CT255 [2D games in Java]

Week#5 Sample Solution – Invaders is finished! ☺

The main application class (single instance)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.image.*;
import java.util.ArrayList;
import java.util.Iterator;

public class InvadersApplication extends JFrame implements Runnable, KeyListener {
    // member data
  public static final Dimension WindowSize = new Dimension(800,600); // this is now public so that
other classes can read!
  private BufferStrategy strategy;
  private Graphics offscreenBuffer;
  private static final int NUMALIENS = 30;
  private Alien[] AliensArray = new Alien[NUMALIENS];
  private Spaceship PlayerShip;
  private Image bulletImage;
  private ArrayList bulletsList = new ArrayList();
  private boolean isInitialised = false;
  private static String workingDirectory;
  private boolean isGameInProgress = false;
  private int enemyWave = 1;
  private int score = 0;
  private int highscore = 0;

  // constructor
  public InvadersApplication() {
        //Display the window, centred on the screen
        Dimension screensize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        int x = screensize.width/2 - WindowSize.width/2;
        int y = screensize.height/2 - WindowSize.height/2;
        setBounds(x, y, WindowSize.width, WindowSize.height);
        setVisible(true);
        this.setTitle("Space Invaders! (finished at last)");

        // load images from disk. Make sure you have the path right!
        ImageIcon icon = new ImageIcon(workingDirectory + "\\alien_ship_1.png");
        Image alienImage = icon.getImage();
        icon = new ImageIcon(workingDirectory + "\\alien_ship_2.png");
        Image alienImage2 = icon.getImage();
        icon = new ImageIcon(workingDirectory + "\\bullet.png");
        bulletImage = icon.getImage();

        // create and initialise some aliens, passing them each the image we have loaded
        for (int i=0; i<NUMALIENS; i++) {
          AliensArray[i] = new Alien(alienImage,alienImage2);
        }

        // create and initialise the player's spaceship
        icon = new ImageIcon(workingDirectory + "\\player_ship.png");
        Image shipImage = icon.getImage();
        PlayerShip = new Spaceship(shipImage, bulletImage);

        // create and start our animation thread
        Thread t = new Thread(this);
        t.start();

        // send keyboard events arriving into this JFrame back to its own event handlers
        addKeyListener(this);

        // initialise double-buffering
        createBufferStrategy(2);
        strategy = getBufferStrategy();
        offscreenBuffer = strategy.getDrawGraphics();

        isInitialised = true;
  }
```

```
// thread's entry point
public void run() {
  while ( 1==1 ) {

    // 1: sleep for 1/50 sec
    try {
      Thread.sleep(20);
    } catch (InterruptedException e) { }

    // 2: animate game objects, if the game state is 'in progress'
    if ( isGameInProgress ) {
      boolean anyAliensAlive = false;
      boolean alienDirectionReversalNeeded = false;
      for (int i=0;i<NUMALIENS; i++) {
        if ( AliensArray[i].isAlive ) {
          anyAliensAlive = true;
          if (AliensArray[i].move())
            alienDirectionReversalNeeded=true;

          // check for alien #i collision with player ship => game over
          if ( isCollision(PlayerShip.x,AliensArray[i].x,PlayerShip.y,AliensArray[i].y,54,50,32,32) )
          {
            isGameInProgress=false;
          }

        }
      }
      if (alienDirectionReversalNeeded) {
        for (int i=0;i<NUMALIENS; i++) {
          if ( AliensArray[i].isAlive ) {
            AliensArray[i].reverseDirection();
            // if passed off bottom of screen, game over
            if ( AliensArray[i].y>WindowSize.height-20) {
              isGameInProgress=false;
            }
          }
        }
      }

      if (!anyAliensAlive) {
        enemyWave++;
        startNewWave();
      }

      PlayerShip.move();

      Iterator iterator = bulletsList.iterator();
      while(iterator.hasNext()){
        PlayerBullet b = (PlayerBullet) iterator.next();
        if (b.move()) {
          // true was returned by move method if bullet needs destroying due to going offscreen
          // iterator.remove is a safe way to remove from the ArrayList while iterating thru it
          iterator.remove();
        }
        else {
          // check for collision between this bullet and any alien
          double x2 = b.x, y2 = b.y;
          double w1 = 50, h1 = 32;
          double w2 = 6, h2 = 16;
          for (int i=0;i<NUMALIENS; i++) {
            if ( AliensArray[i].isAlive ) {
              double x1 = AliensArray[i].x;
              double y1 = AliensArray[i].y;
              if ( isCollision(x1,x2,y1,y2,w1,w2,h1,h2) ) {
                // destroy alien and bullet
                AliensArray[i].isAlive=false;
                iterator.remove(); // this is a safe way to remove from an ArrayList while iterating
                score+=10;
                if (score>highscore)
                  highscore=score;

                break; // no need to keep checking aliens so break out of for loop
              }
```

```
            }
          }
        }
      }

    }

    // 3: force an application repaint
    this.repaint();
  }
}

// Three Keyboard Event-Handler methods
  public void keyPressed(KeyEvent e) {
    if ( isGameInProgress ) {
      if (e.getKeyCode()==KeyEvent.VK_LEFT)
        PlayerShip.setXSpeed(-4);
      else if (e.getKeyCode()==KeyEvent.VK_RIGHT)
        PlayerShip.setXSpeed(4);
      else if (e.getKeyCode()==KeyEvent.VK_SPACE)
        bulletsList.add(PlayerShip.shootBullet());
    }
    else {
      startNewGame();
    }
  }

  public void keyReleased(KeyEvent e) {
    if (e.getKeyCode()==KeyEvent.VK_LEFT || e.getKeyCode()==KeyEvent.VK_RIGHT)
      PlayerShip.setXSpeed(0);
  }

  public void keyTyped(KeyEvent e) { }
  //

  public void startNewGame() {
    enemyWave = 1;
    score = 0;
    isGameInProgress = true;
    startNewWave();
  }

  public void startNewWave() {
    // re-position aliens and player ship
    for (int i=0; i<NUMALIENS; i++) {
        double xx = (i%5)*80 + 70;
        double yy = (i/5)*40 + 50;
        AliensArray[i].setPosition(xx, yy);
        AliensArray[i].setXSpeed(1+enemyWave);
        AliensArray[i].isAlive=true;
        AliensArray[i].framesDrawn=0;
    }
    PlayerShip.setPosition(300,530);
  }

  // helper method for collision detection
private boolean isCollision(double x1,double x2,double y1,double y2,double w1,double w2,double
h1,double h2) {
  if (
      ((x1<x2 && x1+w1>x2) || (x2<x1 && x2+w2>x1))
      &&
      ((y1<y2 && y1+h1>y2) || (y2<y1 && y2+h2>y1))
    )
    return true;
  else
    return false;
}

// helper method for drawing strings centred at specified position
private void writeString(Graphics g, int x, int y, int fontSize, String message) {
    Font f = new Font( "Times", Font.PLAIN, fontSize );
    g.setFont(f);
    FontMetrics fm = getFontMetrics(f);
    int width = fm.stringWidth(message);
```

```
      g.drawString(message, x-width/2, y);
  }

  // application's paint method
  public void paint(Graphics g) {
    if (!isInitialised)
      return;

    g = offscreenBuffer; // draw to offscreen buffer

    // clear the canvas with a big black rectangle
    g.setColor(Color.BLACK);
    g.fillRect(0, 0, WindowSize.width, WindowSize.height);

    if ( isGameInProgress ) {
      // redraw all game objects
      for (int i=0;i<NUMALIENS; i++)
        AliensArray[i].paint(g);

      PlayerShip.paint(g);

      Iterator iterator = bulletsList.iterator();
      while(iterator.hasNext()){
        PlayerBullet b = (PlayerBullet) iterator.next();
        b.paint(g);
      }

      // score
      g.setColor(Color.WHITE);
      writeString(g,WindowSize.width/2,60,30,"Score: "+score+"    Best: "+highscore);
    }
    else {
      // redraw the menu screen
      g.setColor(Color.WHITE);
      writeString(g,WindowSize.width/2,200,60,"GAME OVER");
      writeString(g,WindowSize.width/2,300,30,"Press any key to play");
      writeString(g,WindowSize.width/2,350,25,"[Arrow keys to move, space to fire]");
    }

    // flip the buffers
    strategy.show();
  }

  // application entry point
  public static void main(String[] args) {
    workingDirectory = System.getProperty("user.dir");
    System.out.println("Working Directory = " + workingDirectory);
    InvadersApplication w = new InvadersApplication();
  }
}
```

The Sprite2D class (superclass for Alien, Bullet and Spaceship classes)

```
import java.awt.*;
public class Sprite2D {
  // member data
  protected double x,y;
  protected double xSpeed=0;
  protected Image myImage, myImage2;
  int framesDrawn=0;

  // constructor
  public Sprite2D(Image i, Image i2) {
    myImage = i;
    myImage2 = i2;
  }
  public void setPosition(double xx, double yy) {
    x=xx;
    y=yy;
  }
  public void setXSpeed(double dx) {
    xSpeed=dx;
  }
```

```
  public void paint(Graphics g) {
    framesDrawn++;
    if ( framesDrawn%100<50 )
      g.drawImage(myImage, (int)x, (int)y, null);
    else
      g.drawImage(myImage2, (int)x, (int)y, null);
  }
}
```

The Spaceship class (for player spaceship: one instance)

```
import java.awt.Image;
public class Spaceship extends Sprite2D {
  private Image bulletImage;

  public Spaceship(Image i, Image bullet) {
    super(i,i); // invoke constructor on superclass Sprite2D
    bulletImage = bullet;
  }
  public void move() {
    // apply current movement
    x+=xSpeed;

    // stop movement at screen edge?
    if (x<=0) {
      x=0;
      xSpeed=0;
    }
    else if (x>=InvadersApplication.WindowSize.width-myImage.getWidth(null)) {
      x=InvadersApplication.WindowSize.width-myImage.getWidth(null);
      xSpeed=0;
    }
  }

    // method to handle shooting
    public PlayerBullet shootBullet() {
      // add a new bullet to our list
      PlayerBullet b = new PlayerBullet(bulletImage);
      b.setPosition(this.x+54/2, this.y);
      return b;
    }
}
```

The Alien class (the Application class maintains an array of instances of these)

```
import java.awt.Graphics;
import java.awt.Image;
public class Alien extends Sprite2D {
  public boolean isAlive = true;

  public Alien(Image i, Image i2) {
    super(i,i2); // invoke constructor on superclass Sprite2D
  }
  public void paint(Graphics g) {
    if (isAlive)
      super.paint(g);
  }
  public boolean move() {
    x+=xSpeed;

    // direction reversal needed?
    if (x<=0 || x>=InvadersApplication.WindowSize.width-myImage.getWidth(null))
      return true;
    else
      return false;
  }
  public void reverseDirection() {
    xSpeed=-xSpeed;
    y+=20;
```

```
  }
}
```

The PlayerBullet class (the Application class maintains an ArrayList of instances of these)

```
import java.awt.Image;
public class PlayerBullet extends Sprite2D {
  public PlayerBullet(Image i) {
    super(i,i); // invoke constructor on superclass Sprite2D
  }
  public boolean move() {
    y-=10;
    return (y<0); // return true if bullet is offscreen and needs destroying
  }
}
```