Project Proposal

Isabella Rhee

**CMU Buggy Racers**

Cmu buggy racers is a 3rd person racing game that includes features such as random track generation, opponent AI, and side scrolling. Players will be able to use the keyboard to “race” their buggies around a track. Edges of the track and other obstacles will slow them down while staying on the track will increase speed slowly. The goal is for the completed project to have background music, character choices, and a competitive AI opponent. Also, I would like to be able to have pictures for the “buggies” that look like CMU buggies.

* **Competitive Analysis**

The main objective of myproject is similar to some I’ve seen online. I think that some programs have their students build racing games as a way to learn about making AI’s. However, those usually use modules for the graphics or are given a track that they need to make a car drive around. My project is a little bit more involved than these in that I am generating a track and coding all of the car mechanics and such. Theirs looks prettier though with fancy graphics.

From what I’ve seen on the interweb, a lot of projects that work with making an AI racing game use reinforcement learning. They build a model based on images of track and what the human would do to teach the computer to drive around it. The other method I saw was pathfinding, which is what I think I would like to do with mine. Basically you have the current location and like a goal point and have the computer find a path from point A to point B, then follow that path to reach the end.

* **Structural Plan**

MyModalApp

I’m using mymodal app classes to create the different screens for my game. Most of the code will be in the GameMode class.

StartMode

This is just a splashscreen for the game that goes to the MenuMode when any button is pressed. I

plan to put a picture of CMU or of Buggy on this screen.

MenuMode

This will be used for any extra screens I need like instructions, picking characters, and starting the game.

GameMode

This is where the actual game code will be. I will draw the track and racers here, as well as do controls for speed and stuff.

PickPlayerMode

Just going to have like different colored buggies to choose from. Has a back button to go back to the menu.

Racer class

This will build the characters, both the player and the opponents. Opponent will be a subclass of racer most likely. Contains information such as the name, color, coordinates, x and y speeds and pictures.

Create Track

I basically used the maze solver code as starter code for this and adapted it to fit my own needs. This just like takes the empty grid and produces and random track starting from the bottom middle and ending at the top middle of the grid.

* **Algorithmic Plan**

creating a track

For this I used backtracking logic like solving a maze. Also, I make sure to check if a box is next to one that’s already in the visited list to it’s more of a path shape and avoids making “blobs” of track. Also, I make sure to shuffle the list of possible directions to try to avoid creating the same path every time.

Opponent AI

This will probably use maze solver code as well, but it’ll have to be like by pixels rather than squares on a grid just because of how small the player is in comparison to the grid. I will probably have it check to see how far it is from the wall on all side and move towards the side where the wall is farther away. The tricky part will be differentiating which way is going closer to the end of the track and which is the opposite way. Also, the speed control has to be similar to that of the player which shouldn’t be too bad because I already coded it. It will use the same friction coefficient and add the same number of scrollX and scrollY when it updates. The hard part will just be making it be able to navigate the track on its own. One way I might do this is program the end coordinates of the finish square into the AI data and then have it check if it’s getting closer to it as it moves. However, this might not work if the track loops around or does a turn. Another way I could do it is like store the path in the AI memory like a line down the middle and have it follow that line, and if it starts to go off the path readjust the direction so it stays on.

Controlling the player

The arrow keys are used to control the players “movements” which is shown with the side scrolling. There is a variable for how much it should scroll for both the x and y components, which can be maxed out. Also, as the timer fires, these variables subtract the value “friction” making it slow down over time and come to a stop. Currently working on keeping the player within the walls of the path, making it come to a stop if it runs into a wall. The player is represented with images of the “buggy” instead that I made with Snapchat lol. There are 5 different pictures to show different angles, sideways left, turning 45 degrees left, straight ahead, turning 45 degrees right, and sideways right. I have a variable number that is changed every time an arrow key is pressed to represent the direction that the buggy should be facing.

* **Timeline Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 11/29 | 11/30 | 12/1 | 12/2 | 12/3 | 12/4 | 12/5 |
| Player able to drive around track with a finish line | Random track generator done |  |  | AI component working | Pictures/graphics | MVP due |
| 12/6 | 12/7 | 12/8 | 12/9 | 12/10 | 12/11 | 12/12 |
| Adding items | Adding music |  | Due date |  |  |  |

* **Version Control Plan** [1.5 pts]:

I’m using Github to back up my code.

Graphical user interface, text, website

Description automatically generated

A screenshot of a computer screen

Description automatically generated

* **Module List**
  + N/a

Sources:

http://www.cs.cmu.edu/~112/notes/notes-animations-part3.html#sidescrollerExamples

<https://twitter.com/cmubuggy>

<https://www.cmu.edu/brand/brand-guidelines/visual-identity/colors.html>

TP2

Design updates:

AI algorithm:

My AI basically has a list containing visited cells in the grid and once it gets in one, it adds it to the list. When the car gets to the center of the cell, it checks in all directions around it to see which cell next to it is not in the visited list and is a cell that is part of the path. Then, it moves it in that direction. Repeat. Boom.

Other:

I made the pick player mode the second screen the player sees. If I didn’t do this it’s possible the user would never enter this mode and the color variable would never be instantiated in the GameMode appStarted function because I was using the input from pick player mode to make the Racer object.