CECS 277 – Lecture 19 – Animation using Threads

Animation using threads is easy. The parameters of a given image are set in the class, then, using those parameters, an image is drawn using the paint method. The parameters of the image are then modified and then the image is redrawn. This process of updating and redrawing can be performed many times per second to produce an animation. In the example below, a circle is drawn on an empty field, the location of the circle is modified and then the image is repainted. A thread is used to continuously repeat the process of calling the methods that modify and repaint the circle. Normally, when repaint is called, the image is cleared and then redrawn, which can often produce an undesired flicker. A process called double buffering is used here to eliminate this flickering. Double buffering is the process of drawing the next image to a secondary canvas and then transferring that image over as the new one. If there is considerable calculation for the next frame, an extra threads can be used to perform the necessary calculations.

Example: A Bouncing Ball Application.

```
import java.awt.*;
import javax.swing.*;
public class Bounce extends JPanel implements Runnable {
   int x,y;
   int dx,dy;
   int size = 25;
   Thread t;
   public Bounce(){
      setDoubleBuffered(true);
      t = new Thread(this);
      x = 50;
      y = 50;
      dx = 1;
      dy = 1;
      t.start();
   }
   public void paint(Graphics g){
      super.paint(g);
      move();
      g.setColor(Color.YELLOW);
      g.fillRect(0, 0, getWidth(), getHeight());
      g.setColor(Color.GRAY);
      g.fillOval(x, y, size, size);
      //Toolkit.getDefaultToolkit().sync();
      g.dispose();
   }
   @Override
   public void run() {
      while(true){
         repaint();
         try {
```

```
            Thread.sleep(16); //60 fps
         }catch (InterruptedException e) {}
      }
   }
   public void move(){
      // test for edges
      if(dx > 0) {
         if(x > getWidth() - size){
            dx = -dx;
         }
      } else {
         if(x < 0){
            dx = -dx;
         }
      }
      if(dy > 0) {
         if(y > getHeight() - size){
            dy = -dy;
         }
      } else {
         if(y < 0){
            dy = -dy;
         }
      }
      // update position
      x += dx;
      y += dy;
   }
}
import javax.swing.JFrame;
public class Bounce extends JFrame{
   public BounceMain(){
      add(new Bounce());
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setSize(300,300);
      setVisible(true);
   }
   public static void main(String [] args){
      new BounceMain();
   }
}
```