:boolean;
    end;
    TInputParam=record
      H_Instance:LongWord;
      Handle:HWND;
      CooperativeLevel:TCooperativeLevel;
```

```
  JoyStickType:TJoyStickType;  //hanya untuk joystick
end;

TDeviceItem=record
   DeviceGUID:TGUID;
   ProductName:string;
   InstanceName:string;
   Used:boolean;
 end;
PDeviceItem=^TDeviceItem;

EBaseInputError=class(Exception);
EKeyboardInputError=class(Exception);
EMouseInputError=class(Exception);
EJoystickInputError=class(Exception);
TBaseInput=class(TComponent)
   private
     MyDirectInput:IDirectInput;
     FInitSucceeded: boolean;
     FJoystickList:TList;
     FParam:TInputParam;
     function GetJoySticks(const index: integer): PDeviceItem;
     function GetJoystickCount:integer;
      Procedure EnumJoystick;
   public
      constructor Create(AOwner:TComponent;const aParam:TInputParam);reintroduce;
      destructor Destroy;override;
      procedure Free;
      property DirectInputObject:IDirectInput read MyDirectInput;
      property JoySticks[const index:integer]:PDeviceItem read GetJoySticks;
   published
     property InitSucceeded:boolean read FInitSucceeded;
     property JoystickCount:integer read GetJoystickCount;
   end;
TKeyboardInput=class(TComponent)
```

```
    private
      FBaseInput:TBaseInput;
      FKeyboardDevice:IDirectInputDevice;
      FKeyboardBuffer:TKeyboardBuffer;
    public
      constructor Create(AOwner:TComponent;const AParam:TInputParam);reintroduce;
      destructor Destroy;override;
      procedure Free;
      procedure GetDeviceState;
      function KeyDown(key:byte):boolean;
      property KeyboardBuffer:TKeyboardBuffer read FKeyboardBuffer;
  end;
 TMouseInput=class(TComponent)
    private
      FBaseInput:TBaseInput;
      FMouseDevice:IDirectInputDevice;
      FMouseBuffer:TMouseBuffer;
    public
      destructor Destroy;override;
      procedure Free;
      procedure GetDeviceState;
      constructor Create(AOwner: TComponent; const AParam: TInputParam);reIntroduce;
      Function MouseDown(button:byte):boolean;
      procedure MousePos(var x,y,z:integer);
      property MouseBuffer:TMouseBuffer read FMouseBuffer;
  end;
 TJoystickInput=class(TComponent)
    private
      FBaseInput:TBaseInput;
      FJoyStickDevice:IDirectInputDevice;
      FJoyStickDevice2:IDirectInputDevice2;
      FJoyStickBuffer:TJoystickBuffer;
      Fcaps:TDIDevCaps;
      procedure SetJoyStickBuffer(const Value: TJoyStickBuffer);
function GetTotalAxes: integer;
```

```
    function GetTotalButtons: integer;
    function GetTotalPOVs: integer;
      public
        constructor Create(AOwner:TComponent;const AParam:TInputParam);reintroduce;
        destructor Destroy;override;
        procedure Free;
        procedure GetDeviceState;
        Function JoyStickDown(button:byte):boolean;
        procedure JoyStickPos(var x,y,z:integer);
        property JoyStickBuffer:TJoyStickBuffer read FJoyStickBuffer write SetJoyStickBuffer;
      published
        property TotalButtons:integer read GetTotalButtons;
        property TotalAxes:integer read GetTotalAxes;
        property TotalPOVs:integer read GetTotalPOVs;
    end;

implementation

function JoyStickEnumCallBack(var devInstance:TDIDeviceInstance;mydata:pointer):integer;stdcall;
var item:PDeviceItem;
    DeviceList:TList;
    i:integer;
begin
  new(item);
  item.DeviceGUID:=devInstance.guidInstance;
  item.Used:=false;
  item.ProductName:='';
  for i:=low(devInstance.tszProductName) to High(devInstance.tszProductName) do
  begin
    item.ProductName:=item.ProductName+devInstance.tszProductName[i];
  end;
  item.InstanceName:='';
  for i:=low(devInstance.tszInstanceName) to High(devInstance.tszInstanceName) do
  begin
    item.InstanceName:=item.InstanceName+devInstance.tszInstanceName[i];
```

```
  end;
  DeviceList:=TList(myData);
  DeviceList.Add(item);
  result:=DIENUM_CONTINUE;
end;

function Jst2DIDev(
  const aJoystickType: TJoyStickType): cardinal;
var devtype:cardinal;
begin
  devtype:=0;
  if AJoystickType.jstUnknown then devtype:=devtype or DIDEVTYPEJOYSTICK_UNKNOWN;
  if AJoystickType.jstTraditional then devtype:=devtype or DIDEVTYPEJOYSTICK_TRADITIONAL;
  if AJoystickType.jstGamePad then devtype:=devtype or DIDEVTYPEJOYSTICK_GAMEPAD;
  if AJoystickType.jstFlightStick then devtype:=devtype or DIDEVTYPEJOYSTICK_FLIGHTSTICK;
  if AJoystickType.jstWheel then devtype:=devtype or DIDEVTYPEJOYSTICK_WHEEL;
  if AJoystickType.jstHeadTracker then devtype:=devtype or DIDEVTYPEJOYSTICK_HEADTRACKER;
  if AJoystickType.jstRudder then devtype:=devtype or DIDEVTYPEJOYSTICK_RUDDER;
  result:=devtype;
end;

{ TKeyboardInput }

constructor TKeyboardInput.Create(AOwner: TComponent;const AParam:TInputParam);
var hr:HResult;
    dataformat:TDIDataFormat;
    flags:cardinal;
begin
 try
  inherited Create(Aowner);
  FKeyboardDevice:=nil;
  FBaseInput:=TBaseInput(AOwner);
  hr:=FBaseInput.MyDirectInput.CreateDevice(GUID_SysKeyboard,FKeyboardDevice,nil);
  if failed(hr) then
    raise EKeyboardInputError.Create('Unable to create keyboard device.');
```

```
  dataformat:=c_dfDIKeyboard;
  hr:=FKeyboardDevice.SetDataFormat(dataformat);
  if failed(hr) then
    raise EKeyboardInputError.Create('Unable to set keyboard data format.');

  flags:=0;
  if (Aparam.CooperativeLevel.Exclusive) and
     (Aparam.CooperativeLevel.NonExclusive) then
    raise EKeyboardInputError.Create('Invalid cooperative level combination EXCLUSIVE and
NONEXCLUSIVE.');

  if Aparam.CooperativeLevel.Exclusive then flags:=DISCL_EXCLUSIVE;
  if Aparam.CooperativeLevel.NonExclusive then flags:=flags or DISCL_NONEXCLUSIVE;

  if (Aparam.CooperativeLevel.Background) and
     (Aparam.CooperativeLevel.Foreground) then
    raise EKeyboardInputError.Create('Invalid cooperative level combination BACKGROUND and
FOREGROUND.');

  if Aparam.CooperativeLevel.Background then flags:=flags or DISCL_BACKGROUND;
  if Aparam.CooperativeLevel.Foreground then flags:=flags or DISCL_FOREGROUND;
  if Aparam.CooperativeLevel.NoWinKey then flags:=flags or DISCL_NOWINKEY;
  hr:=FKeyboardDevice.SetCooperativeLevel(AParam.Handle,flags);
  hr:=FKeyboardDevice.Acquire;
 except
  on EKeyboardInputError do
  begin
    FKeyboardDevice:=nil;
  end;
 end;
end;

destructor TKeyboardInput.Destroy;
var hr:HResult;
begin
  hr:=FKeyboardDevice.Unacquire;
```

```
  if failed(hr) then
    raise EKeyboardInputError.Create('Unable to unacquire keyboard device.');
  FKeyboardDevice:=nil;
  inherited;
end;

procedure TKeyboardInput.Free;
begin
  if self<>nil then destroy;
end;

procedure TKeyboardInput.GetDeviceState;
var hr:HResult;
begin
  hr:=FKeyboardDevice.GetDeviceState(sizeof(TKeyboardBuffer),@FKeyboardBuffer);
  if (hr=DIERR_INPUTLOST) or (hr=DIERR_NOTACQUIRED) then
  begin
    FKeyboardDevice.Acquire;
    hr:=FKeyboardDevice.GetDeviceState(sizeof(TKeyboardBuffer),@FKeyboardBuffer);
  end;
end;

function TKeyboardInput.KeyDown(key: byte): boolean;
begin
  Result:=(FKeyboardBuffer[key] and $80)=$80;
end;

{ TBaseInput }

constructor TBaseInput.Create(AOwner: TComponent;const aParam:TInputParam);
var hr:HResult;
begin
 try
  inherited Create(AOwner);
  if assigned(MyDirectInput) then
```

```
  MyDirectInput:=nil;
  FInitSucceeded:=false;
  FParam:=aParam;
  hr:=DirectInputCreate(AParam.H_Instance,DIRECTINPUT_VERSION,MyDirectInput,nil);
  if succeeded(hr) then
  begin
    FInitSucceeded:=true;
  end else
    Raise EBaseInputError.Create('Unable to create DirectInput Object-'+DIErrorString(hr));
  FJoystickList:=TList.Create;
  EnumJoystick;
 except
 end;
end;

destructor TBaseInput.Destroy;
var i:integer;
    item:PDeviceItem;
begin
  MyDirectInput:=nil;
  for i:=FJoystickList.Count-1 downto 0 do
  begin
    item:=FJoystickList.Items[i];
    dispose(item);
    FJoystickList.Delete(i);
  end;
  FJoystickList.Free;
  inherited;
end;

procedure TBaseInput.EnumJoystick;
var devInstance:TDIDeviceInstance;
    hr:HResult;
begin
  fillChar(devInstance,sizeOf(TDIDeviceInstance),0);
```

```
  devInstance.dwSize:=sizeOf(TDIDeviceInstance);
  hr:=MyDirectInput.EnumDevices(Jst2DIDev(FParam.JoystickType),JoyStickENumCallBack,
Pointer(FJoystickList),DIEDFL_ATTACHEDONLY);
  if Failed(hr) then raise EBaseInputError.Create('Unable to enum joystick.');
end;

procedure TBaseInput.Free;
begin
  if self<>nil then destroy;
end;

function TBaseInput.GetJoystickCount: integer;
begin
  result:=FJoystickList.Count;
end;

function TBaseInput.GetJoySticks(const index: integer): PDeviceItem;
begin
  result:=FJoystickList.Items[index];
end;

{ TMouseInput }

constructor TMouseInput.Create(AOwner: TComponent;const AParam:TInputParam);
var hr:HResult;
    dataformat:TDIDataFormat;
    flags:cardinal;
begin
 try
  inherited Create(Aowner);
  FMouseDevice:=nil;
  FBaseInput:=TBaseInput(AOwner);
  hr:=FBaseInput.MyDirectInput.CreateDevice(GUID_SysMouse,FMouseDevice,nil);
  if failed(hr) then
    raise EMouseInputError.Create('Unable to create mouse device.');
  dataformat:=c_dfDIMouse;
```

```
  hr:=FMouseDevice.SetDataFormat(dataformat);
  if failed(hr) then
    raise EMouseInputError.Create('Unable to set mouse data format.');

  flags:=0;
  if (Aparam.CooperativeLevel.Exclusive) and
     (Aparam.CooperativeLevel.NonExclusive) then
    raise EMouseInputError.Create('Invalid cooperative level combination EXCLUSIVE and
NONEXCLUSIVE.');

  if Aparam.CooperativeLevel.Exclusive then flags:=DISCL_EXCLUSIVE;
  if Aparam.CooperativeLevel.NonExclusive then flags:=flags or DISCL_NONEXCLUSIVE;

  if (Aparam.CooperativeLevel.Background) and
     (Aparam.CooperativeLevel.Foreground) then
    raise EMouseInputError.Create('Invalid cooperative level combination BACKGROUND and
FOREGROUND.');

  if Aparam.CooperativeLevel.Background then flags:=flags or DISCL_BACKGROUND;
  if Aparam.CooperativeLevel.Foreground then flags:=flags or DISCL_FOREGROUND;
  if Aparam.CooperativeLevel.NoWinKey then flags:=flags or DISCL_NOWINKEY;
  hr:=FMouseDevice.SetCooperativeLevel(AParam.Handle,flags);
  hr:=FMouseDevice.Acquire;
 except
  on EMouseInputError do
  begin
    FMouseDevice:=nil;
  end;
 end;
end;

destructor TMouseInput.Destroy;
var hr:HResult;
begin
  hr:=FMouseDevice.Unacquire;
  if failed(hr) then
```

```
    raise EKeyboardInputError.Create('Unable to unacquire mouse device.');
  FMouseDevice:=nil;
  inherited;
end;

procedure TMouseInput.Free;
begin
  if self<>nil then destroy;
end;

procedure TMouseInput.GetDeviceState;
var hr:HResult;
begin
  hr:=FMouseDevice.GetDeviceState(sizeof(TMouseBuffer),@FMouseBuffer);
  if (hr=DIERR_INPUTLOST) or (hr=DIERR_NOTACQUIRED) then
  begin
    FMouseDevice.Acquire;
    hr:=FMouseDevice.GetDeviceState(sizeof(TMouseBuffer),@FMouseBuffer);
  end;
end;

function TMouseInput.MouseDown(button: byte): boolean;
begin
  Result:=((FMouseBuffer.rgbButtons[button] and $80)=$80);
end;

procedure TMouseInput.MousePos(var x, y,z: integer);
begin
  x:=FMouseBuffer.lX;
  y:=FMouseBuffer.lY;
  z:=FMouseBuffer.lZ;
end;
{ TJoystickInput }

constructor TJoystickInput.Create(AOwner: TComponent;const AParam:TInputParam);
```

```
var hr:HResult;
    dataformat:TDIDataFormat;
    flags:cardinal;
    item:PDeviceItem;
    i:integer;
begin
 try
  inherited Create(Aowner);
  FJoyStickDevice:=nil;
  FBaseInput:=TBaseInput(AOwner);
  if FBaseInput.JoystickCount<>0 then
  begin
    for i:=0 to FBaseInput.JoystickCount-1 do
    begin
      item:=FBaseInput.JoySticks[i];
      if not item.Used then
      begin
        FBaseInput.MyDirectInput.CreateDevice(item.DeviceGUID,FJoyStickDevice,nil);
        hr:=FJoystickDevice.QueryInterface(IID_IDirectInputDevice2,FJoystickDevice2);
        if failed(hr) then
          raise EJoyStickInputError.Create('Unable to create joystick device.');
        FJoystickDevice:=nil;
        dataformat:=c_dfDIJoyStick;
        hr:=FJoystickDevice2.SetDataFormat(dataformat);
        if failed(hr) then
          raise EJoystickInputError.Create('Unable to set joystick data format.');
        flags:=0;
        if (Aparam.CooperativeLevel.Exclusive) and
           (Aparam.CooperativeLevel.NonExclusive) then
        raise EJoyStickInputError.Create('Invalid cooperative level combination EXCLUSIVE and
NONEXCLUSIVE.');

        if Aparam.CooperativeLevel.Exclusive then flags:=DISCL_EXCLUSIVE;
        if Aparam.CooperativeLevel.NonExclusive then flags:=flags or DISCL_NONEXCLUSIVE;

        if (Aparam.CooperativeLevel.Background) and
```

```
        (Aparam.CooperativeLevel.Foreground) then
       raise EJoyStickInputError.Create('Invalid cooperative level combination BACKGROUND and
FOREGROUND.');

      if Aparam.CooperativeLevel.Background then flags:=flags or DISCL_BACKGROUND;
      if Aparam.CooperativeLevel.Foreground then flags:=flags or DISCL_FOREGROUND;
      if Aparam.CooperativeLevel.NoWinKey then flags:=flags or DISCL_NOWINKEY;
      hr:=FJoystickDevice2.SetCooperativeLevel(AParam.Handle,flags);

      FCaps.dwSize:=sizeof(TDIDevCaps);
      FJoystickDevice2.GetCapabilities(FCaps);
      hr:=FJoystickDevice2.Acquire;
      item.Used:=true;
      break;
    end;
  end;
 end else raise EJoystickInputError.Create('No joystick attached.');
 except
  on EJoyStickInputError do
  begin
   FJoystickDevice2:=nil;
  end;
 end;
end;

destructor TJoystickInput.Destroy;
var hr:HResult;
begin
 hr:=FJoystickDevice2.Unacquire;
 if failed(hr) then
   raise EJoyStickInputError.Create('Unable to unacquire joystick device.');
 FJoystickDevice2:=nil;
 inherited;
end;

procedure TJoystickInput.Free;
```

```
begin
  if self<>nil then destroy;
end;

procedure TJoystickInput.GetDeviceState;
var hr:HResult;
begin
  FJoystickDevice2.Poll;
  hr:=FJoystickDevice2.GetDeviceState(SizeOF(TJoyStickBuffer),@FJoyStickBuffer);
  if (hr=DIERR_INPUTLOST) or (hr=DIERR_NOTACQUIRED) then
  begin
    FJoystickDevice2.Acquire;
    FJoystickDevice2.Poll;
    FJoystickDevice2.GetDeviceState(SizeOF(TJoyStickBuffer),@FJoyStickBuffer);
  end;
end;

function TJoystickInput.GetTotalAxes: integer;
begin
  result:=FCaps.dwAxes;
end;

function TJoystickInput.GetTotalButtons: integer;
begin
  result:=FCaps.dwButtons;
end;

function TJoystickInput.GetTotalPOVs: integer;
begin
  result:=FCaps.dwPOVs;
end;

function TJoystickInput.JoyStickDown(button: byte): boolean;
begin
  Result:=((FJoystickBuffer.rgbButtons[button] and $80)=$80);
```

```
end;

procedure TJoystickInput.JoyStickPos(var x, y, z: integer);
begin
  x:=FJoyStickBuffer.lX;
  y:=FJoyStickBuffer.lY;
  z:=FJoyStickBuffer.lZ;
end;

procedure TJoystickInput.SetJoyStickBuffer(const Value: TJoyStickBuffer);
begin
  FJoyStickBuffer := Value;
end;
end.
```

9.3 Kelas TBaseInput.

9.3.1 Properti.

Properti yang dimiliki oleh kelas ini adalah

- *DirectInputObject* bertipe IDirectInput menunjuk ke objek DirectInput.
- *InitSucceeded* bertipe Boolean mencatat status keberhasilan proses inisialisasi.
- *Joysticks* bertipe array PDeviceItem. PdeviceItem adalah pointer ke tipe TdeviceItem yang menyimpan informasi device joystick seperti GUID, nama produk dan nama instance.
- *JoystickCount* bertipe integer menyimpan informasi jumlah joysyick yang tercatat dalam daftar joystick.

9.3.2 Inisialisasi.

Kelas TBaseInput adalah kelas dasar yang digunakan untuk menciptakan kelas-kelas lain seperti TMouseInput, TKeyboardInput dan TJoyStickInput. Kelas TBaseInput harus diciptakan terlebih dahulu sebelum menciptakan kelas lainnya. Proses inisialisasinya dikerjakan pada konstruktor Create yaitu dengan menciptakan objek MyDirectInput.

Parameter konstruktor ini adalah *AOwner* bertipe TComponent dan *InputParam* bertipe TInputParam. Input param ini berisi data handle instance dan handle window aplikasi serta level kooperatif yang diinginkan. Selain itu kita

menciptakan suatu list yang akan menyimpan informasi daftar GUID tiap joystick yang terpasang dan melakukan proses enumerasi untuk mencatat semua joystick yang terpasang dengan memanggil rutin *EnumJoystick*. Metode ini hanya mencatat joystick yang terpasang namun tidak menciptakan device joystick itu sendiri.

Pembaca dapat melihat source code inisialisasi TBaseInput pada sub bab di atas. Penulis sengaja tidak menulis ulang source codenya.

9.3.3 Melakukan Proses Enumerasi Joystick.

Proses enumerasi dikerjakan oleh EnumJoystick yang bersifat privat Implementasinya adalah sebagai berikut:

```
procedure TBaseInput.EnumJoystick;

var devInstance:TDIDeviceInstance;

   hr:HResult;

begin

  fillChar(devInstance,sizeOf(TDIDeviceInstance),0);

  devInstance.dwSize:=sizeOf(TDIDeviceInstance);
```

Kita inisialisasi variabel *devInstance* dengan nol dan mengisi field *dwSize* dengan ukuran struktur TDIDevInstance. Variabel devInstance akan kita gunakan untuk melakukan proses enumerasi. Tiap kali sebuah joystick ditemukan, informasi mengenai joystick ini akan disimpan di variabel ini.

```
hr:=MyDirectInput.EnumDevices(Jst2DIDev(FParam.JoystickType),JoyStickENumCallBack,
Pointer(FJoystickList),DIEDFL_ATTACHEDONLY);
```

Proses enumerisasi joystick yang terpasang pada komputer, membutuhkan dua fungsi tambahan yaitu fungsi callback *JoystickEnumCallBack* dan *Jst2DIDev*. Fungsi yang pertama adalah fungsi yang akan dipanggil oleh EnumDevice tiap kali EnumDevice menemukan joystick yang terpasang pada komputer. Pada fungsi ini kita akan mencatat semua joystick yang ditemukan kedalam *FJoystickList*. Informasi ini meliputi GUID instance joystick, nama produk, dan nama instance. Field yang paling penting adalah GUID instance joystick. Selanjutnya proses enumerisasi kita lanjutkan dengan mengembalikan nilai fungsi sama dengan DIENUM_CONTINUE untuk mencari joystick lain.

Fungsi Jst2DIDev berguna untuk mengkonversi format tipe joystick dari TjoystickType menjadi nilai cardinal.

Pembaca bisa mempelajari implementasi fungsi-fungsi ini yang penulis rasa tidak terlalu sulit.

```
  if Failed(hr) then raise EBaseInputError.Create('Unable to enum joystick.');

end;
```

9.3.4 Finalisasi.

Finalisasi dikerjakan oleh destruktor Destroy atau Free.

9.4 Kelas TKeyboardInput.

9.4.1 Properti.

Kelas ini menangani input dari keyboard. Untuk mengerjakan fungsinya tersebut kita perlu menambahkan properti *KeyboardBuffer* yang bertipe *TKeyboardBuffer*. Tipe ini dideklarasikan di unit uDirectInput.pas berupa array bertipe byte berukuran 256. KeyboardBuffer ini mencatat status penekanan tombol-tombol keyboard saat ini. Property ini kita buat read-only.

9.4.2 Inisialisasi.

Inisialisasi dikerjakan oleh konstruktor *Create*. Parameternya adalah komponen yang menjadi owner instance ini serta InputParam bertipe TInputParam. Owner instance ini adalah instance TBaseInput karena TKeyboardInput membutuhkan alamat IDirectInput.

Proses yang dikerjakan di konstruktor ini, secara garis besar terdiri atas proses menciptakan device, mengatur format data, mengatur level kooperatif dan terakhir memanggil Acquire untuk mendapatkan akses ke keyboard.

Kita juga perlu mengecek level kooperatif yang diinginkan sebelum memanggil fungsi SetCooperativeLevel, karena ada beberapa level kooperatif yang tidak boleh digunakan bersama-sama.

9.4.3 Mengambil Status Keyboard.

Untuk mengambil status keyboard kita tambahkan metode *GetDeviceState*. Prosedur ini tidak membutuhkan parameter apa-apa. Prosedur ini adalah rutin yang akan sering kita panggil dalam looping. Oleh karena itu harus dibuat seringkas dan secepat mungkin.

Proses yang dikerjakan adalah memanggil fungsi milik FKeyboardDevice yaitu GetDeviceState. Keluaran fungsi ini disimpan di FKeyboardBuffer bertipe TKeyboardBuffer.

Karena fungsi GetDeviceState ini mungkin gagal saat mengambil status keyboard, maka kita perlu mengecek nilai kembali fungsi ini. Jika input hilang atau kita belum memanggil fungsi Acquire untuk mendapatkan akses keyboard, maka kita panggil Acquire sekali lagi dan selanjutnya mengambil status keyboard.

9.4.4 Mengecek Penekanan Tombol.

Fungsi *KeyDown* kita perlukan untuk mengerjakan proses pengecekan ini. Parameter fungsi ini adalah scan code tombol keyboard yang akan kita cek

statusnya. Nilai yang dikembalikan fungsi ini adalah true jika tombol sedang ditekan dan false bila tidak.

Untuk mengecek status suatu tombol apakah sedang ditekan atau tidak caranya adalah dengan membaca status yang tersimpan dalam FKeyboardBuffer. Scan code tombol yang akan kita cek kita gunakan sebagai indeks ke FKeyboardBuffer. Kita lakukan operasi *and* pada byte yang ditunjuk oleh scan code dengan 80h. Bit paling signifikan akan berisi 1 bila tombol sedang ditekan dan 0 bila tidak. Jika hasil operasi *and* sama dengan 80h maka ini mengindikasikan bahwa tombol sedang ditekan.

9.4.5 Finalisasi.

Finalisasi dikerjakan untuk membebaskan akses ke keyboard serta membebaskan memori yang dipergunakan oleh FKeyboardDevice. Seperti biasanya destruktor Destroy bertanggung jawab atas proses ini.

9.5 Kelas TMouseInput.

9.5.1 Properti.

Properti yang dimiliki oleh kelas ini adalah MouseBuffer bertipe TMouseBuffer. Properti ini bersifat read-only, seperti pada kelas TKeyboardInput, isi buffer ini akan di-update saat pemanggilan metode GetDeviceState. Tipe TMouseBuffer sendiri sama dengan tipe TDIMouseState berupa sebuah record berisi data posisi perubahan posisi x, y, z (disimpan di field *lX*, *lY*, *lZ*) dan informasi status tombol (field *rgbButtons*).

9.5.2 Inisialisasi.

Inisialisasi ditangani oleh konstruktor Create. Prosesnya hampir sama dengan Create milik TKeyboardInput. Perbedaannya terletak pada penggunaan konstanta GUID mouse system (*GUID_SysMouse*) ketika menciptakan device mouse, serta konstanta *c_dfDIMouse* saat mengatur format data yang ingin kita pergunakan.

9.5.3 Mengambil Status Mouse.

Untuk mengambil status mouse, kelas ini kita lengkapi dengan metode *GetDeviceState*. Prosedur ini tidak membutuhkan parameter apa-apa. Prosedur ini adalah rutin yang akan sering kita panggil dalam looping. Oleh karena itu harus dibuat seringkas dan secepat mungkin.

Proses yang dikerjakan adalah memanggil fungsi milik FMouseDevice yaitu GetDeviceState. Keluaran fungsi ini disimpan di FMouseBuffer bertipe TMouseBuffer.

Karena fungsi GetDeviceState ini mungkin gagal saat mengambil status keyboard, maka kita perlu mengecek nilai kembali fungsi ini. Jika input hilang atau kita belum memanggil fungsi Acquire untuk mendapatkan akses mouse, maka kita panggil Acquire sekali lagi dan selanjutnya mengambil status mouse.

9.5.4 Mengecek Penekanan Tombol .

Fungsi *MouseDown* kita perlukan untuk mengerjakan proses pengecekan ini. Parameter fungsi ini adalah tombol mouse yang akan kita cek statusnya. Tombol mouse kiri kodenya adalah 0, tombol mouse kanan berkode 1. Nilai yang dikembalikan fungsi ini adalah true jika tombol sedang ditekan dan false bila tidak.

Untuk mengecek status suatu tombol apakah sedang ditekan atau tidak caranya adalah dengan membaca status yang tersimpan dalam FMouseBuffer. Kode tombol yang akan kita cek kita gunakan sebagai indeks ke field rgbButtons milik FMouseBuffer. Kita lakukan operasi *and* pada byte yang ditunjuk oleh kode tombol dengan 80h. Bit paling signifikan akan berisi 1 bila tombol sedang ditekan dan 0 bila tidak. Jika hasil operasi *and* sama dengan 80h maka ini mengindikasikan bahwa tombol sedang ditekan.

9.5.5 Mengambil Informasi Posisi Mouse.

Posisi mouse saat ini dapat kita ketahui dengan membaca field lX, lY, lZ milik FMouseBuffer. Untuk saat ini kita hanya membutuhkan posisi x dan y sehingga field lZ belum kita butuhkan.

Metode yang bertanggung jawab akan proses ini adalah *MousePos*. Prosedur ini akan membaca nilai lX, lY dan lZ dan mengembalikannya ke parameter input fungsi ini (x, y, z). Oleh karena itu parameter ini dideklarasikan dengan menambahkan kata tercadang *var*.

Nilai yang dikembalikan ke x, y, dan z adalah perubahan posisi relatif terhadap posisi sebelumnya. Oleh karena itu posisi ini bukanlah nilai koordinat pada layar. Untuk mendapatkan posisi kursor pada layar, koordinat kursor mouse sebelumnya harus ditambahkan dengan posisi saat ini.

9.5.6 Finalisasi.

Finalisasi dikerjakan untuk membebaskan akses ke mouse serta membebaskan memori yang dipergunakan oleh FMouseDevice. Seperti biasanya destruktor Destroy bertanggung jawab atas proses ini.

9.6 Kelas TJoyStickInput.

9.6.1 Properti.

Properti yang dimiliki oleh kelas ini adalah JoyStickBuffer bertipe TJoyStickBuffer. Properti ini bersifat read-only, seperti pada kelas-kelas sebelumnya, isi buffer ini akan di-update saat pemanggilan metode GetDeviceState. Tipe TJoyStickBuffer sendiri sama dengan tipe TDIJoyState berupa sebuah record berisi data posisi perubahan posisi x, y, z (disimpan di field *lX, lY, lZ*) dan informasi status tombol (field *rgbButtons*). Ada beberapa field lain pada record TjoyStickBuffer yaitu: *lRX, lRY, lRZ, rglSlider, rgdwPOV*. Namun untuk game 2D field-field ini belum begitu diperlukan. Oleh karena itu implementasi rutin untuk membaca field-field tersebut belum kita buat.

9.6.2 Inisialisasi.

Inisialisasi ditangani oleh konstruktor Create. Prosesnya hampir sama dengan Create milik kelas-kelas sebelumnya. Perbedaannya terletak pada proses untuk mendapatkan GUID joystick yang terpasang pada komputer ketika menciptakan device joystick, serta konstanta *c_dfDIJoystick* saat mengatur format data yang ingin kita pergunakan. GUID joystick kita ambil dari property *Joysticks* milik instance kelas BaseInput.

9.6.3 Mengambil Status JoyStick.

Untuk mengambil status joystick, caranya sama dengan kelas-kelas sebelumnya yaitu dengan menggunakan *GetDeviceState*. Fungsi ini akan mengupdate isi FJoystickBuffer tiap kali fungsi ini dipanggil.

9.6.4 Mengecek Penekanan Tombol.

Fungsi *JoystickDown* kita perlukan untuk mengerjakan proses pengecekan ini. Parameter fungsi ini adalah tombol joystick yang akan kita cek statusnya.

Untuk mengecek status suatu tombol apakah sedang ditekan atau tidak caranya adalah dengan membaca status yang tersimpan dalam FJoystickBuffer. Kode tombol yang akan kita cek kita gunakan sebagai indeks ke field rgbButtons milik FJoystickBuffer. Kita lakukan operasi *and* pada byte yang ditunjuk oleh kode tombol dengan 80h. Bit paling signifikan akan berisi 1 bila tombol sedang ditekan dan 0 bila tidak. Jika hasil operasi *and* sama dengan 80h maka ini mengindikasikan bahwa tombol sedang ditekan.

9.6.5 Mengambil Informasi Posisi JoyStick.

Posisi joystick saat ini dapat kita ketahui dengan membaca field lX, lY, lZ milik FJoystickBuffer. Untuk saat ini kita hanya membutuhkan posisi x dan y sehingga field lZ belum kita butuhkan.

Metode yang bertanggung jawab akan proses ini adalah *JoyStickPos*. Prosedur ini akan membaca nilai lX, lY dan lZ dan mengembalikannya ke parameter

input fungsi ini (x, y, z). Oleh karena itu parameter ini dideklarasikan dengan menambahkan kata tercadang *var*.

Seperti pada mouse, nilai yang dikembalikan ke x, y, dan z adalah perubahan posisi relatif terhadap posisi sebelumnya. Oleh karena itu posisi ini bukanlah nilai koordinat pada layar. Untuk mendapatkan posisi joystick pada layar, koordinat joystick sebelumnya harus ditambahkan dengan posisi saat ini.

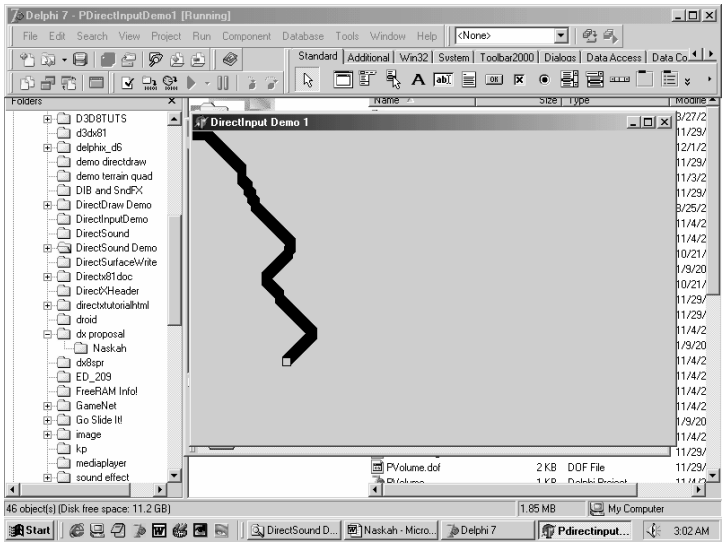
9.6.6 Finalisasi.

Finalisasi dikerjakan untuk membebaskan akses ke joystick serta membebaskan memori yang dipergunakan oleh FJoystickDevice. Proses ini dikerjakan oleh destruktor Destroy. Untuk finalisasi yang aman kita gunakan Free.

Bab 10
Demo Program 3
Menambahkan Input

10.1 Listing Program 3.1 Mengakses Keyboard dengan DirectInput.

Listing program berikut ini dapat anda temukan di direktori dimana anda menginstall CD-ROM pada folder Pemrograman Game Dengan DirectX\Bab 10\ Demo 1\ dengan nama file PDirectInputDemo1.dpr.



Gambar 10.1 Mengakses keyboard dengan DirectInput.

```
unit ufrmDirectInputDemo1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,DirectInput,uDirectInput, ExtCtrls;

type
```

```
 TForm1 = class(TForm)
   Timer1: TTimer;
   procedure FormCreate(Sender: TObject);
   procedure FormDestroy(Sender: TObject);
   procedure Timer1Timer(Sender: TObject);
 private
   MyBaseInput:TBaseInput;
   MyKeyboard:TKeyboardInput;
   posx,posY:integer;
   { Private declarations }
 public

   { Public declarations }
 end;

var
 Form1: TForm1;

implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
var param:TInputParam;
begin
 try
  fillchar(param,sizeOf(TInputParam),0);
  param.Handle:=Handle;
  param.H_Instance:=hInstance;
  param.CooperativeLevel.Exclusive:=true;
  param.CooperativeLevel.Background:=true;
  MyBaseInput:=nil;
  MyKeyboard:=nil;
  MyBaseInput:=TBaseInput.Create(self,param);
  MyKeyboard:=TKeyboardInput.Create(MyBaseInput,param);
  timer1.Enabled:=true;
  posX:=0;
```

```
  posY:=0;
 except
  close;
 end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  MyKeyboard.Free;
  MyBaseInput.Free;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  MyKeyboard.GetDeviceState;
  if MyKeyboard.KeyDown(DIK_ESCAPE) then Close;
  if MyKeyboard.KeyDown(DIK_UP) then
    dec(posY);
  if MyKeyboard.KeyDown(DIK_DOWN) then
    inc(posy);
  if MyKeyboard.KeyDown(DIK_LEFT) then
    dec(posx);
  if MyKeyboard.KeyDown(DIK_RIGHT) then
    inc(posx);
  canvas.Rectangle(posx,posy,posx+10,posy+10);
end;
end.
```

10.2 Penjelasan Program 3.1.

10.2.1 Inisialisasi.

Rutin inisialisasi dikerjakan oleh prosedur *FormCreate*. Pada prosedur ini kita menciptakan instance-instance kelas TBaseInput dan TKeyboardInput serta menginisialisasi variable-variabel yang akan kita pergunakan.

```
procedure TForm1.FormCreate(Sender: TObject);
var param:TInputParam;
```

```
begin
 try
  fillchar(param,sizeOf(TInputParam),0);
```

Kita isi variable sementara param dengan nol.

```
  param.Handle:=Handle;
  param.H_Instance:=hInstance;
  param.CooperativeLevel.Exclusive:=true;
  param.CooperativeLevel.Background:=true;
```

Inisialisasi variable param dengan handle form utama aplikasi serta handle instance aplikasi. Level kooperatif yang kita inginkan adalah eksklusif dan akses background.

```
  MyBaseInput:=nil;
  MyKeyboard:=nil;
  MyBaseInput:=TBaseInput.Create(self,param);
  MyKeyboard:=TKeyboardInput.Create(MyBaseInput,param);
```

Kita ciptakan instance kelas TBaseInput dan TKeyboardInput.

```
  timer1.Enabled:=true;
  posX:=0;
  posY:=0;
```

Posisi yang mencatat koordinat kursor kita isi dengan nol.

```
 except
  close;
```

Jika ada error maka kita tutup aplikasi.

```
 end;
end;
```

10.2.2 Event OnTimer.

Rutin pengecekan dilakukan pada saat event timer dibangkitkan. Pada saat event ini terjadi maka kita baca status perangkat input dan mengerjakan proses yang harus dilakukan bila ada tombol keyboard yang sedang ditekan.

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  MyKeyboard.GetDeviceState;
```

Ambil status keyboard.

```
  if MyKeyboard.KeyDown(DIK_ESCAPE) then Close;
```

Cek apakah yang sedang ditekan adalah tombol ESC. Jika ya maka tutup aplikasi.

```
if MyKeyboard.KeyDown(DIK_UP) then
  dec(posY);
```

Cek apakah yang sedang ditekan adalah tombol panah atas. Jika ya maka kurangi koordinat y.

```
if MyKeyboard.KeyDown(DIK_DOWN) then
  inc(posy);
```

Cek apakah yang sedang ditekan adalah tombol panah bawah. Jika ya maka tambahkan koordinat y.

```
if MyKeyboard.KeyDown(DIK_LEFT) then
  dec(posx);
```

Cek apakah yang sedang ditekan adalah tombol panah kiri. Jika ya maka kurangi koordinat x.

```
if MyKeyboard.KeyDown(DIK_RIGHT) then
  inc(posx);
```

Cek apakah yang sedang ditekan adalah tombol panah kanan. Jika ya maka tambahkan koordinat x.

```
canvas.Rectangle(posx,posy,posx+10,posy+10);
```

Gambar persegi empat pada koordinat posx,posy dengan lebar 10x10.

```
end;
```

Dengan model pengecekan kondisi seperti ini, maka bila ada dua tombol yang ditekan bersamaan, sebagai contoh panah atas dan panah kanan, gambar persegi empat akan terlihat bergerak diagonal ke arah kanan atas. Hal ini tidak dapat kita lakukan bila menggunakan event OnKeyDown milik form. Silahkan pembaca membandingkannya dengan program PNoDirectInput pada direktori DirectInput\Exe\PnoDirectInput.exe.

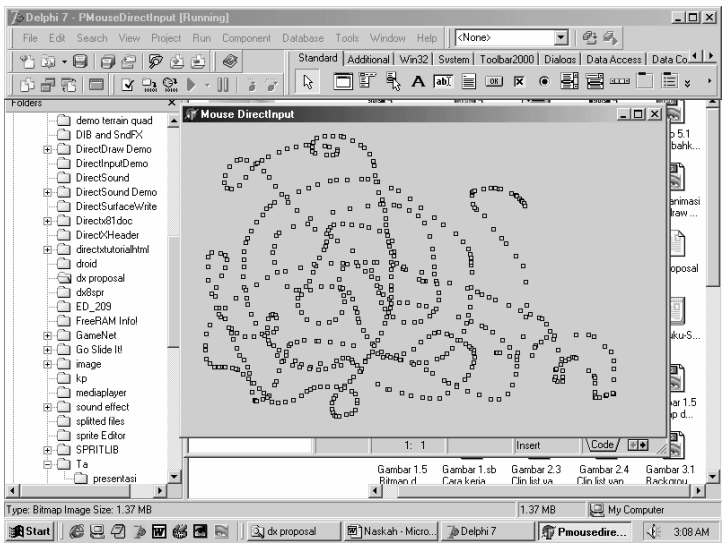
10.2.3 Finalisasi.

Seperti biasanya proses finalisasi dikerjakan untuk membebaskan memori yang telah dipergunakan oleh instance MyKeyboard dan MyBaseInput.

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  MyKeyboard.Free;
  MyBaseInput.Free;
end;
```

10.3 Listing Program 3.2 Mengakses Mouse dengan DirectInput.

Listing program ini dapat anda temukan di direktori Pemrograman Game Dengan DirectX\Bab 10\Demo 2 dengan nama projectnya PMouseDirectInput.dpr.



Gambar 10.2 Mengakses mouse dengan DirectInput.

```
unit ufrmMouseDirectInput;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,DirectInput,UDirectInput, ExtCtrls;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    BaseInput:TBaseInput;
```

```
    MouseInput:TMouseInput;
    posx,posY:integer;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
var param:TInputParam;
begin
  fillchar(param,sizeof(TInputParam),0);
  param.H_Instance:=HInstance;
  param.Handle:=Handle;
  param.CooperativeLevel.Exclusive:=true;
  param.CooperativeLevel.Foreground:=true;
  BaseInput:=TBaseInput.Create(self,param);
  MouseInput:=TMouseInput.Create(BaseInput,Param);
  posx:=clientwidth div 2;
  posy:=clientheight div 2;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  MouseInput.Free;
  BaseInput.Free;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
```

```
var x,y,z:integer;
begin
  MouseInput.GetDeviceState;
  MouseInput.MousePos(x,y,z);
  inc(posx,x);
  inc(posy,y);
  if MouseInput.MouseDown(0) then
    canvas.TextOut(posx,posy,'Left Mouse');
  if MouseInput.MouseDown(1) then
    canvas.TextOut(posx,posy,'Right Mouse');
  canvas.Rectangle(posx,posy,posx+5,posy+5);
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
   VK_ESCAPE:close;
  end;
end;
end.
```

10.4 Penjelasan Program 3.2.

10.4.1 Inisialisasi.

Inisialisasi dilakukan oleh FormCreate. Proses yang dikerjakan pada rutin ini hampir sama dengan proses inisialisasi yang dikerjakan di contoh program sebelumnya. Perbedaannya terletak pada tiga baris terakhir dimana kita menciptakan instance kelas TMouseInput dan mengisi nilai awal posisi koordinat kursor posx dan posy agar berada di tengah-tengah form.

10.4.2 Event OnTimer.

Rutin pengecekan korrdinat mouse dan tombol yang sedang ditekan dikerjakan saat event timer dibangkitkan.

```
procedure TForm1.Timer1Timer(Sender: TObject);
var x,y,z:integer;
```

```
begin
  MouseInput.GetDeviceState;
```

Ambil status perangkat mouse.

```
  MouseInput.MousePos(x,y,z);
```

Ambil koordinat mouse. Yang perlu menjadi catatan, nilai yang tersimpan di x, y, z bukan nilai yang mengacu pada koordinat (0,0) melainkan perpindahan posisi x, y, z kursor relatif terhadap posisi kursor sebelumnya. Oleh karena itu pada baris berikut koordinat mouse perlu kita perbarui nilainya dengan menambahkannya dengan nilai x, y (nilai z kita abaikan, karena kita tidak memerlukannya mengingat aplikasi kita adalah aplikasi 2D).

```
  inc(posx,x);
  inc(posy,y);
  if MouseInput.MouseDown(0) then
    canvas.TextOut(posx,posy,'Left Mouse');
```

Cek apakah tombol yang ditekan adalah tombol kiri. Jika ya, tulis string "Left Mouse".

```
  if MouseInput.MouseDown(1) then
    canvas.TextOut(posx,posy,'Right Mouse');
```

Cek apakah tombol yang ditekan adalah tombol kanan. Jika ya, tulis string "Right Mouse".

```
  canvas.Rectangle(posx,posy,posx+5,posy+5);
```

Gambar kursor aplikasi kita pada koordinat posx, posy. Catatan tambahan: jika pembaca menggunakan DirectInput untuk mengakses mouse, kita harus menggambar kursor mouse kita sendiri. DirectInput tidak menyediakan fungsi-fungsi untuk menggambar kursor.

```
end;
```

10.4.3 Event OnKeyDown.

Handler event OnKeyDown milik form kita tambahkan untuk memproses jika tombol ESC ditekan. Penekanan tombol ini kita gunakan sebagai salah satu cara mengakhiri aplikasi.

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
   VK_ESCAPE:close;
  end;
end;
```

10.4.4 Finalisasi.

Finalisasi kita lakukan dengan membebaskan instance MouseInput dan BaseInput.

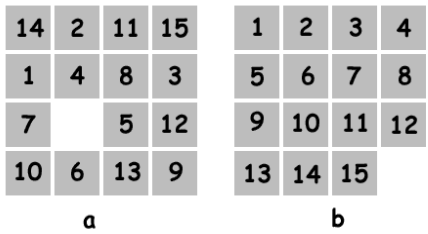
Bab 11
Membuat Game Puzzle

11.1 Perencanaan Game.

11.1.1 Peraturan Game.

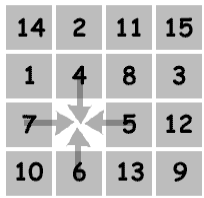
Pada bab ini kita akan membuat game yang merupakan gabungan semua teknik yang telah telah kita pelajari sebelumnya. Kita akan membuat game puzzle, penulis memutuskan menjadikan game puzzle sebagai proyek akhir kita mengingat game puzzle relatif sederhana, mudah dibuat dan tidak membutuhkan gambar dan suara dalam jumlah besar.

Game ini kita beri nama “Go Slide It”. Permainan ini tujuannya adalah untuk mengatur posisi sejumlah kotak yang berisi angka-angka atau potongan gambar sehingga angka-angka atau gambar-gambar tersebut tersusun secara berurutan. Untuk game ini penulis menggunakan gambar angka-angka untuk merepresentasikan kotak-kotak tersebut. Kotak-kotak ini selanjutnya akan kita sebut sebagai sel. Untuk lebih jelasnya perhatikan gambar berikut



Gambar 11.1(a) Sel-sel yang belum terurut (b) sel-sel yang telah terurut.

Pada gambar di atas terdapat sebuah sel kosong yang berfungsi sebagai sel bantuan guna memindahkan sel-sel lain. Untuk mengurutkan sel-sel tersebut, pemain hanya dapat melakukannya dengan memindahkan sel-sel tersebut dengan cara menggeser sel yang hendak dipindah ke sel kosong yang posisinya bersebelahan dengan sel tersebut. Untuk dapat menggeser sel tersebut, sel kosong harus terletak dalam baris atau kolom yang sama dengan baris atau kolom sel yang hendak dipindah. Untuk jelasnya, silakan perhatikan ilustrasi di bawah ini:



Gambar 11.2 Peraturan perpindahan sel.

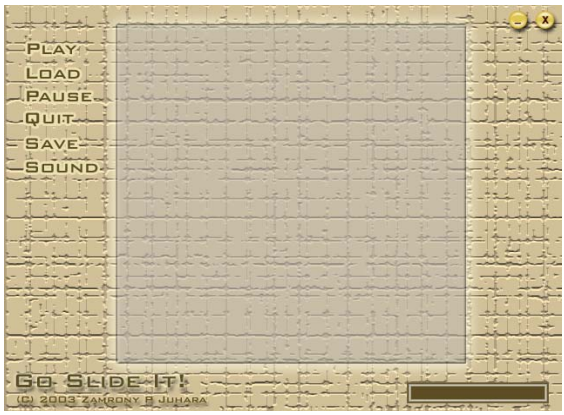
11.1.2 Grafis Game.

11.1.2.1 Mode Grafis yang Digunakan.

Game ini menggunakan resolusi 640x480 dengan kedalaman warna 16 bit full screen. Untuk game ini kita hanya akan menggunakan mode full screen. Pembaca bisa memodifikasinya menjadi mode windowed atau menambah kemampuan mengubah mode full screen menjadi mode windowed atau sebaliknya.

11.1.2.2 Tampilan Menu Utama Program.

Tampilan menu utama game adalah sebagai berikut:



Gambar 11.3 Tampilan menu utama game.

Tampilan utama ini, selanjutnya akan kita sebut sebagai board, akan kita hasilkan dari gambar bitmap. Kita akan menggunakan kelas TBackgroundEngine untuk menampilkan board karena board ini berupa bakground statis yang tidak dapat digeser-geser. Oleh karena itu board ini akan kita buat dari gambar bitmap yang berukuran 640x480, sama dengan resolusi game yang kita gunakan. Gambar board ini disimpan di file *board.bmp*.

11.1.2.3 Gambar Sel-Sel.

Sel-sel akan di animasi ketika kursor mouse berada di atasnya, sehingga sel yang sedang berada dibawah kursor akan tampak berkedip-kedip. Untuk menghasilkan tampilan seperti ini, maka gambar tiap sel akan terdiri atas beberapa bentuk perubahan yakni perubahan sel secara gradasi dari putih menjadi kuning. Jumlah frame perubahan tiap sel adalah 5. Tiap-tiap sel akan dibuat berukuran 64x64 pixel. Gambar sel-sel tersebut digabung menjadi satu file yakni *Go Slide It!\cell.bmp* dan *Go Slide It!\cellspritedata.txt* berisi data sprite tiap-tiap sel. Sel-sel tersebut akan kita simpan dalam kelas TSpriteEngine.

Gambar 11.4 Gambar Sel

11.1.2.4 Gambar Menu.

Ketika kursor mouse berada diatas sebuah menu, misalnya "Play" (lihat Gambar 14.3), maka menu play akan dibuat tampak timbul yang mengindikasikan menu tersebut sedang aktif, demikian pula untuk menu-menu lainnya. Oleh karena itu kita akan membutuhkan gambar-gambar menu aktif. Gambar-gambar tersebut akan kita tumpuk dengan gambar board ketika kita menggambar tampilan game. Gambar-gambar tersebut disimpan di file *Go Slide It!\ button.bmp.*

11.2 Implementasi Game.

11.2.1 Implementasi Peraturan Game.

11.2.1.1 Listing Unit uGameLogic.pas

```
unit ugameLogic;
interface
uses classes,windows;
const MaxCell=4;
type  TGameCell=array[0..MaxCell-1,0..MaxCell-1]of integer;
    TSlidePuzzle=class
      private
       FCells,WinCell: TGameCell;
       FOnRightPlace: TNotifyEvent;
       FOnWin: TNotifyEvent;
       function isUnique(const cellvalue:integer):boolean;
       procedure BuildCell;
       procedure InitCell;
```

```
        function FindEmptyCell(var i,j:integer):boolean;
        procedure SetOnRightPlace(const Value: TNotifyEvent);
    procedure SetOnWin(const Value: TNotifyEvent);
      public
        constructor Create;
        destructor Destroy;override;
        procedure Free;
        procedure Setup;
        function isWin:boolean;
        procedure MoveCell(const i,j:integer);
        property Cells:TGameCell read FCells;
      published
        property OnRightPlace:TNotifyEvent read FOnRightPlace write SetOnRightPlace;
        property OnWin:TNotifyEvent read FOnWin write SetOnWin;
    end;
implementation
{ TSlidePuzzle }
procedure TSlidePuzzle.BuildCell;
var i,j,cellValue:integer;
begin
  for i:=0 to Maxcell-1 do
  for j:=0 to Maxcell-1 do
  begin
    Repeat
      CellValue:=Random(MaxCell*Maxcell);
    Until isUnique(cellvalue);
    FCells[i,j]:=cellValue;
  end;
end;
constructor TSlidePuzzle.Create;
begin
  FOnRightPlace:=nil;
end;
destructor TSlidePuzzle.Destroy;
begin
```

```
  FOnRightPlace:=nil;
  inherited;
end;
function TSlidePuzzle.FindEmptyCell(var i, j: integer): boolean;
var k,l:integer;
    found:boolean;
begin
  found:=false;
  for k:=0 to MaxCell-1 do
  for l:=0 to MaxCell-1 do
  begin
     if FCells[k,l]=0 then
    begin
      found:=true;
      i:=k;
      j:=l;
    end;
    if found then break;
  end;
  result:=found;
end;
procedure TSlidePuzzle.Free;
begin
  if self<>nil then destroy;
end;
procedure TSlidePuzzle.InitCell;
var i,j:integer;
begin
  for i:=0 to Maxcell-1 do
  for j:=0 to Maxcell-1 do
  begin
    FCells[i,j]:=-1;
    WinCell[i,j]:=i*Maxcell+j+1;
  end;
  WinCell[Maxcell-1,Maxcell-1]:=0;
```

```
end;
function TSlidePuzzle.isUnique(const cellvalue: integer): boolean;
var i,j:integer;
    res:boolean;
begin
  res:=true;
  for i:=0 to Maxcell-1 do
  for j:=0 to Maxcell-1 do
  begin
    if FCells[i,j]=CellValue then res:=false;
    if res=false then break;
  end;
  Result:=res;
end;
function TSlidePuzzle.isWin: boolean;
var i,j:integer;
    res:boolean;
begin
  res:=true;
  for i:=0 to Maxcell-1 do
  for j:=0 to Maxcell-1 do
  begin
    if FCells[i,j]<>WinCell[i,j] then res:=false;
    if res=false then break;
  end;
  Result:=res;
end;
procedure TSlidePuzzle.MoveCell;
var ii,jj:integer;
begin
  if findEmptyCell(ii,jj) then
  begin
    if (ii=j) and (jj=i) then
    else
    begin
```

```
    if ((ii=j-1) and (jj=i)) or
      ((ii=j+1) and (jj=i)) or
      ((ii=j) and (jj=i-1)) or
      ((ii=j) and (jj=i+1)) then
    begin
      FCells[ii,jj]:=FCells[j,i];
      FCells[j,i]:=0;
      if FCells[ii,jj]=WinCell[ii,jj] then
      begin
        if Assigned(FOnRightPlace) then
          FOnRightPlace(self);
      end;
    end;
  end;
 end;
 if isWin then
 begin
   if Assigned(FOnWin) then
     FOnWin(self);
 end;
end;
procedure TSlidePuzzle.SetOnRightPlace(const Value: TNotifyEvent);
begin
 FOnRightPlace := Value;
end;
procedure TSlidePuzzle.SetOnWin(const Value: TNotifyEvent);
begin
 FOnWin := Value;
end;
procedure TSlidePuzzle.Setup;
begin
 InitCell;
 BuildCell;
end;
initialization
 Randomize;
```

end.

11.2.1.2 Kelas TSlidePuzzle.

11.2.1.2.1 Pendahuluan.

Kelas ini merupakan kelas yang menangani proses pengacakan sel, pemindahan isi sel sesuai dengan peraturan game yang telah kita buat, sert melakukan pengecekan apakah pemain memenangkan permainan atau tidak.

11.2.1.2.2 Properti.

Properti kelas TSlidePuzzle terdiri atas dua properti event yakni properti event *OnRightPlace* dan event *OnWin*. Event OnRightPlace akan ditimbulkan pada saat sebuah sel menempati posisi yang benar, event OnWin akan ditimbulkan saat pemain berhasil menyelesaikan permainan. Properti lain adalah *Cells* yang berisi informasi isi sel-sel saat ini. Properti ini mengakses data yang tersimpan di variabel internal *FCells*.

11.2.1.2.3 Inisialisasi.

Seperti biasanya inisialisasi dikerjakan oleh konstruktor *Create*. Pada saat inisialisasi kelas ini kita isikan nilai awal event *FonRightPlace* dan *FonWin* sama dengan *nil.*

```
constructor TSlidePuzzle.Create;
begin
  FOnRightPlace:=nil;
  FonWin:=nil
end;
```

11.2.1.2.4 Menyiapkan Sel-Sel.

Untuk mengerjakan proses ini, kelas TSlidePuzzle dilengkapi dengan metode *Setup*.

```
procedure TSlidePuzzle.Setup;
begin
  InitCell;
  BuildCell;
end;
```

Metode Setup memanggil dua metode privat milik TSlidePuzzle yakni *InitCell* dan *BuildCell*. Metode InitCell bertugas untuk menginisialisasi isi FCells dengan -1 dan mengisikan WinCell dengan data sel kemenangan. Setelah InitCell dipanggil maka WinCell akan berisi data-data berikut:

Wincell[0,0]=1, WinCell[0,1]=2, WinCell[0,2]=3, WinCell[0,3]=4,

Wincell[1,0]=5, WinCell[1,1]=6, WinCell[1,2]=7, WinCell[1,3]=8,

Wincell[2,0]=9, WinCell[2,1]=10, WinCell[2,2]=11, WinCell[2,3]=12,

Wincell[3,0]=13, WinCell[3,1]=14, WinCell[3,2]=15, WinCell[3,3]=0

```
procedure TSlidePuzzle.InitCell;
var i,j:integer;
begin
  for i:=0 to Maxcell-1 do
  for j:=0 to Maxcell-1 do
  begin
    FCells[i,j]:=-1;
    WinCell[i,j]:=i*Maxcell+j+1;
  end;
  WinCell[Maxcell-1,Maxcell-1]:=0;
end;
```

Metode BuildCell bertugas untuk mengacak isi FCells sedemikian rupa sehingga semua datanya akan berkisar antara 0 sampai 15, dimana data 0 mewakili sel kosong. Setiap sel akan berisi data yang unik yang acak. Suatu data disebut unik bila tidak ada satu pun sel yang berisi data ini. Berikut ini adalah implementasi BuildCell,

```
procedure TSlidePuzzle.BuildCell;
var i,j,cellValue:integer;
begin
  for i:=0 to Maxcell-1 do
  for j:=0 to Maxcell-1 do
  begin
```

Kita lakukan proses looping sebanyak MaxCell*Maxcell yakni 4x4=16

```
    Repeat
      CellValue:=Random(MaxCell*Maxcell);
```

Ambil nilai antara 0-(16-1) secara acak dengan fungsi Random dan kita simpan nilainya di *CellValue*. Kemudian kita tes apakah nilai yang ada di CellValue bersifat unik dengan memanggil fungsi *isUnique*. Jika ya, kita keluar dari blok *Repeat..Until*, jika tidak, maka kita cari nilai lainnya sampai ditemukan nilai yang unik.

```
    Until isUnique(cellvalue);
    FCells[i,j]:=cellValue;
```

Jika unik isikan nilai ini ke FCells. Looping lagi, sampai semua sel selesai di inisialisasi

```
  end;
```

```
end;
```

Berikut ini adalah implementasi fungsi IsUnique. Fungsi ini membandingkan isi semua sel yang ada di FCells, jika ada yang sama, maka fungsi ini menghasilkan nilai false. Jika unik, maka nilai yang dikembalikan adalah true.

```
function TSlidePuzzle.isUnique(const cellvalue: integer): boolean;
var i,j:integer;
   res:boolean;
begin
 res:=true;
 for i:=0 to Maxcell-1 do
 for j:=0 to Maxcell-1 do
 begin
  if FCells[i,j]=CellValue then res:=false;
  if res=false then break;
 end;
 Result:=res;
end;
```

11.2.1.2.5 Memindahkan Isi Sel.

Proses ini ditangani oleh metode *MoveCell*. Parameternya adalah indeks sel yang akan dipindah. Jika sel ini memenuhi kriteria sebagai sel yang dapat dipindah maka sel tersebut akan dipindah. Jika tidak maka sel tidak akan dipindah.

```
procedure TSlidePuzzle.MoveCell;
var ii,jj:integer;
begin
```

Kita panggil metode *FindEmptyCell*. Fungsi ini berguna untuk mendapatkan indeks sel kosong. Jika sel kosong ditemukan, maka *ii* dan *jj* akan berisi baris dan kolom sel kosong.

```
 if findEmptyCell(ii,jj) then
 begin
  if (ii=j) and (jj=i) then
  else
  begin
   if ((ii=j-1) and (jj=i)) or
      ((ii=j+1) and (jj=i)) or
      ((ii=j) and (jj=i-1)) or
      ((ii=j) and (jj=i+1)) then
```

```
    begin
```

Jika sel yang hendak dipindah bukan sel kosong maka pindahkan isi sel tersebut ke sel kosong dan isi sel tersebut kita isi nol sehingga menjadi sel kosong yang baru.

```
      FCells[ii,jj]:=FCells[j,i];
      FCells[j,i]:=0;
      if FCells[ii,jj]=WinCell[ii,jj] then
      begin
```

Jika isi sel sama dengan isi sel kemenangan maka kita timbulkan event OnRightPlace dengan memanggil handler event ini.

```
        if Assigned(FOnRightPlace) then
          FOnRightPlace(self);
      end;
    end;
  end;
 end;
 if isWin then
 begin
```

Kita tes apakah semua sel berisi data-data yang benar dengan memanggil fungsi *isWin*. Jika ya maka kita timbulkan event OnWin.

```
   if Assigned(FOnWin) then
      FOnWin(self);
 end;
end;
```

Metode MoveCell memanggil dua metode yakni FindEmptyCell dan isWin. FindEmptyCell mencari indeks sel kosong dengan looping. Jika ditemukan sel yang berisi nol, maka indeks sel ini dikembalikan ke parameter dan proses keluar dari looping. Fungsi IsWin juga melakukan looping dan membandingkan isi FCells dengan WinCell. Jika isi FCells seluruhnya sama dengan isi WinCell, berarti pemain telah berhasil menyelesaikan permainan ini.

Berikut ini adalah listing FindEmptyCell dan isWin,

```
function TSlidePuzzle.FindEmptyCell(var i, j: integer): boolean;
var k,l:integer;
   found:boolean;
begin
 found:=false;
```

Kita asumsikan sel kosong belum ditemukan dengan mengisi *found* sama dengan false. Kita lakukan looping sebanyak jumlah sel untuk mencari sel kosong.

```
 for k:=0 to MaxCell-1 do
 for l:=0 to MaxCell-1 do
 begin
   if FCells[k,l]=0 then
  begin
```

Jika sel adalah sel kosong, maka kita telah menemukan apa yang kita cari, sehingga kita simpan indeks baris dan kolom sel kosong yang telah ditemukan serta segera keluar dari looping.

```
    found:=true;
    i:=k;
    j:=l;
  end;
  if found then break;
 end;
 result:=found;
```

Kita isi nilai fungsi dengan status yang tercatat di found.

```
end;
function TSlidePuzzle.isWin: boolean;
var i,j:integer;
   res:boolean;
begin
 res:=true;
```

Kita asumsikan bahwa pemain telah berhasil menyelesaikan permainan.

```
 for i:=0 to Maxcell-1 do
 for j:=0 to Maxcell-1 do
 begin
   if FCells[i,j]<>WinCell[i,j] then res:=false;
   if res=false then break;
```

Jika isi FCells dan WinCell tidak sama, maka pemain belum berhasil menyelesaikan permainan sehingga status *res* kita isi false dan keluar dari looping.

```
 end;
 Result:=res;
end;
```

11.2.2 Implementasi Aplikasi Game Puzzle.

11.2.2.1 Listing dan Penjelasan Unit UfrmMain.Pas.

Sedikit berbeda dengan bab-bab sebelumnya, di mana sub bab listing dan sub bab penjelasan listing program penulis pisahkan, pada sub bab ini listing program dan penjelasan listing program digabung menjadi satu mengingat keterbatasan ruang. Semoga hal ini tidak membuat pembaca bingung.

```
unit ufrmMain;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs,uDirectDraw,uDirectSound,uDirectInput, ExtCtrls,uGameLogic, ufilelist;
type
  TfrmMain = class(TForm)
    Timer1: TTimer;
```

Pada form utama, kita tambahkan instance kelas TTimer. TTimer kita gunakan untuk melakukan proses update tampilan dilayar, proses update data dan deteksi input. Properti interval timer kita atur sebesar 30 ms dan rutin dimana proses update ini kita lakukan adalah prosedur Timer1Timer.

```
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    SaveDir:String;
    SoundFX:TSoundEffect;
    MouseInput:TMouseInput;
    KeyboardInput:TKeyboardInput;
    BaseInput:TBaseInput;
    GraphicEngine:TGraphicEngine;
    Back:TBackgroundEngine;
    CellSprite:TSpriteEngine;
    MiscSprite:TSpriteEngine;
    TxtSprite:TFontEngine2;
    SlidePuzzle:TSlidePuzzle;
```

Kita deklarasikan variable-variabel privat yang bertanggungjawab menangani proses input (*MouseInput*, *KeyboardInput*, *BaseInput*), proses update tampilan layar (*GraphicEngine*), proses menampilkan sprite, background, font (*Back*, *CellSprite*, *MiscSprite*, *TxtSprite*), proses memainkan suara (*SoundFX*) serta proses implementasi peraturan game (*SlidePuzzle*). Kita juga menyediakan tempat penampung nama direktori yang digunakan untuk menampung file saved game (*SaveDir*).

```
MouseXY:TPoint;
MouseDwn,MouseOver,Quit,AnimHighlight:boolean;
ActiveCellX,ActiveCellY,
PreviousActiveCellX,PreviousActiveCellY:integer;
CellOffset:integer;
MenuHighlight,PrevMenuHighlight:integer;
TimeTick,StartTick:integer;
Started:boolean;
Filename:string;
```

MouseXY kita gunakan untuk menyimpan koordinat mouse saat ini, *MouseDwn* mencatat status penekanan tombol mouse, *MouseOver* mencatat status mouse apakah sedang berada di atas menu atau cell. *Quit* digunakan untuk keluar dari program aplikasi. *AnimHighlight* digunakan untuk melakukan proses animasi sel yang terlihat berkedip-kedip ketika mouse berada di atas sel. *ActiveCellX* dan *ActiveCellY* menyimpan data baris dan kolom sel yang berada di bawah mouse. *PreviousActiveCellX* dan *PreviousActiveCellY* menyimpan data baris dan kolom sel yang aktif sebelumnya. *MenuHighlight* mencatat indeks menu yang saat ini sedang berada dibawah mouse. *PreviousMenuHighlight* menyimpan indeks menu yang sebelumnya berada dibawah mouse. *TimeTick* diperlukan untuk menyimpan waktu yang telah dilampaui oleh pemain sejak pertama kali mulai, sedangkan *StartTick* mencatat kapan pemain mulai start. *Started* digunakan untuk memulai proses penghitungan waktu. Started akan bernilai true saat pemain mengklik sel untuk pertama kali. Filename duperlukan untuk menampilkan nama file saved game.

```
//inisialisasi & finalisasi
procedure CreateGraphicEngine;
procedure DestroyGraphicEngine;
procedure CreateBackground;
procedure DestroyBackground;
procedure CreateCellSprite;
procedure DestroyCellSprite;
procedure CreateMiscSprite;
procedure DestroyMiscSprite;
procedure CreateTxtSprite;
```

```
    procedure DestroyTxtSprite;

    procedure CreateBaseInput;
    procedure DestroyBaseInput;
    procedure CreateMouseInput;
    procedure DestroyMouseInput;
    procedure CreateKeyboardInput;
    procedure DestroyKeyboardInput;
    procedure CreateSoundEffect;
    procedure DestroySoundEffect;
    procedure CreateSlidePuzzle;
    procedure DestroySlidePuzzle;
  //Loading data
    procedure LoadCellSpriteData;
    procedure LoadTxtSpriteData;
    procedure LoadBackground;
    procedure OpenSavedData(const filename:string);
  //rendering
    procedure UpdateScreen;
    procedure UpdateData;
    procedure DetectInput;
    procedure DetectMouseOnCells;
    procedure DetectMouseOnMenu;
    function  isMouseXYIn(const mousex,mousey:integer;const ARect:TRect):boolean;
    procedure LoadmiscSpriteData;
    procedure DisplayMouse;
    procedure DisplayCell;
    procedure DisplayTime;
    function  TickToStr(const ticks:integer):string;
    procedure DisplayMenuHighLight;
  //dialog
    function DisplayCloseDlg:integer;
    function DisplaySoundDlg:integer;
    procedure DisplayCloseDlgMenu(const x,y:integer;var menuindex:integer;var clicked:boolean);
    procedure DisplaySoundDlgMenu(const x,y:integer;var menuindex:integer;var clicked:boolean);
```

```
    procedure DisplaySoundDlgTrackBar(const minx, maxx: integer; var xx,
      yy: integer;var clicked:boolean);
    function DisplayPauseDlg:integer;
    procedure DisplayPauseDlgMenu(const x,y:integer;var menuindex:integer;var clicked:boolean);
    function DisplayWinDlg:integer;
    procedure DisplayWinDlgMenu(const x,y:integer;var menuindex:integer;var clicked:boolean);
    function DisplaySaveDlg:integer;
    procedure DisplaySaveDlgMenu(const x,y:integer;var menuindex:integer;var
clicked:boolean;highlightfile:TFillRect;const trackbarclicked:boolean);
    function DisplayLoadDlg:integer;
    procedure DisplayLoadDlgMenu(const x,y:integer;var menuindex:integer;var
clicked:boolean;highlightfile:TFillRect;const TrackBarClicked:boolean);
    procedure DisplayFileList(const x,y:integer;afileList:TFileList;const
FirstIndex,LastIndex:integer;txtSprite:TFontEngine2);
    procedure DisplayFileHighlight(const x,y:integer;afillrect:TFillRect);
    procedure DoCloseApp;
    procedure DoSoundDlg;
    procedure DoPauseDlg;
    procedure DoWinDlg;
    procedure DoSaveDlg;
    procedure DoLoadDlg;
  //playsound
    procedure PlayEffect(const effect:integer);
    procedure PlayBackgroundMusic;
  //------event handler--
    procedure CellInRightPlace(sender:TObject);
    procedure Win(sender:TObject);
    function DisplayCredit: integer;
    procedure DisplayCreditMenu(const x, y: integer;
      var menuindex: integer; var clicked: boolean);
    procedure DoCredit;
    function DisplayFileNotFound: integer;
    procedure DisplayFileNotFoundMenu(const x, y: integer;
      var menuindex: integer; var clicked: boolean);
    procedure DoFileNotFound;
    procedure DisplaySaveLoadDlgTrackBar(const minx, maxx,afilecount,maxfiledisplay: integer; var xx,
      yy,indexFirst,indexLast: integer; var aclicked: boolean);
```

```
  { Private declarations }
 public
  { Public declarations }
 end;
var
 frmMain: TfrmMain;
implementation
uses DirectInput,uconst, usnd;
{$R *.dfm}
```

CreateBackground digunakan untuk menginisialisasi instance TBackgroundEngine.

```
procedure TfrmMain.CreateBackground;
begin
 Back:=TBackgroundEngine.Create(GraphicEngine);
end;
```

CreateBaseInput menginisialisasi instance TBaseInput yang selanjutnya akan digunakan untuk menciptakan instance TKeyboardInput dan TMouseInput.

```
procedure TfrmMain.CreateBaseInput;
var InputParam:TInputParam;
begin
 ZeroMemory(@InputParam,SizeOf(TInputParam));
Kita isi InputParam dengan nol.
 InputParam.H_Instance:=HInstance;
 InputParam.Handle:=Handle;
 InputParam.CooperativeLevel.Exclusive:=true;
 InputParam.CooperativeLevel.Foreground:=true;
 BaseInput:=TBaseInput.Create(self,InputParam);
```

Kita lakukan inisialisasi InputParam dengan handle instance dan handle window form utama kita serta level kooperatif yang kita inginkan. Selanjutnya instance TBaseInput kita ciptakan.

```
end;
```

CreateCellSprite digunakan menciptakan instance TSpriteEngine. CellSprite ini digunakan untuk menampilkan sprite sel-sel ke layar.

```
procedure TfrmMain.CreateCellSprite;
begin
 CellSprite:=TSpriteEngine.Create(GraphicEngine);
```

```
end;
```

Proses menciptakan instance TGraphicEngine dan instance TKeyboardInput dan TMouseInput hampir sama dengan contoh-contoh program kita sebelumnya, pembaca bisa melihat contoh-contoh program sebelumnya jika bingung dengan cara insialisasinya.

```
procedure TfrmMain.CreateGraphicEngine;
var geParam:TGraphicEngineParam;
begin
  ZeroMemory(@geParam,SizeOf(TGraphicEngineParam));
  geParam.Handle:=Handle;
  geParam.Width:=640;
  geParam.Height:=480;
  geParam.BitPerPixel:=16;
  geParam.BackBufferCount:=1;
  geParam.FullScreen:=true;
  geParam.AllowReboot:=true;
  geParam.Clipping:=true;
  GraphicEngine:=TGraphicEngine.Create(geParam);
end;
procedure TfrmMain.CreateKeyboardInput;
var InputParam:TInputParam;
begin
  ZeroMemory(@InputParam,SizeOf(TInputParam));
  InputParam.H_Instance:=HInstance;
  InputParam.Handle:=Handle;
  InputParam.CooperativeLevel.Exclusive:=true;
  InputParam.CooperativeLevel.Foreground:=true;
  KeyboardInput:=TKeyboardInput.Create(BaseInput,InputParam);
end;
```

MiscSprite adalah instance yang akan kita gunakan untuk menampilkan sprite menu seperti menu Load, Quit, Save serta gambar tombol-tombol pada window dialog.

```
procedure TfrmMain.CreateMiscSprite;
begin
  MiscSprite:=TSpriteEngine.Create(GraphicEngine);
end;
procedure TfrmMain.CreateMouseInput;
```

```
var InputParam:TInputParam;
begin
  ZeroMemory(@InputParam,SizeOf(TInputParam));
  InputParam.H_Instance:=HInstance;
  InputParam.Handle:=Handle;
  InputParam.CooperativeLevel.Exclusive:=true;
  InputParam.CooperativeLevel.Foreground:=true;
  MouseInput:=TMouseInput.Create(BaseInput,InputParam);
end;
procedure TfrmMain.CreateSoundEffect;
var SoundParam:TSoundParam;
begin
  ZeroMemory(@SoundParam,SizeOf(TSoundParam));
  SoundParam.Handle:=Handle;
  SoundParam.CooperativeLevel:=cplNormal;
  SoundParam.Caps.scStatic:=true;
  SoundParam.Caps.scCtrlVolume:=true;
  SoundParam.Caps.scCtrlPan:=true;
  SoundParam.Caps.scCtrlFrequency:=true;
```

SoundFile adalah nama file yang berisi daftar file-file .WAV yang akan kita gunakan untuk efek suara. Deklarasinya ada pada file uconst.pas

```
  SoundParam.Filename:=SoundFile;
  SoundFX:=TSoundEffect.Create(SoundParam);
end;
procedure TfrmMain.CreateTxtSprite;
begin
  TxtSprite:=TFontEngine2.Create(GraphicEngine);
end;
procedure TfrmMain.DestroyBackground;
begin
  back.Free;
end;
procedure TfrmMain.DestroyBaseInput;
begin
  BaseInput.Free;
```

```
end;
procedure TfrmMain.DestroyCellSprite;
begin
  CellSprite.Free;
end;
procedure TfrmMain.DestroyGraphicEngine;
begin
  GraphicEngine.Free;
end;
procedure TfrmMain.DestroyKeyboardInput;
begin
  KeyboardInput.Free;
end;
procedure TfrmMain.DestroyMiscSprite;
begin
  MiscSprite.Free;
end;
procedure TfrmMain.DestroyMouseInput;
begin
  MouseInput.Free;
end;
procedure TfrmMain.DestroySoundEffect;
begin
  SoundFx.Free;
end;
procedure TfrmMain.DestroyTxtSprite;
begin
  TxtSprite.Free;
end;
```

DetectInput adalah prosedur yang kita gunakan untuk mendeteksi adanya input dari pengguna. Prosedur ini dipanggil setiap kali event OnTimer milik Timer1 dibangkitkan. Karena sering dipanggil prosedur ini harus dibuat secepat dan seringkas mungkin.

```
procedure TfrmMain.DetectInput;
var x,y,z:integer;
begin
```

```
 MouseInput.GetDeviceState;
 KeyboardInput.GetDeviceState;
```

Kita ambil status mouse dan keyboard. Jika tombol keyboard yang ditekan adalah tombol Escape maka kita jalankan prosedur *DoCloseApp* yang akan menutup aplikasi.

```
 if KeyboardInput.KeyDown(DIK_ESCAPE) then DoCloseApp;
 MouseInput.MousePos(X,y,z);
```

Kita ambil posisi mouse saat ini dan kita simpan di x, y, z. Seperti yang telah penulis jelaskan sebelumnya. Posisi yang dikembalikan ke x, y, z bukanlah nilai yang mengacu pada koordinat (0,0) melainkan total perpindahan relatif terhadap posisi sebelumnya. Oleh karena itu untuk mendapatkan koordinat mouse yang benar maka x, dan y (z kita abaikan) kita tambahkan dengan koordinat mouse sebelumnya.

```
 MouseXY.X:=MouseXY.X+X;
 MouseXY.Y:=MouseXY.Y+Y;
```

Kita lakukan pengecekan posisi mouse apakah berada di atas menu atau sel dengan menjalankan rutin berikut ini.

```
 DetectMouseOnMenu;
 DetectMouseOnCells;
end;
```

Proses inisialisasi instance dan data-data game dikerjakan pada FormCreate dengan memanggil prosedur-prosedur yang bertanggung jawab melakukan inisialisasi instance, sedangkan proses finalisasi, seperti biasa dikerjakan oleh FormDestroy.

```
procedure TfrmMain.FormCreate(Sender: TObject);
begin
 CreateBaseInput;
 CreateMouseInput;
 CreateKeyboardInput;
 CreateSoundEffect;

 CreateGraphicEngine;
 CreateBackground;
 CreateCellSprite;
 CreateMiscSprite;
 CreateTxtSprite;
 CreateSlidePuzzle;
 LoadBackground;
```

```
  LoadMiscSpritedata;
  LoadCellSpritedata;
  LoadtxtSpriteData;
  TimeTick:=0;
  Timer1.Enabled:=true;
  PlayBackgroundMusic;
  Filename:='';
  SaveDir:=ExtractFilePath(Application.ExeName)+'save\';
end;
procedure TfrmMain.FormDestroy(Sender: TObject);
begin
  DestroyKeyboardInput;
  DestroyMouseInput;
  DestroyBaseInput;
  DestroySoundEffect;
  DestroyBackground;
  DestroyCellSprite;
  DestroyMiscSprite;
  DestroyGraphicEngine;
  DestroyTxtSprite;
  DestroySlidePuzzle;
end;
```

Prosedur-prosedur berikut ini kita gunakan untuk membaca file-file data yang diperlukan meliputi pembacaan file background dan file data sprite-sprite. Nama file background disimpan di *BackgroundFile* yang deklarasinya terdapat pada file uconst.pas, demikian juga *CellSpriteFile* dan *MiscSpriteFile* deklarasinya terdapat pad uconst.pas.

```
procedure TfrmMain.LoadBackground;
begin
  back.LoadFromFile(BackgroundFile);
end;
procedure TfrmMain.LoadCellSpriteData;
begin
  CellSprite.LoadFromFile(CellSpriteFile);
end;
procedure TfrmMain.LoadMiscSpriteData;
```

```
begin
  miscSprite.LoadFromFile(miscSpriteFile);
end;
```

Prosedur LoadTxtSpriteData tidak mengerjakan apa-apa karena instance TFontEngine2 saat diciptakan secara otomatis telah berisi data sprite font yang diperlukan. Penulis menggunakan rutin ini untuk membaca sprite font karena sebelumnya txtSprite bertipe TFontEngine, yang kemudian penulis ubah menjadi TFontEngine2

```
procedure TfrmMain.LoadTxtSpriteData;
begin
//  txtSprite.LoadFromFile(TxtSpriteFile);
end;
```

Prosedur Timer1Timer merupakan prosedur utama game. Dan dipanggil saat terjadi event OnTimer. Proses yang dilakukan adalah dengan mendeteksi input, memperbarui data-data sel dan melakukan proses penggambaran.

```
procedure TfrmMain.Timer1Timer(Sender: TObject);
begin
  DetectInput;
  UpdateData;
  UpdateScreen;
end;
```

DisplayMouse sesuai namanya, bertugas menampilkan sprite kursor mouse. Posisi dimana sprite diletakkan disimpan di MouseXY. Nomer spritenya adalah nomer 4.

```
Procedure TfrmMain.DisplayMouse;
begin
  MiscSprite.FrameNow:=4;
  Miscsprite.Position:=SetPosition(MouseXY.X,MouseXY.Y,0);
  MiscSprite.Show;
end;
```

DisplayMenuHighLight bertugas menampilkan menu-menu yang sedang disorot oleh mouse. Menu yang sedang disorot adalah menu yang tepat berada dibawah kursor mouse.

```
Procedure TfrmMain.DisplayMenuHighLight;
begin
  if (MouseXY.X>16) and (MouseXY.X<=90) and
     (MouseXY.Y>36) and (MouseXY.Y<=66) then
```

```
 begin
```

Jika mouse berada di dalam rectangle (16,36)(90,66) maka kursor menyorot menu Play. Kita kembalikan index menu yang bersesuaian dengan mengisi MenuHiglight=0.

```
   MiscSprite.FrameNow:=0; //play highlight
   Miscsprite.Position:=SetPosition(16,36,0);
   MenuHighlight:=0;
 end else
 if (MouseXY.X>16) and (MouseXY.X<=90) and
    (MouseXY.Y>65) and (MouseXY.Y<=95) then
 begin
```

Jika mouse berada di dalam rectangle (16,65)(90,95) maka kursor menyorot menu Load.

```
   MiscSprite.FrameNow:=3; //Load highlight
   Miscsprite.Position:=SetPosition(16,65,0);
   MenuHighlight:=1;
 end else
 if (MouseXY.X>16) and (MouseXY.X<=114) and
    (MouseXY.Y>93) and (MouseXY.Y<=123) then
 begin
```

Jika mouse berada di dalam rectangle (16,93)(114,123) maka kursor menyorot menu Pause.

```
   MiscSprite.FrameNow:=1; //Pause highlight
   Miscsprite.Position:=SetPosition(16,93,0);
   MenuHighlight:=2;
 end else
 if (MouseXY.X>16) and (MouseXY.X<=90) and
    (MouseXY.Y>119) and (MouseXY.Y<=149) then
 begin
```

Jika mouse berada di dalam rectangle (16,119)(90,149) maka kursor menyorot menu Quit.

```
   MiscSprite.FrameNow:=2; //quit highlight
   Miscsprite.Position:=SetPosition(16,119,0);
   MenuHighlight:=3;
 end else
 if (MouseXY.X>15) and (MouseXY.X<=98) and
```

```
    (MouseXY.Y>148) and (MouseXY.Y<=178) then
  begin
```

Jika mouse berada di dalam rectangle (15,148)(98,178) maka kursor menyorot menu Save.

```
   MiscSprite.FrameNow:=10; //Save highlight
    Miscsprite.Position:=SetPosition(15,148,0);
    MenuHighlight:=4;
  end else
  if (MouseXY.X>15) and (MouseXY.X<=112) and
    (MouseXY.Y>178) and (MouseXY.Y<=208) then
  begin
```

Jika mouse berada di dalam rectangle (15,178)(112,208) maka kursor menyorot menu Sound.

```
    MiscSprite.FrameNow:=11; //Sound highlight
    Miscsprite.Position:=SetPosition(15,178,0);
    MenuHighlight:=5;
  end else
//credit
  if (MouseXY.X>577) and (MouseXY.X<=607) and
    (MouseXY.Y>3) and (MouseXY.Y<=36) then
  begin
```

Jika mouse berada di dalam rectangle (577,3)(607,36) maka kursor menyorot menu Credit (tombol _ di pojok kanan atas).

```
    MiscSprite.FrameNow:=12; //credit highlight
    Miscsprite.Position:=SetPosition(577,3,0);
    MenuHighlight:=6;
  end else
//exit
  if (MouseXY.X>607) and (MouseXY.X<=637) and
    (MouseXY.Y>3) and (MouseXY.Y<=36) then
  begin
```

Jika mouse berada di dalam rectangle (607,3)(637,36) maka kursor menyorot menu Exit (tombol X pada pojok kanan atas board).

```
    MiscSprite.FrameNow:=13; //exit highlight
    Miscsprite.Position:=SetPosition(607,3,0);
    MenuHighlight:=7;
```

```
 end;
```

Gambar sprite yang bersesuaian dengan menu yang sedang disorot.

```
 MiscSprite.Show;
 if MenuHighlight<>PrevMenuHighlight then
 begin
```

Jika indeks menu saat ini tidak sama dengan indeks menu yang aktif sebelumnya, maka ini berarti kursor mouse menyorot menu yang lain. Kita mainkan efek suara no.0 untuk mengindikasikan perubahan ini. Kita juga mengisikan indeks menu saat ini ke PrevMenuHighlight untuk keperluan pengecekan berikutnya.

```
  PlayEffect(0);
  PrevMenuHighlight:=MenuHighlight;
 end;
end;
```

DisplayCell berfungsi untuk menggambar sel-sel.

```
Procedure TfrmMain.DisplayCell;
var i,j,x,y,cellvalue,frame:integer;
begin
 for i:=0 to Maxcell-1 do
 for j:=0 to Maxcell-1 do
 begin
  cellValue:=SlidePuzzle.Cells[i,j];
```

Kita perlu mengecek apakah isi sel bernilai nol karena isi sel nol bersesuaian dengan sel kosong yang tidak perlu kita tampilkan.

```
  if cellValue<>0 then
  begin
```

Perhitungan di bawah ini digunakan untuk menghitung nomer frame yang bersesuaian dengan sel yang harus ditampilkan. Nilai sel perlu dikurangi satu karena nomer frame dimulai dari nol yang kemudian kita kalikan dengan 5 karena pada file cell.bmp terdapat lima perubahan animasi untuk tiap-tiap sel.

```
   frame:=(cellValue-1) shl 2+(cellValue-1);//frame=(cellvalue-1)*5
   if (i=ActiveCellX) and (j=ActiveCellY) then
```

Jika sel i, j adalah sel yang sedang disorot oleh kursor mouse, maka kita tampilkan animasi perubahan warna sel dari putih ke kuning serta sebaliknya. Untuk mendapatkan efek ini, frame kita tambahkan dengan CellOffset. CellOffset ini nilainya akan di update terus selama kursor mouse berada diatas sel. Jika nilainya kurang dari 5, nilai CellOffset ditambah satu, sedangkan jika sama dengan 5, nilainya akan dikurangi satu.

```
   CellSprite.FrameNow:=frame+CellOffset else
```

Jika sel bukan sel yang sedang disorot tampilkan sel seperti apa adanya.

```
      CellSprite.FrameNow:=frame;
      x:=CellPos[i,j].Left;
      y:=CellPos[i,j].Top;
      Cellsprite.Position:=SetPosition(x,y,0);
      CellSprite.Show;
    end;
  end;
end;
procedure TfrmMain.UpdateData;
var ticks:integer;
begin
   if started then
  begin
```

Proses yang dikerjakan oleh UpdateData sebenarnya adalah menghitung waktu yang telah dilampaui oleh pemain. Jika permainan telah dimulai maka kita lakukan prose penghitungan waktu. Waktu sejak Windows di start dapat kita ketahui dengan fungsi Windows API GetTickCount. Nilai yang dikembalikan fungsi ini kita simpan di variabel sementara ticks. Untuk mendapatkan waktu yang telah dilampaui pemain, kita kurangkan dengan waktu saat pemain mulai.

```
    Ticks:=GetTickCount;
    TimeTick:=Ticks-StartTick;
  end;
end;
```

Proses penggambaran layar dikerjakan di prosedur UpdateScreen.

```
procedure TfrmMain.UpdateScreen;
begin
  back.Show;
```

Kita hapus gambar yang ada di back buffer dengan gambar background.

```
  DisplayCell;
```

Sprite sel-sel digambar di back buffer.

```
  DisplayMenuHighLight;
```

Kita tampilkan menu-menu yang aktif jika ada menu yang disorot oleh mouse.

```
  DisplayTime;
```

Kita tampilkan informasi waktu yang telah dilampaui.

```
  DisplayMouse;
```

Kita gambar kursor mouse paling akhir agar kursor mouse selalu berada diatas sprite-sprite lain.

```
  GraphicEngine.Show;
```

Back buffer yang telah disiap,dipindahkan ke front buffer agar terlihat dilayar.

```
end;
```

Dua prosedur berikut ini mengerjakan inisialisasi dan finalisasi instance SlidePuzzle.

```
procedure TfrmMain.CreateSlidePuzzle;
begin
  SlidePuzzle:=TSlidePuzzle.Create;
  SlidePuzzle.OnRightPlace:=CellInRightPlace;
  SlidePuzzle.OnWin:=Win;
```

Jika event OnRightPlace dibangkitkan maka rutin-rutin yang ada pada prosedur CellInRightPlace akan dijalankan. Demikian pula jika event OnWin terjadi rutin yang ada pada prosedur Win akan dijalankan.

```
  SlidePuzzle.Setup;
```

Kita panggil metode Setup untuk mengacak isi sel-sel.

```
end;
procedure TfrmMain.DestroySlidePuzzle;
begin
  SlidePuzzle.Free;
end;
```

PlayEffect dan PlayBackgroundMusic sesuai namanya digunakan untuk memainkan efek suara dan musik latar selama game berjalan.

```
procedure TfrmMain.PlayEffect(const effect: integer);
begin
  soundfx.EffectNow:=effect;
  soundfx.Play;
end;
procedure TfrmMain.PlayBackgroundMusic;
begin
  soundfx.EffectNow:=14;
  soundfx.PlayLoop;
end;
```

Prosedur DetectMouseOnCells kita pergunakan untuk mendeteksi apakah kursor mouse berada diatas sel.

```
procedure TfrmMain.DetectMouseOnCells;
var i,j:integer;
   mousein:boolean;
begin
  for i:=0 to MaxCell-1 do
  for j:=0 to MaxCell-1 do
  begin
```

Untuk tiap-tiap sel kita lakukan pengecekan apakah kursor berada diatas suatu sel.

```
   mousein:=isMouseXYIn(MouseXY.X,mouseXY.Y,CellPos[i,j]);
   if mouseIn then
   begin
```

Jika ya maka kita simpan indeks baris dan kolom sel yang sebelumnya aktif, kita ganti isi sel yang aktif saat ini dengan indeks baris dan kolom yang baru.

```
    PreviousActiveCellX:=ActiveCellX;
    PreviousActiveCellY:=ActiveCellY;
    ActiveCellX:=i;
    ActiveCellY:=j;
```

Variabel animHighlight kita gunakan sebagai indikator animasi.

```
    if animHighlight then
    begin
```

Jika animHighlight bernilai true, berarti kita akan melakukan animasi perubahan sel dari sel berwarna putih ke sel berwarna kuning. Untuk menghasilkan efek perubahan sel putih ke kuning, nilai CellOffset kita jumlahkan dengan satu. Kita juga perlu membatasi nilai maksimum CellOffset yaitu 4. Ini berkaitan dengan jumlah frame perubahan sel-sel yakni 5. Jika nilai CellOffset sama dengan 4 maka kita ubah nilai animHighlight menjadi false. Tujuannya agar ketika DetectMouseOnCells dipanggil lagi rutin yang dijalankan adalah rutin animasi perubahan sel kuning menjadi sel putih.

```
     if cellOffset<4 then inc(cellOffset)
     else
      animHighlight:=false;
    end else
    begin
```

Jika animHighlight bernilai true, false nilai CellOffset kita kurangi dengan satu. Kita juga perlu membatasi nilai minimum CellOffset yaitu 0. Jika nilai CellOffset sama dengan 0 maka kita ubah nilai animHighlight menjadi true.

```
    if cellOffset>0 then dec(cellOffset)
    else
     animHighlight:=true;
  end;
  if MouseInput.MouseDown(0) then
  begin
```

Jika tombol kiri mouse ditekan pada saat kursor berada di atas sel, kita perlu mengetahui apakah permainan baru saja dimulai. Jika started bernilai false maka klik yang baru saja terjadi merupakan tanda pemain telah siap memulai permainan, oleh karena itu kita ubah nilai started menjadi true, waktu saat game pertama kali di start disimpan di StartTick. Saat klik terjadi kita mainkan efek suara nomer 12 yang berisi suara tembakan dan melakukan pemindahan isi sel jika ada pemindahan yang harus dilakukan.

```
    if not started then
    begin
      started:=true;
      StartTick:=GetTickCount;
      Timer1Timer(self);
    end;
    PlayEffect(12);
    SlidePuzzle.MoveCell(j,i);
  end;
```

Keluar dari prosedur.

```
    exit;
  end;
end;
```

Jika kursor tidak berada diatas sel manapun maka kita isi nilai CellOffset dengan 0 sehingga nantinya sel-sel akan ditampilkan dengan gambar defaultnya yang berwarna putih.

```
 CellOffset:=0;
end;
```

Fungsi berikut berguna untuk melakukan pengecekan posisi kursor mouse apakah berada dalam suatu rectangle.

```
function TfrmMain.isMouseXYIn(const mousex,mousey:integer;
```

```
  const ARect: TRect): boolean;
begin
  result:=false;
  if (MouseX>ARect.Left)and
    (MouseX<Arect.Right) and
    (MouseY>Arect.Top) and
    (MouseY<Arect.Bottom) then
  begin
    Result:=true;
  end;
end;
```

Prosedur di bawah dijalankan tiap kali sebuah sel yang baru dipindah berada pada tempat yang benar. Tiap kali event OnRightPlace terjadi maka kita mainkan suara nomer 13 (suara ledakan).

```
procedure TfrmMain.CellInRightPlace(sender: TObject);
begin
  PlayEffect(13);
end;
procedure TfrmMain.DetectMouseOnMenu;
begin
  if (MouseXY.X>16) and (MouseXY.X<=90) and
     (MouseXY.Y>36) and (MouseXY.Y<=66) then
  begin
    if MouseInput.MouseDown(0) then
    begin
```

Jika mouse diklik saat berada pada menu Play. Kita acak ulang isi sel dan restart game. Kita juga memainkan suara nomer 6.

```
      SlidePuzzle.Setup;
      started:=false;
      TimeTick:=0;
      PlayEffect(6);
    end;
  end else
  if (MouseXY.X>16) and (MouseXY.X<=90) and
     (MouseXY.Y>65) and (MouseXY.Y<=95) then
  begin
```

```
  if MouseInput.MouseDown(0) then
  begin
```

Jika mouse diklik saat berada pada menu Load. Kita tampilkan dialog load untuk membaca file saved game.

```
    PlayEffect(8);
    DoLoadDlg;
  end;
end else
if (MouseXY.X>16) and (MouseXY.X<=114) and
  (MouseXY.Y>93) and (MouseXY.Y<=123) then
begin
  if MouseInput.MouseDown(0) then
  begin
```

Jika game dihentikan sementara, tampilkan informasi pause serta kita mainkan suara nomer 8.

```
    PlayEffect(8);
    DoPausedlg;
  end;
end else
if (MouseXY.X>15) and (MouseXY.X<=112) and
  (MouseXY.Y>178) and (MouseXY.Y<=208) then
begin
  if MouseInput.MouseDown(0) then
  begin
```

Jika menu Sound diklik kita mainkan suara nomer 7 dan menampilkan dialog sound.

```
    PlayEffect(7);
    DoSoundDlg;
  end;
end else
if (MouseXY.X>15) and (MouseXY.X<=98) and
  (MouseXY.Y>148) and (MouseXY.Y<=178) then
begin
  if MouseInput.MouseDown(0) then
  begin
```

Jika mouse diklik saat berada pada menu Save. Kita tampilkan dialog save untuk menyimpan file saved game.

```
    PlayEffect(8);
    DoSaveDlg;
  end;
 end else
 if (MouseXY.X>16) and (MouseXY.X<=90) and
   (MouseXY.Y>119) and (MouseXY.Y<=149) then
 begin
```

Jika menu Quit diklik tutup aplikasi.

```
  if MouseInput.MouseDown(0) then DoCloseApp;
 end else
 if (MouseXY.X>577) and (MouseXY.X<=607) and
   (MouseXY.Y>3) and (MouseXY.Y<=36) then
 begin
```

Tombol kredit (tombol _ pada pojok kanan atas) diklik, tampilkan informasi kredit.

```
  if MouseInput.MouseDown(0) then DoCredit;
 end else
//exit
 if (MouseXY.X>607) and (MouseXY.X<=637) and
   (MouseXY.Y>3) and (MouseXY.Y<=36) then
 begin
```

Tombol exit (tombol X pada pojok kanan atas) diklik, tutup aplikasi. *if MouseInput.MouseDown(0) then DoCloseApp;*

```
 end;
end;
procedure TfrmMain.DoCloseApp;
begin
```

Tampilkan dialog konfirmasi untuk menutup aplikasi. Jika tombol Ok ditekan, nilai yang dikembalikan fungsi DisplayCloseDlg sama dengan 0.

```
 if DisplayCloseDlg=0 then
 begin
```

Tutup aplikasi.

```
  Timer1.Enabled:=false;
  Close;
```

```
  end;
end;
```

Fungsi berikut merupakan fungsi yang menampilkan dialog konfirmasi penutupan aplikasi. Cara kerjanya mirip dengan fungsi MessageDlg yaitu tidak akan kembali ke rutin pemanggilnya sebelum tobol Ok atau Cancel diklik.

```
function TfrmMain.DisplayCloseDlg:integer;
var menuindex,xx,yy:integer;
    x,y,z:integer;
    clicked:boolean;
    closedlgspr:TSpriteEngine;
begin
 try
```

Kita ciptakan sprite yang akan menampung sprite dialog close. Kita juga membaca file data sprite ini.

```
  closedlgspr:=TSpriteEngine.Create(GraphicEngine);
  closedlgspr.LoadFromFile('closedlg.txt');
  menuindex:=-1;
  xx:=175;
  yy:=150;
```

Posisi dialog kita gambar di koordinat (175,150).

```
  closedlgspr.Position:=SetPosition(xx,yy,0);
```

Kita perlu menghentikan timer agar, ketika dialog muncul pencatatan waktu dihentikan sementara.

```
  timer1.Enabled:=false;
  repeat
```

Karena dialog yang akan kita tampilkan bersifat modal dialog maka kita harus melakukan proses rendering di dalam looping ini. Langkah pertama adalah menimpa gambar yang ada di back buffer dengan gambar background.

```
    back.show;
```

Kita ambil status mouse.

```
    MouseInput.GetDeviceState;
    MouseInput.MousePos(X,y,z);
    MouseXY.X:=MouseXY.X+X;
    MouseXY.Y:=MouseXY.Y+Y;
```

Sel-sel digambar di back buffer.

```
    DisplayCell;
```

```
Gambar dialog close kita letakkan di back buffer.
   closedlgspr.Show;
```

Kita cek posisi mouse apakah tombol Ok atau Cancel ditekan. Tombol apa yang ditekan informasinya disimpan di variabel *menuindex*. Jika mouse berada di atas salah satu tombol, maka tombol akan ditampilkan tersorot (highlight).

```
   DisplayCloseDlgMenu(xx,yy,menuindex,clicked);
```

Sprite kursor mouse kita gambar di back buffer.

```
   DisplayMouse;
```

Langkah terakhir adalah meggambar back buffer ini ke front buffer agar terlihat.

```
   GraphicEngine.Show;
 until ((menuindex=0) or (menuindex=1)) and (clicked);
```

Keluar dari looping jika tombol Ok atau tombol Cancel ditekan.

```
 finally
```

Tombol apa yang ditekan kita kembalikan sebagai nilai fungsi. Sprite dialog kita bebaskan dan pencatatan waktu dikerjakan lagi.

```
  result:=menuindex;
  closedlgspr.Free;
  timer1.Enabled:=true;
 end;
end;
procedure TfrmMain.DisplayCloseDlgMenu(const x, y : integer; var menuindex : integer; var clicked : boolean);
var xx,yy,frame:integer;
begin
  menuindex:=-1;
  if (MouseXY.X>81+x) and (MouseXY.X<139+x) and
     (MouseXY.Y>98+y) and (MouseXY.Y<125+y) then
  begin
    menuindex:=0;
    xx:=81+x;
    yy:=98+y;
    frame:=5;
    Clicked:=MouseInput.MouseDown(0);
  end else
  if (MouseXY.X>158+x) and (MouseXY.X<216+x) and
     (MouseXY.Y>98+y) and (MouseXY.Y<125+y) then
```

```
  begin
    menuindex:=1;
    xx:=158+x;
    yy:=98+y;
    frame:=6;
    Clicked:=MouseInput.MouseDown(0);
  end;
  miscsprite.Position:=setPosition(xx,yy,0);
  miscsprite.FrameNow:=frame;
  miscsprite.Show;
end;
```

Fungsi berikut digunakan untuk mengubah setting suara efek dan musik latar.

```
function TfrmMain.DisplaySoundDlg: integer;
var menuindex,xx,yy,minx,maxx:integer;
    sfxvolx,sfxvoly,sfxbalX,sfxBalY:integer;
    Backvolx,Backvoly,BackbalX,BackBalY:integer;
    x,y,z,i:integer;
    sfxvolclicked,clicked:boolean;
    sfxBalclicked,BackVolclicked,BackBalClicked:boolean;
    sounddlgspr:TSpriteEngine;
    soundOption:TSoundOption;
```

Prosedur berikut digunakan untuk mengubah status variable lain menjadi false bila ada salah satu yang bernilai true. Kita akan menggunakan track bar buatan kita sendiri untuk mengatur setting suara. Setting ini meliputi pan(balance) dan volume.

```
  procedure DisableOtherTrackbar;
  begin
    if sfxvolclicked then
    begin
      sfxBalclicked:=false;
      BackVolclicked:=false;
      BackBalClicked:=false;
    end else
    if sfxBalclicked then
    begin
      sfxvolclicked:=false;
```

```
    BackVolclicked:=false;
    BackBalClicked:=false;
   end else
   if BackVolclicked then
   begin
    sfxBalclicked:=false;
    sfxvolclicked:=false;
    BackBalClicked:=false;
   end else
   if BackBalClicked then
   begin
    BackVolclicked:=false;
    sfxBalclicked:=false;
    sfxvolclicked:=false;
   end;
  end;
begin
 try
```

Kita ciptakan sprite untuk gambar dialog.

```
 sounddlgspr:=TSpriteEngine.Create(GraphicEngine);
 sounddlgspr.LoadFromFile('sounddlg.txt');
```

Kelas TSoundOption adalah kelas yang kita ciptakan untuk melakukan proses interpolasi nilai yang akan mengubah nilai yang ditunjukkan oleh posisi track bar menjadi nilai pan atau volume suara. Demikian pula sebaliknya. Deklarasi kelas ini ada pada file usnd.pas.

```
 SoundOption:=TSoundOption.Create;
 menuindex:=-1;
 xx:=150;
 yy:=60;
```

Kita ambil informasi volume dan pan salah satu efek suara (suara nomer 0) dan musik latar (suara nomer 14) dan kita simpan di properti SFXVolume dan SFXBalance untuk efek suara dan BackMusicVolume dan BackMusicBalance untuk musik latar.

```
 soundfx.EffectNow:=0;
 soundOption.SFXVolume:=soundfx.GetVolume;
 soundOption.SFXBalance:=soundfx.GetPan;
```

```
soundfx.EffectNow:=14;
soundOption.BackMusicVolume:=soundfx.GetVolume;
soundOption.BackMusicBalance:=soundfx.GetPan;
```

MinX dan MaxX berisi posisi minimum dan maksimum track bar.

```
minx:=xx+27;
maxx:=xx+250;
SoundOption.SFXBalanceLeftPos:=minx;
SoundOption.SFXBalanceRightPos:=maxx;
SoundOption.BackMusicBalanceLeftPos:=minx;
SoundOption.BackMusicBalanceRightPos:=maxx;
SoundOption.SFXVolumeMinPos:=minx;
SoundOption.SFXVolumeMaxPos:=maxx;
SoundOption.BackMusicVolumeMinPos:=minx;
SoundOption.BackMusicVolumeMaxPos:=maxx;
```

Tipe interpolasi adalah interpolasi untuk mendapatkan posisi track bar dari data volume dan balance saat ini. *Calculate* dipanggil untuk mulai proses perhitungan.

```
soundOption.InterpolateType:=intNonActual;
soundOption.Calculate;
```

Posisi track bar yang didapatkan melalui interpolasi, kita simpan di *sfxBalX*, *sfxVolX*, *BackBalX* dan *BackVolX*.

```
sfxBalX:=SoundOption.SFXBalancePos;
sfxVolX:=SoundOption.SFXVolumePos;
BackBalX:=SoundOption.BackMusicBalancePos;
BackVolX:=SoundOption.BackMusicVolumePos;
sfxVolY:=55+yy;
BackVolY:=110+yy;
sfxBalY:=185+yy;
BackBalY:=225+yy;
```

Interpolasi yang berikutnya kita kerjakan adalah interpolasi untuk mengubah nilai yang ditunjuk oleh posisi track bar menjadi nilai balance atau volume yang sesuai.

```
soundOption.InterpolateType:=intActual;
sounddlgspr.Position:=SetPosition(xx,yy,0);
timer1.Enabled:=false;
repeat
```

```
back.show;
MouseInput.GetDeviceState;
MouseInput.MousePos(X,y,z);
MouseXY.X:=MouseXY.X+X;
MouseXY.Y:=MouseXY.Y+Y;
DisplayCell;
sounddlgspr.Show;
```

Kita lakukan pengecekan posisi kursor mouse serta penekanan tombol mouse.

```
DisplaySoundDlgMenu(xx, yy, menuindex, clicked, SfxVolclicked, BackVolclicked, sfxBalclicked, BackBalclicked);
```

Jika ada track bar yang sedang digeser-geser maka non-aktifkan track bar yang lain.

```
DisableOtherTrackBar;
```

Gambar track bar volume efek suara.

```
DisplaySoundDlgTrackbar(minx, maxx, sfxVolX, sfxVolY, SfxVolclicked, BackVolclicked, sfxBalclicked, BackBalclicked);
```

Gambar track bar volume musik latar.

```
DisplaySoundDlgTrackbar(minx, maxx, BackVolX, BackVolY, BackVolclicked, SfxVolclicked, sfxBalclicked, BackBalclicked);
```

Gambar track bar balance efek suara.

```
DisplaySoundDlgTrackbar(minx, maxx, sfxBalX, sfxBalY, sfxBalclicked, SfxVolclicked, BackVolclicked, BackBalclicked);
```

Gambar track bar balance musik latar.

```
DisplaySoundDlgTrackbar(minx, maxx, BackBalX, BackBalY, BackBalclicked, SfxVolclicked, BackVolclicked, sfxBalclicked);
```

```
DisplayMouse;
GraphicEngine.Show;
```

Hitung nilai volume dan balance sesungguhnya.

```
SoundOption.SFXBalancePos:=SFXBalX;
SoundOption.SFXVolumePos:=sfxVolX;
SoundOption.BackMusicBalancePos:=BackBalX;
SoundOption.BackMusicVolumePos:=BackVolX;
soundOption.Calculate;
```

Untuk semua suara kecuali suara nomer 14 (nomer 14 adalah musik latar), kita ubah nilai volumenya sesuai nilai yang baru saja dihitung. Suara nomer 14 nilai volume dan balance yang kita ambil dari properti *BackMusicBalance* dan *BackMusicVolume*.

```
for i:=0 to soundfx.Count-1 do
```

```
   begin
    if i<>14 then
    begin
     soundfx.EffectNow:=i;
     Soundfx.SetVolume(soundOption.SFXVolume);
     soundfx.SetPan(soundOption.SFXBalance);
    end else
    begin
     soundfx.EffectNow:=i;
     Soundfx.SetVolume(soundOption.BackMusicVolume);
     soundfx.SetPan(soundOption.BackMusicBalance);
    end;
   end;
  until ((menuindex=0) or (menuindex=1)) and (clicked);
```

Jika tombol Ok atau Cancel diklik, kita tutup dialog suara.

```
 finally
  result:=menuindex;
  sounddlgspr.Free;
  soundOption.Free;
  timer1.Enabled:=true;
 end;
end;
procedure TfrmMain.DisplaySoundDlgTrackBar(const minx, maxx: integer; var xx, yy: integer; var
_clicked:boolean; click1, click2, click3:boolean);
var widt,heig:integer;
begin
  miscSprite.FrameNow:=9;
  widt:=miscsprite.Width;
  heig:=miscsprite.Height;
  if (_Clicked)or((MouseXY.X>xx) and (MouseXY.X<xx+widt) and
    (MouseXY.Y>yy) and (MouseXY.Y<yy+heig)) then
  begin
```

Jika mouse diklik dan tidak ada track bar lain yang sedang diklik, kita ubah nilai yang akan dikembalikan *_clicked* menjadi true.

```
   _clicked:=MouseInput.MouseDown(0) and
            (not click1)and
```

```
        (not click2)and
        (not click3);
   if _clicked then
   if (xx>=minx) and (xx<=maxX) then
   begin
```

Jika _clicked bernilai true maka kita posisi track bar kita ubah sesuai posisi mouse dikurangi setengah lebar sprite. Jika koordinat horizontalnya melebihi *MaxX* kita isi *xx* sama dengan maxX, jika kurang dari *MinX* kita isi xx sama dengan MinX

```
     xx:=mousexy.X-(widt shr 1);
     if xx<minx then xx:=minX;
     if xx>maxx then xx:=maxX;
   end;
  end;
  miscsprite.Position:=SetPosition(xx,yy,0);
  miscsprite.Show;
end;
procedure TfrmMain.DisplaySoundDlgMenu(const x, y: integer;
  var menuindex: integer; var clicked: boolean;click1,click2,click3,click4:boolean);
var xx,yy:integer;
begin
 clicked:=false;
```

Jika track bar-track bar tidak ada yang sedang diklik, kita lakukan pengecekan apakah posisi kursor mouse berada diatas tombol Ok atau Cancel.

```
 if (not click1)and (not click2)and
    (not click3)and (not click4) then
 begin
  if (MouseXY.X>88+x) and (MouseXY.X<124+x) and
     (MouseXY.Y>264+y) and (MouseXY.Y<287+y) then
  begin
    menuindex:=0;
    xx:=88+x;
    yy:=264+y;
    miscsprite.FrameNow:=7;
    Clicked:=MouseInput.MouseDown(0);
    miscsprite.Position:=setPosition(xx,yy,0);
    miscsprite.Show;
```

```
  end else
  if (MouseXY.X>153+x) and (MouseXY.X<225+x) and
     (MouseXY.Y>263+y) and (MouseXY.Y<292+y) then
  begin
    menuindex:=1;
    xx:=153+x;
    yy:=263+y;
    miscsprite.FrameNow:=8;
    Clicked:=MouseInput.MouseDown(0);
    miscsprite.Position:=setPosition(xx,yy,0);
    miscsprite.Show;
  end;
 end;
end;
procedure TfrmMain.DoSoundDlg;
begin
  if DisplaySoundDlg=0 then
  begin
  end;
end;
```

Fungsi *DisplayPauseDlg* dan *DisplayPauseDlgMenu* digunakan untuk menampilkan dialog pause. Fungsi *DisplayWinDlg* dan *DisplayWinDlgMen*u digunakan untuk menampilkan dialog saat pemain berhasil menyelesaikan permainan. Fungsi *DisplayCredit* dan *DisplayCreditMenu* digunakan untuk menampilkan dialog kredit. Fungsi *DisplayFileNotFound* dan *DisplayFileNotFoundMenu* digunakan untuk menampilkan dialog file saved game tidak berhasil ditemukan. Semua prosesnya mirip dengan proses menampilkan dialog-dialog yang lain, sehingga penulis rasa tidak perlu dijelaskan lagi.

```
function TfrmMain.DisplayPauseDlg: integer;
var menuindex,xx,yy:integer;
    x,y,z:integer;
    clicked:boolean;
    pausedlgspr:TSpriteEngine;
begin
 try
  pausedlgspr:=TSpriteEngine.Create(GraphicEngine);
  pausedlgspr.LoadFromFile('pausedlg.txt');
```

```
  menuindex:=-1;
  xx:=150;
  yy:=150;
  pausedlgspr.Position:=SetPosition(xx,yy,0);
  timer1.Enabled:=false;
  repeat
    back.show;
    MouseInput.GetDeviceState;
    MouseInput.MousePos(X,y,z);
    MouseXY.X:=MouseXY.X+X;
    MouseXY.Y:=MouseXY.Y+Y;
    DisplayCell;
    pausedlgspr.Show;
    DisplayPauseDlgMenu(xx,yy,menuindex,clicked);
    DisplayMouse;
    GraphicEngine.Show;
  until ((menuindex=0)) and (clicked);
 finally
  result:=menuindex;
  pausedlgspr.Free;
  timer1.Enabled:=true;
 end;
end;
procedure TfrmMain.DisplayPauseDlgMenu(const x, y: integer;
  var menuindex: integer; var clicked: boolean);
var xx,yy,frame:integer;
begin
  menuindex:=-1;
  if (MouseXY.X>101+x) and (MouseXY.X<205+x) and
     (MouseXY.Y>101+y) and (MouseXY.Y<144+y) then
  begin
    menuindex:=0;
    xx:=101+x;
    yy:=101+y;
    frame:=14;
```

```
    Clicked:=MouseInput.MouseDown(0);
  end;
  miscsprite.Position:=setPosition(xx,yy,0);
  miscsprite.FrameNow:=frame;
  miscsprite.Show;
end;

procedure TfrmMain.DoPauseDlg;
begin
  DisplayPauseDlg;
end;

function TfrmMain.DisplayWinDlg: integer;
var menuindex,xx,yy:integer;
    x,y,z:integer;
    clicked:boolean;
    windlgspr:TSpriteEngine;
begin
 try
  windlgspr:=TSpriteEngine.Create(GraphicEngine);
  windlgspr.LoadFromFile('windlg.txt');
  menuindex:=-1;
  xx:=150;
  yy:=150;
  windlgspr.Position:=SetPosition(xx,yy,0);
  timer1.Enabled:=false;
  repeat
    back.show;
    MouseInput.GetDeviceState;
    MouseInput.MousePos(X,y,z);
    MouseXY.X:=MouseXY.X+X;
    MouseXY.Y:=MouseXY.Y+Y;
    DisplayCell;
    windlgspr.Show;
    DisplaywinDlgMenu(xx,yy,menuindex,clicked);
```

```
    DisplayMouse;
    GraphicEngine.Show;
  until ((menuindex=0)) and (clicked);
 finally
  result:=menuindex;
  windlgspr.Free;
  timer1.Enabled:=true;
 end;
end;

procedure TfrmMain.DisplayWinDlgMenu(const x, y: integer;
  var menuindex: integer; var clicked: boolean);
var xx,yy,frame:integer;
begin
  menuindex:=-1;
  if (MouseXY.X>9+x) and (MouseXY.X<67+x) and
     (MouseXY.Y>113+y) and (MouseXY.Y<142+y) then
  begin
    menuindex:=0;
    xx:=9+x;
    yy:=113+y;
    frame:=15;
    Clicked:=MouseInput.MouseDown(0);
  end;
  miscsprite.Position:=setPosition(xx,yy,0);
  miscsprite.FrameNow:=frame;
  miscsprite.Show;
end;

procedure TfrmMain.DoWinDlg;
begin
  DisplayWinDlg;
end;

function TfrmMain.DisplayCredit: integer;
```

```
var menuindex,xx,yy:integer;
    x,y,z:integer;
    clicked:boolean;
    creditspr:TSpriteEngine;
begin
 try
  creditspr:=TSpriteEngine.Create(GraphicEngine);
  creditspr.LoadFromFile('credit.txt');
  menuindex:=-1;
  xx:=150;
  yy:=100;
  creditspr.Position:=SetPosition(xx,yy,0);
  timer1.Enabled:=false;
  repeat
    back.show;
    MouseInput.GetDeviceState;
    MouseInput.MousePos(X,y,z);
    MouseXY.X:=MouseXY.X+X;
    MouseXY.Y:=MouseXY.Y+Y;
    DisplayCell;
    creditspr.Show;
    DisplayCreditMenu(xx,yy,menuindex,clicked);
    DisplayMouse;
    GraphicEngine.Show;
  until ((menuindex=0)) and (clicked);
 finally
  result:=menuindex;
  creditspr.Free;
  timer1.Enabled:=true;
 end;
end;

procedure TfrmMain.DisplayCreditMenu(const x, y: integer;
  var menuindex: integer; var clicked: boolean);
var xx,yy,frame:integer;
```

```
begin
  menuindex:=-1;
  if (MouseXY.X>105+x) and (MouseXY.X<211+x) and
     (MouseXY.Y>237+y) and (MouseXY.Y<287+y) then
  begin
    menuindex:=0;
    xx:=105+x;
    yy:=237+y;
    frame:=19;
    Clicked:=MouseInput.MouseDown(0);
  end;
  miscsprite.Position:=setPosition(xx,yy,0);
  miscsprite.FrameNow:=frame;
  miscsprite.Show;
end;

procedure TfrmMain.DoCredit;
begin
  DisplayCredit;
end;

function TfrmMain.DisplayFileNotFound: integer;
var menuindex,xx,yy:integer;
    x,y,z:integer;
    clicked:boolean;
    filespr:TSpriteEngine;
begin
 try
  filespr:=TSpriteEngine.Create(GraphicEngine);
  filespr.LoadFromFile('filenotfound.txt');
  menuindex:=-1;
  xx:=250;
  yy:=150;
  filespr.Position:=SetPosition(xx,yy,0);
  timer1.Enabled:=false;
```

```
  repeat
    back.show;
    MouseInput.GetDeviceState;
    MouseInput.MousePos(X,y,z);
    MouseXY.X:=MouseXY.X+X;
    MouseXY.Y:=MouseXY.Y+Y;
    DisplayCell;
    filespr.Show;
    DisplayFileNotFoundMenu(xx,yy,menuindex,clicked);
    DisplayMouse;
    GraphicEngine.Show;
  until ((menuindex=0)) and (clicked);
 finally
  result:=menuindex;
  filespr.Free;
  timer1.Enabled:=true;
 end;
end;

procedure TfrmMain.DisplayFileNotFoundMenu(const x, y: integer;
  var menuindex: integer; var clicked: boolean);
var xx,yy,frame:integer;
begin
  menuindex:=-1;
  if (MouseXY.X>44+x) and (MouseXY.X<155+x) and
     (MouseXY.Y>56+y) and (MouseXY.Y<94+y) then
  begin
    menuindex:=0;
    xx:=44+x;
    yy:=56+y;
    frame:=20;
    Clicked:=MouseInput.MouseDown(0);
  end;
  miscsprite.Position:=setPosition(xx,yy,0);
  miscsprite.FrameNow:=frame;
```

```
  miscsprite.Show;
end;


procedure TfrmMain.DoFileNotFound;
begin
  DisplayFileNotFound;
end;
```

Prosedur Win dijalankan saat event OnWin milik SlidePuzzle terjadi.

```
procedure TfrmMain.Win(sender: TObject);
begin
```

Tampilkan dialog informasi pemain telah berhasil menyelesaikan permainan.

```
  DoWinDlg;
```

Jika pemain menutup dialog ini, kita acak ulang isi sel-sel. Kita isi started menjadi false yang berarti permainan di reset.

```
  SlidePuzzle.Setup;
  started:=false;
  TimeTick:=0;
end;
```

DisplayTime digunakan untuk menampilkan informasi waktu yang telah dilampaui pemain.

```
procedure TfrmMain.DisplayTime;
var txt:string;
begin
  txt:=TickToStr(TimeTick);
//  txtsprite.Position:=SetPosition(480,450,0);
  txtsprite.WriteString(480,450,txt);
end;
```

*TickToSt*r adalah fungsi yang kita gunakan untuk mengubah nilai waktu menjadi string.

```
function TfrmMain.TickToStr(const ticks: integer): string;
var h,m,s:integer;
begin
```

Karena ticks bersatuan milidetik, kita perlu mengubahnya menjadi berformat jam:menit:detik nilai milidetik-nya kita abaikan.

```
  s:=(ticks div 1000) mod 60;
```

```
  m:=(ticks div 60000) mod 60;
  h:=ticks div 3600000;
```

Kita konversi h, m, s menjadi string dengan IntToStr. IntToStr di deklarasikan di unit sysutils.pas.

```
  result:='Time '+intTostr(h)+':'+IntToStr(m)+':'+intToStr(s);
end;
```

Fungsi berikut ini digunakan untuk menampilkan scroll bar vertikal pada dialog Save.

```
procedure TfrmMain.DisplaySaveLoadDlgTrackBar(const minx,  maxx, afilecount, maxFiledisplay : integer; var xx, yy, indexfirst, indexLast: integer; var aclicked:boolean);
var exceed,i,tot,widt,heig:integer;
begin
  exceed:=Afilecount-maxFileDisplay;
  if exceed>0 then
  begin
    miscSprite.FrameNow:=9;
    widt:=miscsprite.Width;
    heig:=miscsprite.Height;
    if (aClicked)or((MouseXY.X>xx) and (MouseXY.X<xx+widt) and
      (MouseXY.Y>yy) and (MouseXY.Y<yy+heig)) then
    begin
      aclicked:=MouseInput.MouseDown(0);
      if aclicked then
      if (yy>=minx) and (yy<=maxx) then
      begin
        yy:=mousexy.y-(heig shr 1);
        if yy<minx then yy:=minX;
        if yy>maxx then yy:=maxX;
        tot:=(maxx-minx)div exceed;
        for i:=0 to exceed-1 do
        begin
          if abs(yy-(i*tot+minx))<5 then
          begin
            indexFirst:=i;
            indexLast:=indexFirst+maxFileDisplay;
            break;
```

```
        end;
       end;
      end;
    end;
    miscsprite.Position:=SetPosition(xx,yy,0);
    miscsprite.Show;
  end;
end;
```

Fungsi *DisplaySaveDlg* digunakan untuk melakukan proses save game ke file.

```
function TfrmMain.DisplaySaveDlg: integer;
var menuindex,xx,yy,xtrck,ytrck:integer;
    x,y,z,len,mm:integer;
    clicked,trckbar:boolean;
    savedlgspr:TSpriteEngine;
    astr:string;
    filePuzzle:TFilePuzzle;
    afilelist:TFileList;
    highlightFile:TFillRect;
    StartIndex,LastIndex:integer;
begin
 try
  savedlgspr:=TSpriteEngine.Create(GraphicEngine);
  savedlgspr.LoadFromFile('savedlg.txt');
```

Selain sprite kita juga menciptakan instance kelas TFillRect yang akan kita gunakan untuk menampilkan informasi file yang sedang disorot.

```
  highLightFile:=TFillRect.Create;
  highLightFile.color:=$ff0000ff;
  highLightFile.Surface:=GraphicEngine.BackSurface;
```

Instance kelas TFileList kita ciptakan. TFileList kita gunakan untuk memfilter file-file yang ada pada suatu direktori. Deklarasi kelas ini ada pada file ufilelist.pas. Penulis tidak menjelaskan kelas ini lebih lanjut karena proses yang dikerjakan kelas ini sederhana.

```
  afileList:=TFileList.Create;
```

Kita hanya akan menampilkan file dengan ekstensi SAV. Extensi ini adalah ekstensi file saved game kita.. Formatnya cukup sederhana, pembaca dapat melihat

source code kelas yang digunakan untuk membaca dan menulis file berformat ini pada file uGameLogic.pas. Kelasnya bernama TFilePuzzle.

```
afileList.FilePath:=SaveDir+'*.sav';
afileList.Build;
```

Kita buat daftar semua file yang berekstensi SAV yang terletak di direktori SaveDir.

```
menuindex:=-1;
startindex:=0;
lastIndex:=0;
xx:=175;
yy:=110;
xtrck:=xx+202;
ytrck:=yy+15;
trckbar:=false;
savedlgspr.Position:=SetPosition(xx,yy,0);
timer1.Enabled:=false;
filename:='';
```

Kita perlu melepaskan akses keyboard karena kita tidak akan menggunakan DirectInput untuk melakukan proses input nama file. Untuk keperluan tersebut kita akan menggunakan event OnKeyDown milik form utama.

```
KeyboardInput.UnAcquire;
repeat
  back.show;
  MouseInput.GetDeviceState;
  MouseInput.MousePos(X,y,z);
  MouseXY.X:=MouseXY.X+X;
  MouseXY.Y:=MouseXY.Y+Y;
  DisplayCell;
  savedlgspr.Show;
```

Karena kita perlu merespon adanya event OnKeyDown, maka kita perlu membiarkan aplikasi memproses pesan-pesan yang diterimanya. Jika tidak, kita tidak akan dapat menerima pesan event OnKeyDown.

```
  Application.ProcessMessages;
```

Ketika pemain mengetikkan suatu karakter maka apa yang ada pada Filename kita tampilkan. Jika panjang nya melebihi 17 karakter, kita tampilkan hanya 17 karakter paling belakang.

```
  len:=Length(filename);
```

```
if Len>17 then
astr:=Copy(Filename,len-17,17) else
 astr:=filename;
txtsprite.WriteString(xx+20,yy+230,astr);
```

Prosedur *DisplaySaveDlgMenu* kita gunakan untuk mendeteksi tombol-tombol apa yang ditekan, nama file yang diklik atau mendeteksi drag scroll bar

```
DisplaySaveDlgMenu(xx,yy,menuindex,clicked,highlightfile,trckbar);
if (menuindex>=2) and (menuIndex<=20) and (clicked) then
begin
 mm:=StartIndex+(menuIndex-2);
 if (mm>=0) and (mm<AFileList.Count) then
 begin
  filename:=AFileList[StartIndex+(menuIndex-2)];
  filename:=changeFileExt(filename,");
 end;
end;
if startIndex+AFileList.Count<=18 then
 LastIndex:=AFileList.Count-1 else
 LastIndex:=StartIndex+18;
```

DisplayFileList berfungsi menampilkan daftar nama-nama file saved game. Nama-nama ini disimpan di AfileList dan diindeks oleh startIndex dan LastIndex. Maksimum nama file yang ditampilkan pada satu saat adalah 18.

```
DisplayFileList(xx,yy,AFileList,startIndex,LastIndex,txtSprite);

DisplaySaveLoadDlgTrackBar(yy+15,yy+190,afilelist.count,18,xtrck,ytrck,startIndex,LastIndex,trckbar);
DisplayMouse;
GraphicEngine.Show;
until (((menuindex=0)and(Filename<>")) or (menuindex=1)) and (clicked);
```

Jika tombol Ok diklik maka FileName harus berisi nama file.

```
if menuindex=0 then
begin
 try
```

Jika tombol Ok diklik, maka kita proses data untuk disimpan ke file. Untuk maksud tersebut kita ciptakan instance kelas TFilePuzzle.

```
  FilePuzzle:=TFilePuzzle.Create;
  FilePuzzle.Cells:=SlidePuzzle.Cells;
```

```
    FilePuzzle.Row:=MaxCell;
    FilePuzzle.Col:=MaxCell;
    filename:=lowercase(filename);
    if pos('.sav',filename)=0 then
    filename:=SaveDir+filename+'.sav' else
    filename:=SaveDir+filename;
    FilePuzzle.SaveToFile(filename);
   finally
```

Proses penyimpanan selesai

```
    FilePuzzle.Free;
   end;
  end;
finally
```

Kita kembalikan akses keyboard dan tombol apa yang ditekan kita kembalikan sebagai nilai fungsi. Instance-instance yang tidak kita perlukan kita bebaskan.

```
  KeyboardInput.Acquire;
  result:=menuindex;
  savedlgspr.Free;
  highLightfile.Free;
  afilelist.Free;
  timer1.Enabled:=true;
 end;
end;


procedure TfrmMain.DisplaySaveDlgMenu(const x, y: integer;
  var menuindex: integer; var clicked: boolean;highLightFile:TFillRect;const trackbarclicked:boolean);
var xx,yy,frame:integer;
begin
  menuindex:=-1;
  if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
     (MouseXY.Y>y+15) and (MouseXY.Y<=25+y) then
  begin
    menuindex:=2;
    xx:=20+x;
    yy:=15+y;
```

```
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+25) and (MouseXY.Y<=35+y) then
begin
  menuindex:=3;
  xx:=20+x;
  yy:=25+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+35) and (MouseXY.Y<=45+y) then
begin
  menuindex:=4;
  xx:=20+x;
  yy:=35+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+45) and (MouseXY.Y<=55+y) then
begin
  menuindex:=5;
  xx:=20+x;
  yy:=45+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+55) and (MouseXY.Y<=65+y) then
begin
  menuindex:=6;
  xx:=20+x;
  yy:=55+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
```

```
  (MouseXY.Y>y+65) and (MouseXY.Y<=75+y) then
begin
  menuindex:=7;
  xx:=20+x;
  yy:=65+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
  (MouseXY.Y>y+75) and (MouseXY.Y<=85+y) then
begin
  menuindex:=8;
  xx:=20+x;
  yy:=75+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
  (MouseXY.Y>y+85) and (MouseXY.Y<=95+y) then
begin
  menuindex:=9;
  xx:=20+x;
  yy:=85+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
  (MouseXY.Y>y+95) and (MouseXY.Y<=105+y) then
begin
  menuindex:=10;
  xx:=20+x;
  yy:=95+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
  (MouseXY.Y>y+105) and (MouseXY.Y<=115+y) then
begin
  menuindex:=11;
```

```
  xx:=20+x;
  yy:=105+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+115) and (MouseXY.Y<=125+y) then
begin
  menuindex:=12;
  xx:=20+x;
  yy:=115+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+125) and (MouseXY.Y<=135+y) then
begin
  menuindex:=13;
  xx:=20+x;
  yy:=125+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+135) and (MouseXY.Y<=145+y) then
begin
  menuindex:=14;
  xx:=20+x;
  yy:=135+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+145) and (MouseXY.Y<=155+y) then
begin
  menuindex:=15;
  xx:=20+x;
  yy:=145+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
```

```
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+155) and (MouseXY.Y<=165+y) then
begin
  menuindex:=16;
  xx:=20+x;
  yy:=155+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+165) and (MouseXY.Y<=175+y) then
begin
  menuindex:=17;
  xx:=20+x;
  yy:=165+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+175) and (MouseXY.Y<=185+y) then
begin
  menuindex:=18;
  xx:=20+x;
  yy:=175+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+185) and (MouseXY.Y<=195+y) then
begin
  menuindex:=19;
  xx:=20+x;
  yy:=185+y;
  Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+195) and (MouseXY.Y<=205+y) then
```

```
  begin
    menuindex:=20;
    xx:=20+x;
    yy:=195+y;
    Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
  end else
  if (MouseXY.X>211+x) and (MouseXY.X<=298+x) and
     (MouseXY.Y>203+y) and (MouseXY.Y<=230+y) then
  begin
    menuindex:=0;
    xx:=211+x;
    yy:=203+y;
    frame:=16;
    Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
  end;
  if (MouseXY.X>211+x) and (MouseXY.X<=298+x) and
     (MouseXY.Y>230+y) and (MouseXY.Y<=257+y) then
  begin
    menuindex:=1;
    xx:=211+x;
    yy:=230+y;
    frame:=17;
    Clicked:=MouseInput.MouseDown(0) and (not trackbarClicked);
  end;
  if ((menuindex=0) or (menuindex=1))and (not trackbarClicked) then
  begin
    miscsprite.Position:=setPosition(xx,yy,0);
    miscsprite.FrameNow:=frame;
    miscsprite.Show;
  end else
  if (MenuIndex>=2)and(MenuIndex<=20)and (not trackbarClicked) then
    DisplayFileHighLight(xx,yy,HighLightFile);
end;

procedure TfrmMain.DoSaveDlg;
```

```
begin
  DisplaySaveDlg;
end;
```

Pada saat kita hendak mengisikan nama file, DirectInput tidak kita pergunakan, karena lebih mudah menggunakan event OnKeyDown untuk proses inputnya. Pada saat event OnKeyDown terjadi, kita cek tombol apa yang ditekan, kita tambahkan karakter yang ditekan ini ke variabel Filename. Hal inilah yang menyebabkan Filename lingkupnya harus dibuat global.

```
procedure TfrmMain.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var str:string;
begin
  case Key of
    VK_BACK:Delete(Filename,Length(filename),1);
    Ord('A')..Ord('Z'):begin
                         str:='';
                        str:=str+char(key);
                        if not(ssShift in Shift) then
                           str:=LowerCase(str)   else
                         if (ssShift in Shift) then
                           str:=UpperCase(str);
                       Filename:=Filename+(str[1]);
                     end;
    Ord('0')..Ord('9'):  Filename:=Filename+Char(Key);
  end;
end;
```

Prosedur berikut digunakan untuk membaca isi file saved game. Hasil pembacaannya disimpan di SlidePuzzle. Prosedur ini dipanggil saat menampilkan dialog load.

```
procedure TfrmMain.OpenSavedData(const filename: string);
var filePuzzle:TFilePuzzle;
begin
  try
    FilePuzzle:=TFilePuzzle.Create;
    FilePuzzle.LoadFromFile(Filename);
    SlidePuzzle.Cells:=FilePuzzle.Cells;
```

```
   started:=false;
   TimeTick:=0;
   PlayEffect(6);
  finally
   FilePuzzle.Free;
  end;
end;
```

Fungsi ini mirip dengan fungsi DisplaySaveDlg sehingga Penulis merasa tidak perlu lagi dijelaskan. Perbedaannya hanya terletak pada proses open saved game. Dimana kita memanggil prosedur OpenSavedData saat pemain mengklik Ok.

```
function TfrmMain.DisplayLoadDlg: integer;
var menuindex,xx,yy,xtrck,ytrck:integer;
   x,y,z,len,mm:integer;
   clicked,trckBar:boolean;
   loaddlgspr:TSpriteEngine;
   astr:string;
   afilelist:TFileList;
   highlightFile:TFillRect;
   StartIndex,LastIndex:integer;
begin
 try
  loaddlgspr:=TSpriteEngine.Create(GraphicEngine);
  loaddlgspr.LoadFromFile('loaddlg.txt');
  highLightFile:=TFillRect.Create;
  highLightFile.color:=$ff0000ff;
  highLightFile.Surface:=GraphicEngine.BackSurface;
  afileList:=TFileList.Create;
  afileList.FilePath:=SaveDir+'*.sav';
  afileList.Build;
  menuindex:=-1;
  startIndex:=0;
  LastIndex:=0;
  xx:=175;
  yy:=110;
  xtrck:=xx+202;
  ytrck:=yy+15;
```

```
  trckBar:=false;
  loaddlgspr.Position:=SetPosition(xx,yy,0);
  timer1.Enabled:=false;
  filename:='';
  KeyboardInput.UnAcquire;
  StartIndex:=0;
  LastIndex:=0;
  repeat
    back.show;
    MouseInput.GetDeviceState;
    MouseInput.MousePos(X,y,z);
    MouseXY.X:=MouseXY.X+X;
    MouseXY.Y:=MouseXY.Y+Y;
    DisplayCell;
    loaddlgspr.Show;
    Application.ProcessMessages;
//    txtsprite.Position:=SetPosition(xx+20,yy+230,0);
    len:=Length(filename);
    if Len>17 then
    astr:=Copy(Filename,len-17,17) else
      astr:=filename;
    txtsprite.WriteString(xx+20,yy+230,astr);
    DisplayLoadDlgMenu(xx,yy,menuindex,clicked,HighLightFile,trckBar);
    if (menuindex>=2) and (menuIndex<=20) and (clicked) then
    begin
      mm:=StartIndex+(menuIndex-2);
      if (mm>=0) and (mm<AFileList.Count) then
      begin
        filename:=AFileList[StartIndex+(menuIndex-2)];
        filename:=changeFileExt(filename,'');
      end;
    end;
    if startIndex+AFileList.Count<=18 then
     LastIndex:=AFileList.Count-1 else
     LastIndex:=StartIndex+18;
```

```
   DisplayFileList(xx,yy,AFileList,startIndex,LastIndex,txtSprite);

DisplaySaveLoadDlgTrackBar(yy+15,yy+190,afilelist.count,18,xtrck,ytrck,startIndex,LastIndex,trckbar);
   DisplayMouse;
   GraphicEngine.Show;
  until (((menuindex=0)and
        (Filename<>''))or
      (menuindex=1)) and (clicked);


  if menuindex=0 then
  begin
```

Tes apakah file yang akan kita baca ada pada direktori. Jika ada, lanjutkan dengan proses pembacaan. Jika tidak ada kita tampilkan informasi file tidak berhasil ditemukan.

```
   If FileExists(SaveDir+Filename+'.sav') then
   OpenSavedData(SaveDir+Filename+'.sav') else
    DoFileNotFound;
  end;
 finally
  KeyboardInput.Acquire;
  result:=menuindex;
  loaddlgspr.Free;
  highLightFile.Free;
  afilelist.Free;
  timer1.Enabled:=true;
 end;
end;
procedure TfrmMain.DisplayLoadDlgMenu(const x, y: integer;
  var menuindex: integer; var clicked: boolean;highlightfile:TFillRect;const trackbarClicked:boolean);
var xx,yy,frame,i,xx2,yy2,chheight:integer;
begin
  menuindex:=-1;
  if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
    (MouseXY.Y>y+15) and (MouseXY.Y<=25+y) then
  begin
```

```
  menuindex:=2;
  xx:=20+x;
  yy:=15+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+25) and (MouseXY.Y<=35+y) then
begin
  menuindex:=3;
  xx:=20+x;
  yy:=25+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+35) and (MouseXY.Y<=45+y) then
begin
  menuindex:=4;
  xx:=20+x;
  yy:=35+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+45) and (MouseXY.Y<=55+y) then
begin
  menuindex:=5;
  xx:=20+x;
  yy:=45+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+55) and (MouseXY.Y<=65+y) then
begin
  menuindex:=6;
  xx:=20+x;
  yy:=55+y;
```

```
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+65) and (MouseXY.Y<=75+y) then
begin
  menuindex:=7;
  xx:=20+x;
  yy:=65+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+75) and (MouseXY.Y<=85+y) then
begin
  menuindex:=8;
  xx:=20+x;
  yy:=75+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+85) and (MouseXY.Y<=95+y) then
begin
  menuindex:=9;
  xx:=20+x;
  yy:=85+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+95) and (MouseXY.Y<=105+y) then
begin
  menuindex:=10;
  xx:=20+x;
  yy:=95+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
```

```
  (MouseXY.Y>y+105) and (MouseXY.Y<=115+y) then
begin
  menuindex:=11;
  xx:=20+x;
  yy:=105+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
  (MouseXY.Y>y+115) and (MouseXY.Y<=125+y) then
begin
  menuindex:=12;
  xx:=20+x;
  yy:=115+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
  (MouseXY.Y>y+125) and (MouseXY.Y<=135+y) then
begin
  menuindex:=13;
  xx:=20+x;
  yy:=125+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
  (MouseXY.Y>y+135) and (MouseXY.Y<=145+y) then
begin
  menuindex:=14;
  xx:=20+x;
  yy:=135+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
  (MouseXY.Y>y+145) and (MouseXY.Y<=155+y) then
begin
  menuindex:=15;
```

```
  xx:=20+x;
  yy:=145+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+155) and (MouseXY.Y<=165+y) then
begin
  menuindex:=16;
  xx:=20+x;
  yy:=155+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+165) and (MouseXY.Y<=175+y) then
begin
  menuindex:=17;
  xx:=20+x;
  yy:=165+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+175) and (MouseXY.Y<=185+y) then
begin
  menuindex:=18;
  xx:=20+x;
  yy:=175+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+185) and (MouseXY.Y<=195+y) then
begin
  menuindex:=19;
  xx:=20+x;
  yy:=185+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
```

```
end else
if (MouseXY.X>x+20) and (MouseXY.X<=196+x) and
   (MouseXY.Y>y+195) and (MouseXY.Y<=205+y) then
begin
  menuindex:=20;
  xx:=20+x;
  yy:=195+y;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>212+x) and (MouseXY.X<=297+x) and
   (MouseXY.Y>204+y) and (MouseXY.Y<=231+y) then
begin
  menuindex:=0;
  xx:=212+x;
  yy:=204+y;
  frame:=18;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end else
if (MouseXY.X>211+x) and (MouseXY.X<=298+x) and
   (MouseXY.Y>230+y) and (MouseXY.Y<=257+y) then
begin
  menuindex:=1;
  xx:=211+x;
  yy:=230+y;
  frame:=17;
  Clicked:=MouseInput.MouseDown(0) and (not TrackBarClicked);
end;
if ((MenuIndex=0)or(MenuIndex=1))and (not TrackBarClicked)then
begin
  miscsprite.Position:=setPosition(xx,yy,0);
  miscsprite.FrameNow:=frame;
  miscsprite.Show;
end else
if (MenuIndex>=2)and(MenuIndex<=20)and (not TrackBarClicked) then
  DisplayFileHighLight(xx,yy,HighLightFile);
```

```
end;
procedure TfrmMain.DoLoadDlg;
begin
  DisplayLoadDlg;
end;
procedure TfrmMain.DisplayFileList(const x,y:integer;afileList: TFileList; const FirstIndex,
  LastIndex: integer;txtSprite:TFontEngine2);
var i,indx:integer;
begin
  if (FirstIndex>=0) and (LastIndex<aFileList.Count) and
     (FirstIndex<=LastIndex) then
  begin
    for i:=FirstIndex to LastIndex do
    begin
      indx:=i-FirstIndex;
      indx:=(indx shl 3)+(indx shl 1);
//      txtsprite.Position:=SetPosition(x+20,y+indx+15,0);
      txtsprite.WriteString(x+20,y+indx+txtsprite.GetCharHeight,ChangeFileExt(AFileList.Strings[i],''));
    end;
  end;
end;
procedure TfrmMain.DisplayFileHighlight(const x, y: integer;
  afillrect: TFillRect);
begin
  afillRect.Rect:=Rect(x,y,x+176,y+txtsprite.GetCharHeight);
  afillRect.Draw;
end;
end.
```

Bab 12
Penutup

Dengan berakhirnya pembahasan proyek game kita maka berakhir juga topik Pemrograman DirectX dengan Delphi, penulis rasa apa yang penulis sampaikan pada beberapa bagian masih kurang detail, selain itu kode program tidak seluruhnya optimal (penulis menulis buku ini saat penulis masih belajar Delphi). Oleh karena itu penulis mengharapkan masukan-masukan dari pembaca guna peningkatan mutu tutorial ini.

Pada kesempatan berikutnya, topik yang akan kita bahas masih berkaitan dengan pemrograman game 2D, namun kita akan beralih menggunakan DirectXGraphic dan DirectXAudio dengan topik-topik bahasan yang tentu saja lebih menarik dan menantang. Selamat membuat game.

Zamrony P Juhara

zamronypj@yahoo.com

http://www.geocities.com/zamronypj

CD-ROM

Instalasi.

Jika di komputer anda terdapat Borland Delphi dan anda berniat mengkompilasi ulang contoh-contoh program yang ada, maka anda dapat langsung mengkopi isi folder *Pemrograman DirectX dengan Delphi* yang ada di CD ke komputer anda. Jangan lupa mengubah atribut seluruh file yang telah anda kopi dari read-only menjadi archive. Hal ini diperlukan agar Delphi dapat mengkompilasi file tersebut dan menciptakan file executablenya.

Jika pembaca tidak memiliki Borland Delphi, dan hanya bermaksud melihat contoh-contoh program ketika sedang dijalankan, anda harus menjalankan file instalasi S*etup.exe*. File ini selain akan menginstall source code dan executable contoh-contoh program, juga akan menginstall file run time library yang diperlukan untuk menjalankan file-file executable tersebut. Proses instalasi cukup mudah, anda hanya perlu mengikuti petunjuk-petunjuk yang ditampilkan selama proses instalasi.