Appendix - SpaceMap

```
package src;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.AffineTransform;
import java.io.File;

import javax.swing.JFileChooser;
import javax.swing.JPanel;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;

/**
 *
 */

/**
 * This is the world map actor
 *
 * @author Sky Johnson
 */
public class SpaceMap extends JPanel

{

	/*
	 * Serial Version
	 */
	private static final long serialVersionUID = -3535839203384839672L;

	/*
	 * Title Font
	 */
	private static final Font TITLE_FONT = new Font("Dialog", Font.PLAIN, 25);
	/*
	 * Text font
	 */
	private static final Font SUBTITLE_FONT = new Font("Dialog",
				Font.ROMAN_BASELINE, 15);
```

```
	/*
	 * Random
	 */

	/*
	 * dictator
	 */
	private Dictator dictator;

	public JFileChooser chooser;

	/*
	 * Constructor for SpaceMap
	 */
	SpaceMap(Dictator dic) {
		this.dictator = dic;
		// set window
		setPreferredSize(new Dimension(dic.SIZE_X, dic.SIZE_Y));
		setBackground(Color.BLACK);

	}

	/*
	 * (non-Javadoc)
	 *
	 * @see javax.swing.JComponent#paintComponent(java.awt.Graphics)
	 */
	public void paintComponent(Graphics g) {
		super.paintComponent(g);

		Graphics2D graphics = (Graphics2D) g;
		graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
				RenderingHints.VALUE_ANTIALIAS_ON);
		graphics.setColor(Color.WHITE);

		AffineTransform identity = graphics.getTransform();


		if (!dictator.isGenerated()) {
			graphics.setTransform(identity);
			drawTextCenter("Welcome to Asteroids!", TITLE_FONT, graphics, -100);

			graphics.setTransform(identity);
			drawTextCenter(
					"Select either Seed play or Music play, Press S for seed,
Press M for music play",
```

```
                                        SUBTITLE_FONT, graphics, -50);

                        // Select Options
                        graphics.setTransform(identity);
                        drawTextCenterOffset("M", SUBTITLE_FONT, graphics, 0, -10);

                        graphics.setTransform(identity);
                        drawTextCenterOffset("S", SUBTITLE_FONT, graphics, 0, 10);



                        if (dictator.musicgame && dictator.entered && !dictator.songselected &&
!dictator.jFileChoseOpen&&!dictator.waitingforChoose){

                                dictator.jFileChoseOpen = true;
                                dictator.waitingforChoose = true;

                                  String choosertitle = "Pick a Song";

                                chooser = new JFileChooser();
                           chooser.setCurrentDirectory(new java.io.File("."));
                           chooser.setDialogTitle(choosertitle);
                           chooser.setAcceptAllFileFilterUsed(false);

                           FileNameExtensionFilter filter = new FileNameExtensionFilter("MPEG3
songs", "mp3");
                     chooser.addChoosableFileFilter(filter);

                           if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
                             dictator.song = chooser.getSelectedFile();

                             dictator.songselected = true;
                             }
                           else {

                             dictator.waitingforChoose = false;
                             dictator.entered = false;
                             dictator.jFileChoseOpen= false;
                             }
                        }
                        //first entered select in Seed game select
                        if ( dictator.entered && !dictator.seedtypeing && dictator.seedgame &&
!dictator.firstenteredseedreleased) {
                                dictator.seedtypeing = true;
                                dictator.firstenteredseedreleased = false;
                                dictator.initseedstring = true;
```

```
                }
                // Second Entered command
                if( dictator.seedgame && dictator.seedtypeing && dictator.entered &&
dictator.firstenteredseedreleased){
                        dictator.seedtypeing = false;
                }

                if (dictator.Mpress || dictator.musicgame) {
                        dictator.musicgame = true;
                        dictator.seedgame = false;
                        graphics.setTransform(identity);
                        graphics.drawRect((dictator.SIZE_X / 2) - 22,
                                        dictator.SIZE_Y / 2 - 17, 12, 12);

                        dictator.seedtypeing = false;

                        graphics.setTransform(identity);

                        drawTextCenter("Press enter to open song chooser",
SUBTITLE_FONT, graphics, 100);

                }

                if (dictator.Spress || dictator.seedgame) {

                        dictator.musicgame = false;
                        dictator.seedgame = true;
                        graphics.setTransform(identity);
                        graphics.drawRect((dictator.SIZE_X / 2) -1,
                                        dictator.SIZE_Y / 2 - 17, 12, 12);

                        //Seed Input String!!!!
                        graphics.setTransform(identity);

                        drawTextCenter(dictator.seed, SUBTITLE_FONT, graphics, 100);

                }

                if (dictator.entered && !dictator.seedtypeing && (dictator.seedgame ||
dictator.musicgame)&&!dictator.jFileChoseOpen ||
(dictator.musicgame&&dictator.songselected)) {
                        dictator.selectDecision = true;

                }
        }
```

```
                if (!dictator.checkForRestart() && dictator.isGenerated()) {
                        // DrawScores
                        graphics.setFont(SUBTITLE_FONT);
                        graphics.drawString("SCORE: " + dictator.getScore(), 40,
                                        dictator.SIZE_Y - 40);

                        // draw Lives
                        graphics.translate(dictator.SIZE_X - 100, dictator.SIZE_Y - 40);
                        for (int i = 0; i < dictator.lives; i++) {
                                graphics.drawLine(-8, 10, 0, -10);
                                graphics.drawLine(8, 10, 0, -10);
                                graphics.drawLine(-6, 6, 6, 6);
                                graphics.translate(30, 0);
                        }

                        //draw level progression rect
                        graphics.setTransform(identity);
                        graphics.translate(0, dictator.SIZE_Y-50);
                        graphics.drawRect((dictator.SIZE_X/2)-(dictator.SIZE_X/8), 0,
dictator.SIZE_X/4, 10);

                        graphics.setTransform(identity);
                        graphics.translate(0, dictator.SIZE_Y-50);
                        graphics.setColor(Color.WHITE);
                        double PercentDone =
(double)(dictator.lineSoFar)/(dictator.TotalLines)*(dictator.SIZE_X/4);
                        graphics.fillRect((dictator.SIZE_X/2)-(dictator.SIZE_X/8), 0, (int)
PercentDone, 10);


                        // draw Bullets
                        graphics.setTransform(identity);
                        graphics.translate(dictator.SIZE_X - 110, dictator.SIZE_Y - 80);
                        for (int i = 0; i < dictator.BULLET_MAX - dictator.bulletCount; i++) {
                                graphics.setColor(Color.WHITE);
                                graphics.drawOval(0, 0, 2, 4);
                                graphics.translate(10, 0);
                        }
                        // draw Stars Update
                        for (Star i : dictator.starlist) {
                                graphics.setTransform(identity);
                                i.drawStar(graphics);
                        }

                        // Draw Actors Update
                        for (int i = 0; i < dictator.getActor().size(); i++) {
```

```
                        drawActor(graphics, dictator.getActor().get(i), dictator
                                .getActor().get(i).getPosition());
                        graphics.setTransform(identity);
                }

                // Situational and Menus

                // Paused
                if (dictator.paused) {
                        drawTextCenter("PAUSED", TITLE_FONT, graphics, 0);
                        drawTextCenter("Press Esc to Exit", SUBTITLE_FONT, graphics,
                                -50);
                        drawTextCenter("Press P to Unpause", SUBTITLE_FONT, graphics,
                                -90);
                        drawTextCenter("Press R to Restart", SUBTITLE_FONT, graphics,
                                -70);

                }
                // End Game
                if (dictator.gameOver) {
                        drawTextCenter("GAME OVER", TITLE_FONT, graphics, 0);
                        drawTextCenter("Press Esc to Exit", SUBTITLE_FONT, graphics,
                                -50);
                        drawTextCenter("Press R to Restart", SUBTITLE_FONT, graphics,
                                -70);
                }
                if(dictator.endGame){
                        drawTextCenter("GAME WON! CONGRADULATIONS", TITLE_FONT,
graphics, 0);
                        drawTextCenter("Final Score: "+Integer.toString(dictator.score),
TITLE_FONT, graphics, 50);
                        drawTextCenter("Press Esc to Exit", SUBTITLE_FONT, graphics,
                                -50);
                        drawTextCenter("Press R to Restart", SUBTITLE_FONT, graphics,
                                -70);
                }
        }

        }

        // Rotates and translates the 2dGraphics to oppropriate position object
        private void drawActor(Graphics2D graphics, Actor actor, Position lookingat) {
                // DRAWING STUFF NOT DONE
                graphics.translate(lookingat.getX(), lookingat.getY());
                double rotation = actor.getRotation();
                if (rotation != 0.0f) {
```

```
                graphics.rotate(actor.getRotation());
            }
            actor.draw(graphics, dictator);
        }

        private void drawTextCenter(String string, Font font, Graphics2D g,
                int downspace) {
            g.setColor(Color.WHITE);
            g.setFont(font);
            g.drawString(string, dictator.SIZE_X / 2
                    - g.getFontMetrics().stringWidth(string) / 2, dictator.SIZE_Y
                    / 2 + downspace);
        }

        private void drawTextCenterOffset(String string, Font font, Graphics2D g,
                int downSpace, int rightSpace) {
            g.setColor(Color.WHITE);
            g.setFont(font);
            g.drawString(string, dictator.SIZE_X / 2
                    - g.getFontMetrics().stringWidth(string) + rightSpace,
                    dictator.SIZE_Y / 2 - g.getFontMetrics().stringWidth(string)
                            / 2 + downSpace);
        }
}
```