Appendix - SpawnController

```
package src;

import java.awt.Component;
import java.io.File;
import java.util.ArrayList;
import java.util.Random;

public class SpawnController {

	private Dictator dictator;

	private int spawnCap;

	private SongListener songlistener;

	private SeedListener seedListener;

	private ArrayList<String> level;

	private int currentLine;

	private int listenRateSpeed;

	private String currentString;

	public boolean endgame;

	public String thisString;

	public File thisFile;

	public SpawnController(Dictator dic) {
		this.dictator = dic;
		endgame = false;
		spawnCap = 100;
		currentLine = 0;
		currentString = "asdf";
		listenRateSpeed = 100;

	}

	public int getLineSoFar() {
		return currentLine;
	}
```

```
public void set(int spawnCap, int spawnFreq, int spawn) {
	this.spawnCap = spawnCap;

}

public void setThisString(String thisString) {
	this.thisString = thisString;
}

public void reset() {
	currentString = "asdf";
	currentLine = 0;

}

public void update2(Dictator dic) {
	if (dictator.selectDecision) {
		if (dictator.musicgame) {

			thisFile = dictator.song;
			songlistener = new SongListener(thisFile, dic);
			level = songlistener.getLevel();
			listenRateSpeed = songlistener.getRate();
			dictator.addRandoms();

		} else if (dictator.seedgame) {
			thisString = dictator.seed;
			seedListener = new SeedListener(thisString, dic);
			level = seedListener.getLevel();
			dictator.addRandoms();
		}
		dictator.setGenerated(true);
	}
}

public int findRadius(String s) {
	int RadiusOffSetStart = s.indexOf("AsteroidSize:")
			+ "AsteroidSize:".length();
	int RadiusOffSetEnd = s.indexOf(";Position:");
	int AsteroidRadius = Integer.parseInt(s.substring(RadiusOffSetStart,
			RadiusOffSetEnd));

	return AsteroidRadius;
}
```

```
public Position findPosition(String s) {
	int PositionOffSetStart = s.indexOf(";Position:")
			+ ";Position:".length();
	int PositionOffSetEnd = s.indexOf(",P");
	int positionx = Integer.parseInt(s.substring(PositionOffSetStart,
			PositionOffSetEnd));
	int PositionOffSetStarty = s.indexOf(",P") + ",P".length();
	int PositionOffSetEndy = s.indexOf(";Movement:");
	int positiony = Integer.parseInt(s.substring(PositionOffSetStarty,
			PositionOffSetEndy));

	Position returnPosition = new Position(positionx, positiony);
	return returnPosition;
}

public Movement findMovement(String s) {
	int VelocityOffSetStart = s.indexOf(";Movement:")
			+ ";Movement:".length();
	int VelocityOffSetEnd = s.indexOf(",M");
	int Velocityx = Integer.parseInt(s.substring(VelocityOffSetStart,
			VelocityOffSetEnd));

	int VelocityOffSetStarty = s.indexOf(",M") + ",M".length();
	int VelocityOffSetEndy = s.indexOf(";AsteroidCallEnd:");
	int Velocityy = Integer.parseInt(s.substring(VelocityOffSetStarty,
			VelocityOffSetEndy));

	Movement returnMovement = new Movement(Velocityx, Velocityy);
	return returnMovement;
}

public void update() {

	if (dictator.getTime() % listenRateSpeed == 0) {
		try {
			currentString = level.get(currentLine);
			currentLine++;
		} catch (IndexOutOfBoundsException e) {
			dictator.endGame = true;
		} catch (Exception e) {

		}

		if (currentString.contains("AsteroidsToSpawn")) {
			String asteroidsToSpawn = currentString.substring(
					"AsteroidsToSpawn:".length(),
```

```
                                        currentString.indexOf("AsteroidCall:"));

                                int asteroidsToSpawnInt = Integer.parseInt(asteroidsToSpawn);

                                for (int i = 1; i <= asteroidsToSpawnInt; i++) {

                                        int offsetInt = currentString.indexOf("AsteroidCall:"
                                                + Integer.toString(i));

                                        int offsetIntEnd = currentString.indexOf("AsteroidCallEnd:"
                                                + Integer.toString(i))
                                                + "AsteroidCallEnd:".length();

                                        String asteroidString = currentString.substring(offsetInt,
                                                offsetIntEnd);

                                        int AsteroidRadius = findRadius(asteroidString);

                                        Position pos = findPosition(asteroidString);
                                        if(toClose(dictator.getPlayerPosition(), pos)){
                                                Position a2 = pos;

a2.setX(pos.getX()*dictator.getPlayerPosition().getX());

a2.setY(pos.getY()*dictator.getPlayerPosition().getY());
                                                pos = a2;
                                        }

                                        Movement mov = findMovement(asteroidString);

                                        Asteroid new1 = new Asteroid(dictator, AsteroidRadius,
pos,
                                                        mov);
                                        dictator.addToAddActors(new1);

                                }
                        }
                        // Star Editing
                        for (Star star : dictator.starlist) {
                                if (star.getGroup() == 0) {
                                        star.setToSize(1);

                                } else {
                                        star.setToSize(1);
                                }
                        }
```

```
                }
        }

        private boolean toClose(Position playerPosition, Position a) {

                if (Math.abs(playerPosition.getX() - a.getX()) < 80
                                && Math.abs(playerPosition.getY() - a.getY()) < 80) {
                        return true;
                }

                return false;
        }

        public int getlevelsize() {
                // TODO Auto-generated method stub
                return level.size();
        }
}
```