Charlie Shaeffer

Individual Technology Write-up

## Introduction

My third technology is a continuation of my first technology. Since I originally did audio combined with basic game logic for my first check in, I have gone back and incorporated many new features into both technologies to make them suitable as stand-alone technologies. Below I will discuss each technology in full, detailing the changes I have made to bring it to its current state.

## Audio

In its original state, audio consisted of an acceleration sound, a brake sound, a start sound, and background music. The sounds were activated on key presses, and the music would automatically begin on startup and not stop until the game was exited.

To create a more enriching audio experience, I have added five new sounds, changed the brake sound, and added three new background tracks. The sounds will play when the kart idles, when an item is picked up, when a boost is used, when a missile is fired, when a missile hits its target. In addition, the acceleration sound now loops correctly and scales its volume based on current speed, and all sounds are now triggered through the game logic. This makes it possible for sounds to be triggered by any aspect of the game. In Crusin’ Pangaea, each kart can set Booleans to trigger a sound effect when appropriate. Every frame a check is done to see if a sound should be played. If so, the location of the kart is provided and the sound is played at that location, scaling logarithmically based on distance to the player. This allows the player to hear sounds coming from behind, in front, or to the side of their kart. In addition, due to attenuation any sounds in the player’s immediate vicinity will be significantly more audible than sounds triggered further away.

There were some compromises made along the way. Most notably, other karts do not have acceleration, idle, and brake sounds. This was entirely possible with FMOD, but was not added as a design decision. The amount of noise “cluttered up” the audio and was largely unnecessary for the player’s experience. By removing these sounds, boost and missile sounds are far more audible, as well as the own player’s brake and acceleration sounds. The other compromise that was made was a lack of velocity for each sound. When using attenuation, it is possible to provide the velocity at which the sound is moving in order to determine its pitch. Again, this could have been implemented but, after doing so, it was deemed unnecessary. The crispness of the boost, item pick up, and missile firing sounds was muddled by the pitch change, so in order to keep the auditory cues the same the velocity vector was simply set to zero. This creates the same sound, still scaling with distance, but without the change in pitch.

To summarize, the audio has gone from four sounds for the main player’s kart to nine sound effects, many that can be heard throughout the world, along with four background music tracks that are triggered in different sections of the game. From start screen to end screen, audio is now a key facet to Crusin’ Pangaea.

## Game Logic

Steven and I implemented rudimentary game logic for Crusin’ Pangaea for our first technology, but it was straightforward. Players moved along the spline and their progress was tracked to determine their place in the race. While effective, there was room for improvement. Since the first check in, I have added placement of players on the track when they fell off (with help from Robert Burton), checking power ups for each kart, and better tracking of players moving backwards on the track.

When a kart falls off the track, there was no way to get the kart back on the map without restarting the game. This was unacceptable, and as such I created a function called checkKarts() which checks to see if any kart is far below their current spline point or if they have requested to be placed back on the map (in case they get stuck or some other incident occurs). My initial code placed the player back on the map at the spline point closest to them, and Robert Burton expanded upon this to put the player in the right direction as well as make it accessible by the AI.

The next addition, checking for power ups, was done in the scene itself as well as the game logic. In the scene, a uniform spatial subdivision is used to keep track of the power ups placed throughout the map. Each frame the karts are iterated through, and if they are in a section with a power up (and they are close enough) they are given a power up. Upon activation, a Boolean is triggered. The triggered Boolean tells the game logic to look at that particular kart and activate its power up.

Lastly, the game logic’s method of tracking players moving backwards was made more unforgiving for the most mischievous of players. As a player moves backwards through the map, if they complete an entire lap backwards they now have to “make up” that lap before progressing to the next lap. This was added mostly for humor, but also because nobody likes those people who travel backwards to hit other players and sit on item spots to shoot missiles at everyone else.

## Summary

As seen above, audio now has accurate attenuation and a multitude of new sound additions, and game logic has been improved to account for power ups and for regions of the map where one can fall such as bridges and pits, or in the event a problem occurs.