SongRider Video Game

An interactive video game where the player must navigate obstacles that change based on tempo, rhythm, and melody of a song, while heeding incoming obstacles.

Competitive Analysis

This video game was inspired by [lumosity.com](http://lumosity.com)'s Highway Hazard game. The objective of this game is to measure “Information processing…the initial detection and analysis of incoming sensory input.” (Lumosity) To reach a high speed, the player must heed the warning signs of incoming obstacles and avoid cars already on the road. Once an obstacle is hit, the player’s speed is brought back down to 0mph, but will accelerate consistently as they avoid further obstacles. The maximum speed reached ends up being the player’s score, and determines how quickly the player can process information.

Instead of measuring the player’s ability to process more information faster, Songrider measures the player’s ability to process a song. For instance, slow classical music would be an easier “level” than a fast-paced EDM song or Nirvana. Essentially the same components of information processing are still present, however, the rate and amount of information that must be processed changes with the varying tempo and pitch of the song. So, the player is being tested on how well they understand the song. For example, if the song gets faster all of a sudden, the player will accelerate more and will have less time to avoid announced incoming obstacles, and will have to avoid “road spikes” which appear based on the current melody. The final score will calculate how well the player understood the song by considering: the number of obstacles hit and the maximum sustained speed during the difficult sections of the song.

Structural Plan

Because of the high rate at which audio must be sampled and processed, the project will be split into animations and audio processing portions. While taking heed of MVC guidelines, the model will change at the same speed at which audio is sampled. The gameplay graphics will use OOPy with separate classes for the player, the music-generated obstacles, and game-generated obstacles. An important audio function will be parseAudio(), which rescales and manipulates aubio-calculated pitch and tempo data to appropriate graphic model values.

Algorithm

The trickiest part of the project will likely be generating obstacles based on the tempo of the song. parseAudio() will return a 2D list of gaphics-friendly values calculated for each audio sample, but the space between obstacles (essentially the delay between the addition of a new Obstacle object), will be difficult to calculate. By researching digital signal processing functions, and implementing the ones that aubio doesn’t have, I am confident that I will be able to appropriately space out music-generated obstacles.

Timeline

I plan to work primarily on parseAudio() [11/20 to 11/22] until I can reliably generate appropriately spaced objects based on the tempo and pitch of the song.

Next, I will work on the other game aspects [11/23-11/25], such as the game-generated obstacles and the announced obstacles (each in their own separate class). At this point all the game mechanics will be working, but the graphics will be lacking

Finally, I will update the graphics [11/26-11/27] of the game to resemble Lumosity’s Highway Hazard, specifically changing each class’s draw functions to match the storyboard.

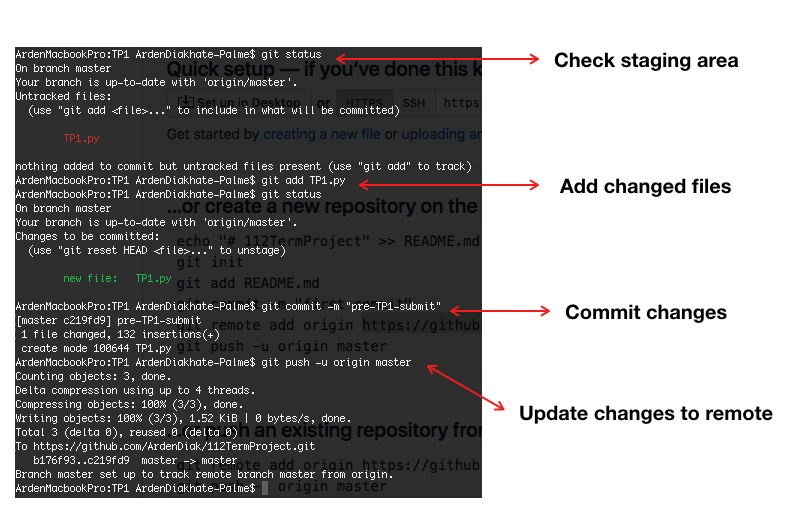
Before TP2 submission [11/27-11/28], I will retest my project, specifically each class and removing any gameplay bugs the player may run into.

Version Control

For my version control I’ve decided to use Github. I’ve created a remote repository called 112TermProject, and set it to “origin” with:

git remote add origin <https://github.com/ArdenDiak/112TermProject.git>

Here is an example of me backing up my code:



Module List

* + - * time
      * pyaudio
      * aubio
      * numpy
      * threading

TP2 Update:

In the updated version, the rate at which blocks appear and the number of blocks that appear is still based on the tempo of the song. Additionally, two powerUp elements have been implemented. The first powerUp protects the player from music blocks, allowing them to run into 5 small blocks or one large block with no score penalty. The second powerUp allows the player to shoot incoming music blocks for three seconds, however, these bullets can also destroy powerUps, so the player must aim carefully. Another significant new game element is music blocks, that when hit, accelerate both the song and the game, increasing the rate at which the player’s score increases, while also making it more difficult to maneuver around obstacles. The objective of the game remains the same, to reach a maximum speed/score at some point in the game