Michael Rallo msr5zb 12358133
Project Documentation

Requirements

1. Object Oriented elements that you write code for:
   a. Classes.
      –Plenty Of These Within Program. Ex: SpaceLevel.java
   b. SubClasses.
      –Plenty Of These as well throughout program. Ex: Msr5zbGameVisualizer is a subclass of Application. There Are plenty of inner/anonymouse subclasses as well. Ex: various runLaters within Enemy/Missle Threads.

c. At Least one anstract class and/or interface.
   -LevelStandards would be a good example of an interface.

2. Code elements that you utilize.
   a. One or more collection classes.
      -I used many ArrayLists consisting of classes. Ex: ArrayList<SpiderEnemy> enemies.
   b. Exception Handling
      -This is done throughout the program as well. Ex: I/O Catches To See If Text file for highscores exists: Lines 23-33 in ProcessFile.java

3. The application must have cleary define model (as the M in MVC)
   -The User uses The Controller which Manipulates The Classes/Model/Scenes (Levels) Which Updates the View which the User Sees and will further use.

4. The UI must utilize multiple scenes and/or a scene where the contents of the scene graph are changed based on application state.
   -Done via Levels.

5. There Must be a way to access “About” information that includes information about you and the application.
   -Can be found in the Information section under the About on the Program

6. Your application must save data and load data.
   -HighScores For Each Level Are Saved To Text Files. Upon Opening a Level, The App checks to see if a highscore exists for that level already. It there is one, it will parse and grab the highscore from it to display to the user while he/she plays. If not, one will be created and update dynamically as the user increases his/her score. NO CHEATING ! (files are not encrypted/hashed/salted, lol)

Expectations
1. The application is functional for a define activity, task, or purpose.

-This application is designed to Entertain!
-It combines the joys of passing time with a game whilst listening to your favorite songs.

2. The user interface is useable, organized, and understandable.
-The Interface is Pretty Standard.
-There is the File Tab for a User to Open their music file.
-There is a Level Tab for a User to Select their Level
-There is a Bands Tab To Change Number of Bands (Visual Effect)
-There is an About Tab The Displays Quick Information About The Game.

3. The Code is Well-Structured and logically Organized
-I did my best to give Classes Appropiate Names. Enemies Have Keyword Enemy In Them. Missle Types have keyword Missle in them. Levels Are in their own classes as well. Plenty of Comments/documentation to traverse.

4. The App is not trivial.
-I think I did a good job on this app. Combines Gameplay with Musical Visuals, nice!

5. Would definitely show this to an employer.

Other:
"If you choose to develop the interface using code to create JavaFX objects rather than using FXML created by Scene Builder, then you need to present a good reason for doing so in the final project documentation."

Because This is a dynamic game (Enemies Spawn in Random Location/In Random Amounts, Player can move where desired, Missles Can be shot from 500+ positions and travel x amount of space) it would be difficult to do using a static FXML position, especially since I did a bit of threading. Also, Bands/Rectangles used as visuals are also dynamically changing in number, so would be difficult to implement via Scene builder. It would have required a looooooot of updating, which can cause lag. Speaking of, I noticed some css elements linked to FXML elements were lagging the interface as well. I did what I could with Scene Builder without going too far out of my way to do it.