VidWorks Entertainment New Developer Training Program
Tutorial #2

Events

Key Events

A big part of computer programs is accepting input from the user. In our examples so far, we have been missing that--input. Unfortunately, keyboard input isn't quite as simple for applets as it was for command-line applications (and even that was pretty complicated). This section will go over the basics of keyboard input for graphical programs.

Now, we'll need to use an interface called `KeyListener`. This interface has the following method prototypes:

```
void keyPressed(KeyEvent e)
void keyReleased(KeyEvent e)
void keyTyped(KeyEvent e)
```

Basically, you implement the `KeyListener`, and whenever a Key Event happens, the corresponding method will be called. `keyPressed()` and `keyReleased()` should be intuitive. A key is considered "typed" when a key is pressed, released, and generates a printable character, for example 'A'.

Of course, it's not enough to just know that an event happened. In the case of the keyboard, we want to know what the user pressed. The key (no pun intended) is in the `KeyEvent` object that is passed to the listener methods. A call to the method `e.getKeyCode()` will return an integer. Comparing this integer to constants such as `KeyEvent.VK_A`, `KeyEvent.VK_0`, `KeyEvent.VK_SHIFT`, `KeyEvent.VK_CONTROL`, `KeyEvent.VK_LEFT`, and other similar constants will allow us to figure out which key was pressed. (For the whole list of `KeyEvent` constants, look at the API on `java.sun.com`.)

Another useful method is `e.getKeyChar()`, which returns the character generated by a key typed event. (It's not guaranteed to be meaningful for key pressed or key released events.)

A few more details: First, a key event listener needs to be registered to the object it's listening to. To do this, we call the method `addKeyListener(KeyListener listener)` *on* the object that will produce these events and *pass to it* the listener. For example, if `bob` was generating key events and `sarah` was a listener listening for these events, then we would write `bob.addKeyListener(sarah)`.

Second, in order for an object to produce events, it needs to have "focus". In our example above, we would have to write `bob.requestFocus()` so that bob could accept input from the keyboard and generate key events.

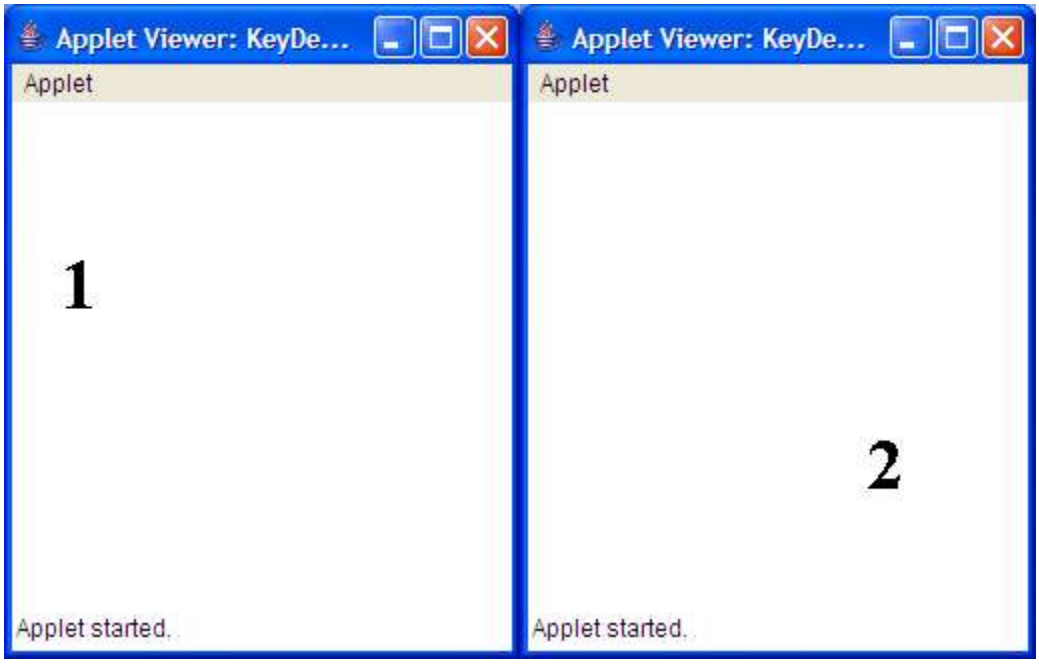
Last, when an event happens, usually, we want this change to be reflected on the screen. However, Java is not smart enough to repaint the applet every time an event happens. (In fact, sometimes we don't want to repaint the screen every time.) In this case, we should call the method `repaint()`, a member function of applet, to repaint the screen for us.

Enough talking, time for an example:

```
/* KeyDemo.java */

import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class KeyDemo extends Applet implements KeyListener {
      static final int SLOW = 2;
      static final int FAST = 10;

      int cx = 100, cy = 100;
      char theKey = '?';
      Font theFont = new Font("Serif", Font.BOLD, 36);
      int dx = SLOW, dy = SLOW;

      public void init() {
            setFont(theFont);
            addKeyListener(this);
            requestFocus();
      }

      public void paint(Graphics g) {
            g.drawString("" + theKey, cx, cy);
      }

      public void keyPressed(KeyEvent e) {
            int code = e.getKeyCode();
            if (code == KeyEvent.VK_CONTROL) {
                  dx = dy = FAST;
            } else if (code == KeyEvent.VK_UP) {
                  cy -= dy;
                  repaint();
            } else if (code == KeyEvent.VK_DOWN) {
                  cy += dy;
                  repaint();
            } else if (code == KeyEvent.VK_LEFT) {
                  cx -= dx;
                  repaint();
            } else if (code == KeyEvent.VK_RIGHT) {
                  cx += dx;
                  repaint();
            }
      }

      public void keyReleased(KeyEvent e) {
            if (e.getKeyCode() == KeyEvent.VK_CONTROL) {
                  dx = dy = SLOW;
            }
      }

      public void keyTyped(KeyEvent e) {
            theKey = e.getKeyChar();
            repaint();
      }
}
```

Note that the line `addKeyListener(this)` basically has our applet registering itself to itself as both the listener *and* event generator.

Anyways, here's some output, as always. In this case, I moved the character up to the upper-left and pressed '1', then moved it to the lower-right and pressed '2'.



Mouse Events

Similarly, there's a `MouseListener` interface, which allows us to listen to mouse events. Its methods are as follows:

`void mousePressed(MouseEvent e)`
`void mouseReleased(MouseEvent e)`
`void mouseClicked(MouseEvent e)` - a mouse press followed by a mouse release
`void mouseEntered(MouseEvent e)` - when the mouse enters the applet
`void mouseExited(MouseEvent e)` - when the mouse leaves the applet

And the class `MouseEvent` has the following useful functions:

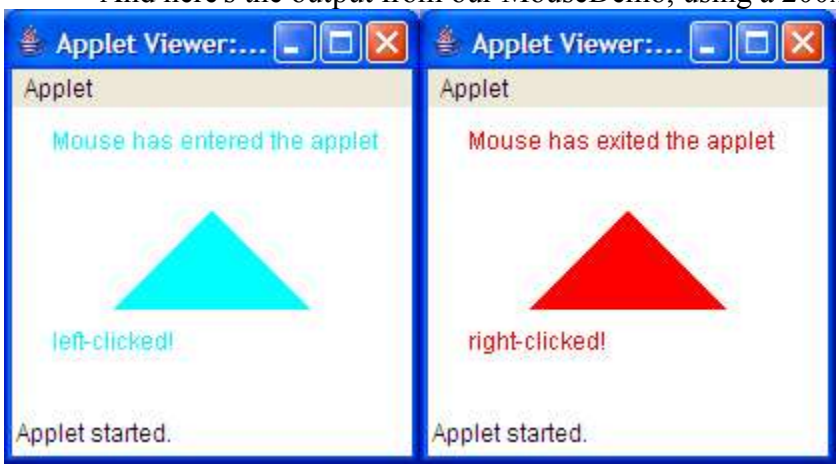
`int getX()` - get the X-coordinate of the mouse event
`int getY()` - get the Y-coordinate of the mouse event
`int getButton()` - get the button used for the mouse event; can return `MouseEvent.NOBUTTON`, `MouseEvent.BUTTON1` (left-click), `MouseEvent.BUTTON2` (middle button), and `MouseEvent.BUTTON3` (right-click)

Note that these mouse button constants are for a 3-button mouse. It is not known whether these constants are correct for a 2-button or (god forbid) 1-button mouse, due to lack of hardware to test with.

Now, for an example:

```
/* MouseDemo.java */

import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class MouseDemo extends Applet implements MouseListener {
      int x[] = {50, 100, 150};
      int y[] = {100, 50, 100};
      Polygon p = new Polygon(x, y, 3);
      String disp1 = "", disp2 = "";

      public void init() {
            addMouseListener(this);
      }

      public void paint(Graphics g) {
            g.drawString(disp1, 20, 20);
            g.fillPolygon(p);
            g.drawString(disp2, 20, 120);
      }

      public void mousePressed(MouseEvent e) {
            if (p.contains(e.getX(), e.getY())) {
                  setForeground(Color.cyan);
                  repaint();
            }
      }

      public void mouseReleased(MouseEvent e) {
            if (p.contains(e.getX(), e.getY())) {
                  setForeground(Color.red);
                  repaint();
            }
      }

      public void mouseClicked(MouseEvent e) {
            int button = e.getButton();
            if (button == MouseEvent.BUTTON1) {
                  disp2 = "left-clicked!";
            } else if (button == MouseEvent.BUTTON3) {
                  disp2 = "right-clicked!";
            }
      }
      public void mouseEntered(MouseEvent e) {
            disp1 = "Mouse has entered the applet";
            repaint();
      }

      public void mouseExited(MouseEvent e) {
            disp1 = "Mouse has exited the applet";
            repaint();
      }
}
```

And here's the output from our MouseDemo, using a 200x150 applet:



Mouse Motion

But mouse clicks and releases aren't the only thing we can detect. We can also generate events whenever the mouse moves. However, mouse motion events are generated so quickly that the Java creators decided to create an extra interface for that, called `MouseMotionListener`, with the following methods:

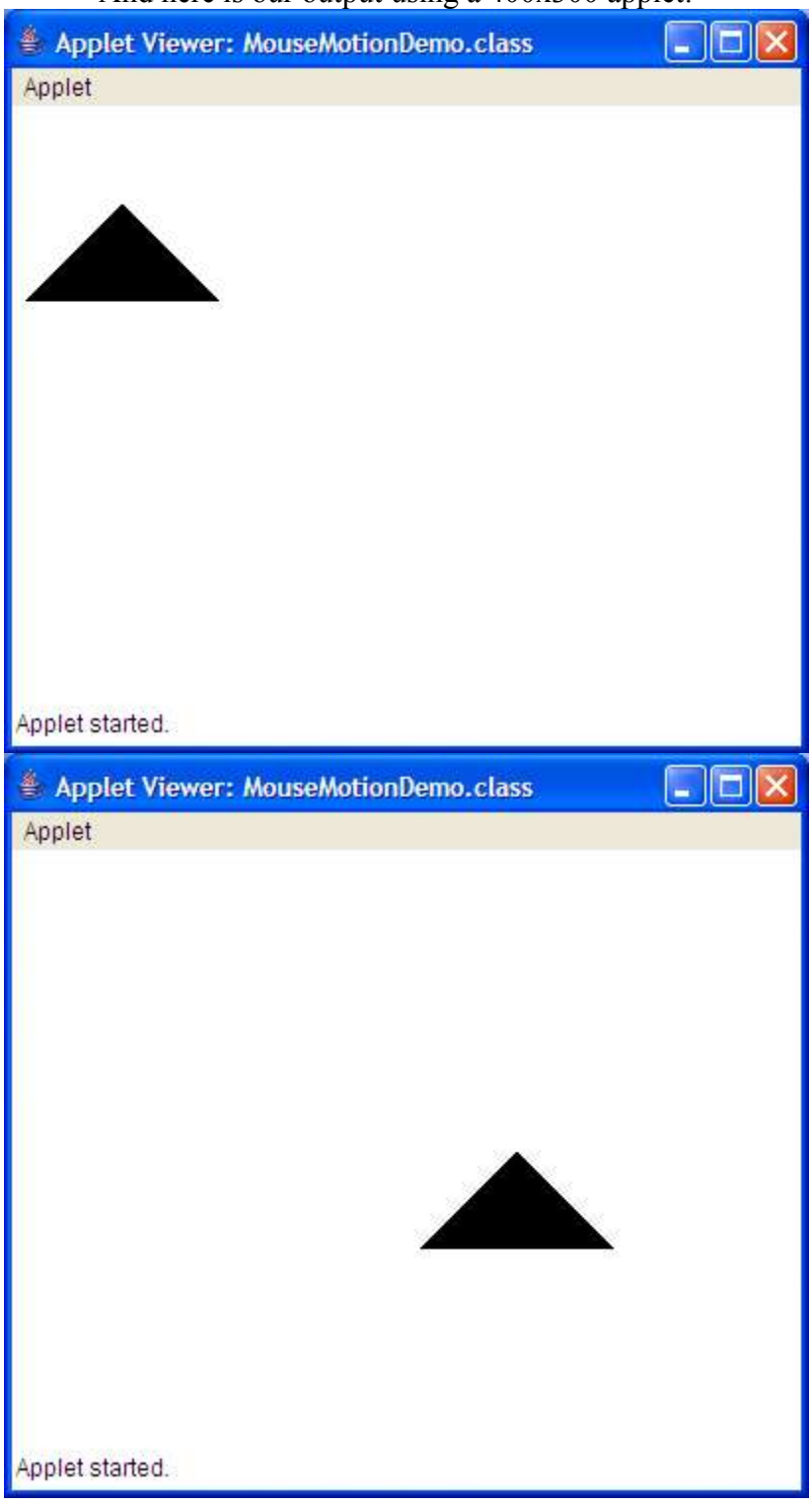
`void mouseMoved(MouseEvent e)` - mouse motion with the mouse buttons released
`void mouseDragged(MouseEvent e)` - mouse motion with the mouse buttons pressed

And here's an example demonstrating the use of the Mouse Motion events:

```
/* MouseMotionDemo.java */

import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class MouseMotionDemo extends Applet
      implements MouseListener, MouseMotionListener {
```

```
	int x[] = {50, 100, 150};
	int y[] = {100, 50, 100};
	Polygon p = new Polygon(x, y, 3);
	int oldx, oldy;

	public void init() {
		addMouseMotionListener(this);
	}

	public void paint(Graphics g) {
		g.fillPolygon(p);
	}

	public void mousePressed(MouseEvent e) {
		int x = e.getX();
		int y = e.getY();
		if (p.contains(x, y)) {
			oldx = x;
			oldy = y;
		}
	}

	public void mouseReleased(MouseEvent e) { }
	public void mouseClicked(MouseEvent e) { }
	public void mouseEntered(MouseEvent e) { }
	public void mouseExited(MouseEvent e) { }

	public void mouseMoved(MouseEvent e) { }

	public void mouseDragged(MouseEvent e) {
		int x = e.getX();
		int y = e.getY();
		if (p.contains(x, y)) {
			p.translate(x-oldx, y-oldy);
			oldx = x;
			oldy = y;
			repaint();
		}
	}
}
```

And here is our output using a 400x300 applet:



Conclusion

Now that we know how to accept input in our applets, we can start making games. Here's a simple example that uses both keyboard input and mouse input:

```
/* ShapeWar.java */

import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class ShapeWar extends Applet implements MouseListener,
KeyListener {
     Font font, bigFont;
     FontMetrics metrics, bigMetrics;
     int titleLeft, nameLeft, instructLeft, height;
     String victory;

     int phase = 0, player = 0, pieceNum = 0, moves = 5;
     int tx, ty, dx, dy;
     int[] left = {10, 10}, hx = new int[2], hy = new int[2];
     Color[] pColor = {Color.red, Color.blue};
     boolean[][] hit = new boolean[2][10];

     SquareWarrior[][] piece = new SquareWarrior[2][10];

     public void init() {
          font = new Font("Serif", font.PLAIN, 14);
          bigFont = new Font("Serif", font.BOLD, 30);
          setFont(font);
          metrics = getFontMetrics(font);
          bigMetrics = getFontMetrics(bigFont);
          titleLeft = (600 - metrics.stringWidth("Shape Wars")) / 2;
          nameLeft = (600 - metrics.stringWidth("by Dwayne Jeng")) / 2;
           instructLeft = (600 - metrics.stringWidth("Instructions:")) /
2;
          height = metrics.getHeight();

          setBackground(Color.black);
          setForeground(Color.white);

          addMouseListener(this);
          addKeyListener(this);
          requestFocus();
     }

     public void paint(Graphics g) {
          if (phase == 0) {
               g.drawString("Shape Wars", titleLeft, 25);
               g.drawString("by Dwayne Jeng", nameLeft, 25 + height);
                  g.drawString("Instructions:", instructLeft, 25 + 3 *
height);
                g.drawString("Each player gets 10 squares, which can be
placed anywhere on your own side of the board. The", 25, 25 + 5 *
height);
               g.drawString("object of the game is to destroy all enemy
```

```
squares. The player who moves first is randomly", 25, 25 + 6 * height);
                g.drawString("selected. During your turn, you can move
the current piece up to 5 times with the arrow keys.", 25, 25 + 7 *
height);
               g.drawString("You have the choice of not moving all five
spaces or even to not move at all. When you are", 25, 25 + 8 * height);
                 g.drawString("ready, click to fire from your square.
Where you click determines how you fire. When you fire,", 25, 25 + 9 *
height);
               g.drawString("3 points will be marked in a straight line
along the center of your square and the point you", 25, 25 + 10 *
height);
                g.drawString("clicked on. The first point will be cyan
and will be on the point you clicked on. The next 2", 25, 25 + 11 *
height);
                 g.drawString("points will be further away and will be
magenta. Only magenta points can do damage. After you", 25, 25 + 12 *
height);
                  g.drawString("fire, you will see the points and any
square hit will be marked in magenta and taken off the", 25, 25 + 13 *
height);
               g.drawString("board next turn. Beware, it is possible to
hit your own piece. After you see this screen, click", 25, 25 + 14 *
height);
              g.drawString("anywhere within the applet to pass the turn
to the other player. Play continues until all of one", 25, 25 + 15 *
height);
                g.drawString("player's squares have been destroyed. The
player who survives is the winner.", 25, 25 + 16 * height);
                 g.drawString("CLICK WITHIN THE APPLET TO CONTINUE...",
25, 25 + 18 * height);
          } else if (phase == 1) {
               g.drawLine(300, 0, 300, 450);
                  g.drawString(((player == 0)?"Red":"Blue") + " Player,
place square number " + (pieceNum + 1), 25, 25);

               for (int j = 0; j < pieceNum; j++) {
                         piece[player][j].drawWarrior(g, pColor[player],
false);
               }

               if (player == 1) {
                    for (int j = 0; j < 10; j++) {
                         piece[0][j].drawWarrior(g, pColor[0], false);
                    }
               }
          } else if (phase == 2) {
               for (int i = 0; i < 2; i++) {
                    for (int j = 0; j < 10; j++) {
                         if (piece[i][j].isAlive()) {
                              if (piece[i][j].getCenterX() < 0) {
                                   piece[i][j].translate(300, 0);
                              } else if (piece[i][j].getCenterX() > 600)
{
                                   piece[i][j].translate(-300, 0);
                              }
                              if (piece[i][j].getCenterY() < 0) {
```

```
                                    piece[i][j].translate(0, 225);
                               } else if (piece[i][j].getCenterY() > 450)
{
                                    piece[i][j].translate(0, -225);
                               }

                                    piece[i][j].drawWarrior(g, pColor[i],
false);
                          }
                     }
                }

                   piece[player][pieceNum].drawWarrior(g, pColor[player],
true);
                  g.drawString(((player == 0)?"Red":"Blue") + " Player's
turn...", 25, 25);
                g.drawString(moves + " moves left", 25, 25 + height);
                 g.drawString(left[0] + " Red squares left", 25, 25 + 2 *
height);
                g.drawString(left[1] + " Blue squares left", 25, 25 + 3 *
height);
           } else if (phase == 3) {
                g.setColor(Color.cyan);
                g.fillOval(tx - 2, ty - 2, 4, 4);
                g.setColor(Color.magenta);
                g.fillOval(hx[0] - 2, hy[0] - 2, 4, 4);
                g.fillOval(hx[1] - 2, hy[1] - 2, 4, 4);
                g.setColor(Color.white);

                for (int i = 0; i < 2; i++) {
                     for (int j = 0; j < 10; j++) {
                          if (hit[i][j]) {
                                piece[i][j].drawWarrior(g, Color.magenta,
true);
                          } else {
                                    piece[i][j].drawWarrior(g, pColor[i],
false);
                          }
                     }
                }
           } else if (phase == 4 || phase == 5 || phase == 6) {
                for (int i = 0; i < 2; i++) {
                     for (int j = 0; j < 10; j++) {
                          piece[i][j].drawWarrior(g, pColor[i], false);
                     }
                }
                g.setFont(bigFont);
                if (phase == 6) {
                  victory = "Draw game!";
                } else {
                       victory = ((phase == 4)?"Red":"Blue") + " Player
Wins!!!";
                  g.setColor((phase == 4)?Color.red:Color.blue);
                }
                     g.drawString(victory, (600 - bigMetrics.stringWidth
(victory)) / 2, (450 - bigMetrics.getHeight()) /2);
```

```
                g.setColor(Color.white);
                g.setFont(font);
                g.drawString("Click to play again...", 25, 25);
           }
      }

      public void mousePressed(MouseEvent event) {
           switch(phase) {
                case 0:
                     phase = 1;
                     repaint();
                     break;
                case 1:
                     if (player == 0 && event.getX() < 300) {
                               piece[player][pieceNum] = new SquareWarrior
(event.getX(), event.getY(), 50);
                          pieceNum++;
                          if (pieceNum > 9) {
                               player++;
                               pieceNum = 0;
                          }
                     } else if (player == 1 && event.getX() > 300) {
                               piece[player][pieceNum] = new SquareWarrior
(event.getX(), event.getY(), 50);
                          pieceNum++;
                          if (pieceNum > 9) {
                                 player = (int)Math.floor(Math.random() *
2);

                                          pieceNum = (int) Math.floor(10 *
Math.random());
                               while (!piece[player][pieceNum].isAlive())
{
                                    pieceNum++;
                                    if (pieceNum == 10) {
                                         pieceNum = 0;
                                    }
                               }

                               phase = 2;
                          }
                     }
                     repaint();
                     break;
                case 2:
                     tx = event.getX();
                     ty = event.getY();
                     dx = tx - piece[player][pieceNum].getCenterX();
                     dy = ty - piece[player][pieceNum].getCenterY();
                     hx[0] = tx + dx;
                     hy[0] = ty + dy;
                     hx[1] = hx[0] + dx;
                     hy[1] = hy[0] + dy;
                     for (int i = 0; i < 2; i++) {
                          for (int j = 0; j < 10; j++) {
                               if (piece[i][j].contains(hx[0], hy[0]) ||
```

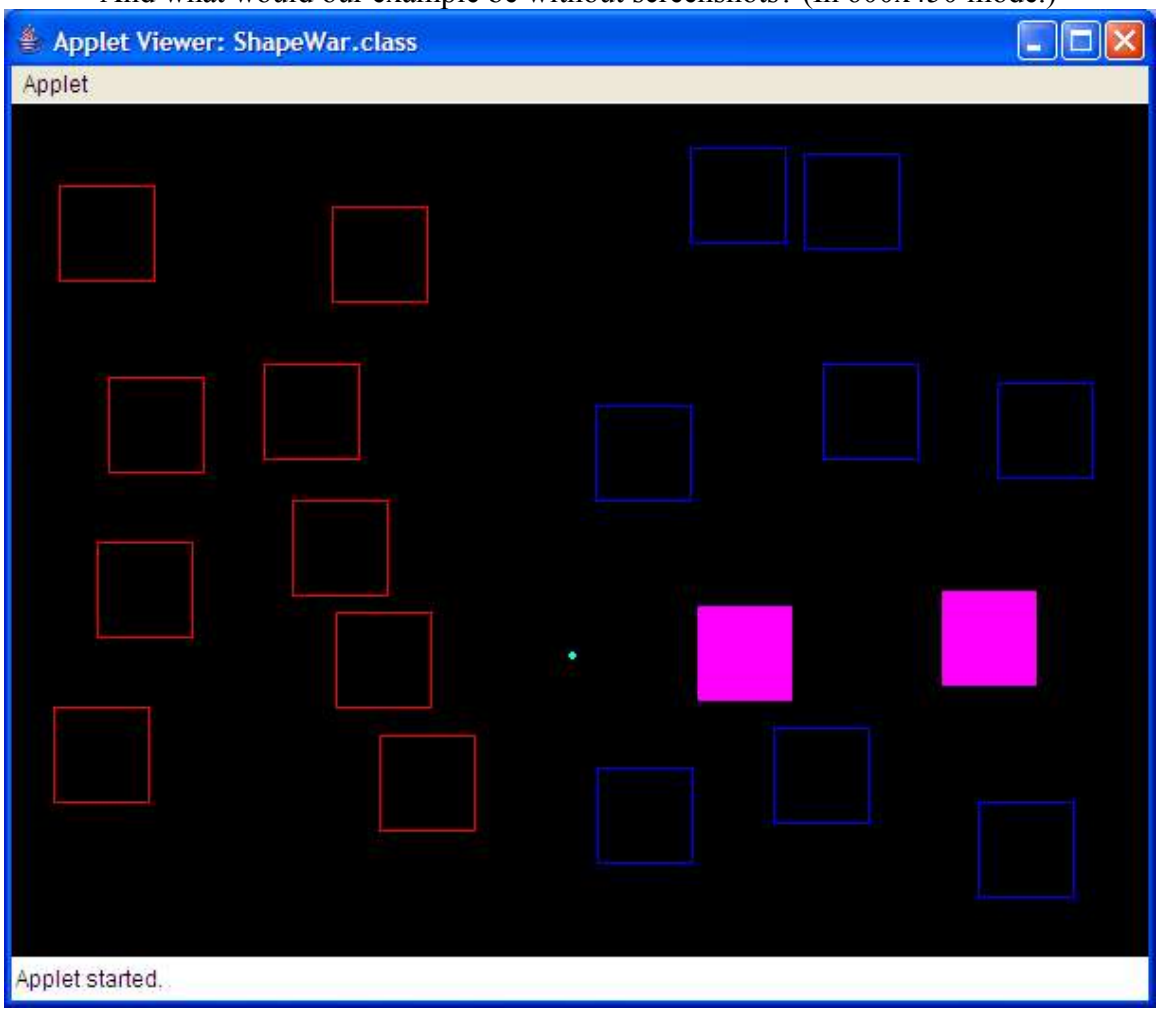
```
piece[i][j].contains(hx[1], hy[1])) {
                                        hit[i][j] = true;
                                   }
                              }
                         }
                         phase = 3;
                         repaint();
                         break;
                    case 3:
                         for (int i = 0; i < 2; i++) {
                              for (int j = 0; j < 10; j++) {
                                   if (hit[i][j]) {
                                        piece[i][j].translate(1000, 1000);
                                        piece[i][j].die();
                                        left[i]--;
                                        hit[i][j] = false;
                                   }
                              }
                         }

                         if (left[0] > 0 && left[1] == 0) {
                              phase = 4;
                         } else if (left[0] == 0 && left[1] > 0) {
                              phase = 5;
                         } else if (left[0] == 0 && left[1] == 0) {
                              phase = 6;
                         } else {
                              moves = 5;
                              phase = 2;
                              player = (player == 1)?0:1;

                                pieceNum = (int) Math.floor(10 * Math.random
());
                              while (!piece[player][pieceNum].isAlive()) {
                                   pieceNum++;
                                   if (pieceNum == 10) {
                                        pieceNum = 0;
                                   }
                              }
                         }
                         repaint();
                         break;
                    case 4:
                    case 5:
                    case 6:
                         left[0] = 10;
                         left[1] = 10;
                         phase = 0;
                         pieceNum = 0;
                         repaint();
                    default:
               }
          }

          public void mouseReleased(MouseEvent event) { }
          public void mouseEntered(MouseEvent event) { }
          public void mouseExited(MouseEvent event) { }
```

```
     public void mouseClicked(MouseEvent event) { }

     public void keyPressed(KeyEvent event) {
          if (phase == 2 && moves > 0) {
               if (event.getKeyCode() == KeyEvent.VK_UP) {
                    piece[player][pieceNum].translate(0, -25);
               } else if (event.getKeyCode() == KeyEvent.VK_DOWN) {
                    piece[player][pieceNum].translate(0, 25);
               } else if (event.getKeyCode() == KeyEvent.VK_LEFT) {
                    piece[player][pieceNum].translate(-25, 0);
               } else if (event.getKeyCode() == KeyEvent.VK_RIGHT) {
                    piece[player][pieceNum].translate(25, 0);
               }
               moves--;
               repaint();
          }
     }

     public void keyReleased(KeyEvent event) { }
     public void keyTyped(KeyEvent event) { }
}
```

```
/* SquareWarrior.java */

import java.awt.*;

public class SquareWarrior {
     int cx, cy;
     int[] px = new int[4], py = new int[4];
     Polygon poly;
     boolean alive = true;

     public SquareWarrior(int x, int y, int size) {
          cx = x;
          cy = y;
          px[0] = cx - size / 2;
          px[1] = cx + size / 2;
          px[2] = cx + size / 2;
          px[3] = cx - size / 2;
          py[0] = cy - size / 2;
          py[1] = cy - size / 2;
          py[2] = cy + size / 2;
          py[3] = cy + size / 2;
          poly = new Polygon(px, py, 4);
     }

     public Point getCenter() {
          return new Point(cx, cy);
     }

     public int getCenterX() {
          return cx;
     }

     public int getCenterY() {
          return cy;
```

```
	}

	public void translate(int dx, int dy) {
		cx += dx;
		cy += dy;
		poly.translate(dx, dy);
	}

	public void drawWarrior(Graphics g, Color c, boolean fill) {
		Color oldColor = g.getColor();
		g.setColor(c);
		if (fill) {
			g.fillPolygon(poly);
		} else {
			g.drawPolygon(poly);
		}
		g.setColor(oldColor);
	}

	public boolean contains(int x, int y) {
		return poly.contains(x, y);
	}

	public boolean contains(Point p) {
		return poly.contains(p);
	}

	public void die() {
		alive = false;
	}

	public boolean isAlive() {
		return alive;
	}
}
```

And what would our example be without screenshots? (In 600x450 mode.)



Homework Assignment!

Make a simple graphic 2-player tic-tac-toe applet. Input may be either with the keyboard or with the mouse. Bonus points if you give the user a choice between both.