Spyral-Narwhals / SoundMaze

Executive Summary

- Purpose:

  Modern Video Games are an incredibly huge market – a market that expands into new niches and communities each year. With that said, there currently are very few games aimed at visually impaired players. The purpose of SoundMaze is to create an application that is fun and accessible to both visioned and non-visioned players.

- Product:

  The most important feature of SoundMaze is our ability to directionalize sound in order to direct the player. To accomplish this, a raycasting system is used to decide where and how to place sounds. In terms of interface, the game uses the A and D keys to turn, and W to move forward.

  The goal of the player is to listen to the various sounds of a level, and find their way to a goal point while avoiding a "monster".

- Design:

  Designing the project with the Unity engine presents a problem. Using MVC architecture – as we did to an extent – is possible, but very abstracted. The Model is almost completely handled by the engine itself, leaving View, Controller, and a simplified Model to be designed. Additionally, since Unity objects require object-specific scripts, our MVC designs were more used for designing each individual object's architecture.

  - VIEW: As our game will have no graphics in the end product, our VIEW consists solely of our sound source objects and the sounds effects each object plays.

  - CONTROLLER: Handles our input – tells the Model to move the player when the A, W, or D keys are pressed. For the monster, it handles the timer that counts down to the next monster move. For rays, it controls when they get destroyed and when they are split.

  - MODEL: All that we had to use the Model for was data storage; particularly where each object is located and its various attributes (Any arrays or logic required by the object).

Most of our design comes from user stories. By focusing on what elements we felt were most required by the game, we were able to translate those stories into drawn-out designs to implement.

- Verification:

  As this project is a game, much of our verification is done through observation. However, we have some implementation tests that ensure the player can move properly. Additionally, our project runs a quick unit test on our raycasting system when it initially loads. These unit tests mainly check to ensure that our rays are properly tracing their progress through rooms and are destroying themselves properly.

- Conclusion:

  The prototype we have created is very promising. The next phase of development would be to go into beta, where we would begin Quality Assurance (QA) testing with visually-impaired players. The backbone of the game has been created, but to make it something truly great will take a huge amount of QA testing and fine-tuning.