Final Project Game Design

Game Idea: Light Racers

Authors: Dalton (Cale) Cohee and Douglas Easom

Game Objective: Light Racers is a *Tron*-inspired light cycles game, which can act as a variant to the *Snake* game concept. In the game, players traverse through the game map while leaving a trail behind them. The objective is to destroy other opponents by forcing them into one of the trails left behind players. The game is both a combination of survival through avoiding all trails and obstacles, and strategy in trying to box in other opponents.

The game is centered around user vs. AI for competition. This project is an expansion of the basic *Tron* arcade game through allowing different maps and play areas, changing the number of players, offering different game modes, and customizing your light racer.

Game Modes: Light Racers will feature the following game modes:

- *Survival:* Each competitor is focused on remaining alive longer than all other competitors. Players must survive through avoiding other light trails, boundary walls, and map obstacles. Each car has one life and is removed from gameplay as soon as they are destroyed. The player to survive the longest wins.
- *Lives:* Each competitor is given a certain number of lives at the beginning of the round. The objective is to diminish other competitor's lives while holding onto as many of theirs as they can. If a car is destroyed with lives still remaining, it loses a life to respawn again. It a car is destroyed with no lives remaining, it is removed from the game. The player to out-live all other competitors wins.
- *Time:* Each competitor respawns infinitely for a set time duration. The objective is to destroy the most light racers before the time is up. Players will be given 1 point for each racer they destroy and have 1 point deducted for each time they destroy themselves. In the event of a draw, each tied player will be respawned once more and last one standing wins.

Main Options: From the game's main menu, user's has the following options to select:

- *Play:* Selecting play begins a new game round for the user to play. The round remembers and includes all the user's selected settings such as the game map to play on, the difficulty, the game mode, the number of players, the selected song, etc.
- *Instructions:* Selecting instructions will take the user to a new page which displays a detailed outline of the game's rules. Instructions detail game objectives, game controls, and distinctions of game modes.
- *Settings:* Selecting the settings page allows the user to change details about the next round they play. Options that they can choose between include the number of players for a round, the song being played in the background, and the difficulty.
- *Customize:* Selecting on customize grants the user ability to customize and change their light racer. Customization includes altering basic color scheme and design.
- *Game Modes:* Selecting game modes allows the user to change the game modes *Survival, Lives,* and *Time* for the next round. It also provides a detailed description of what distinguishes each game mode and the rules to abide by.

Project Classes: This game is designed using the object-oriented programming approach. It is separated by these classes:

- *Light Racers Class*: This class acts as the game's "main". It provides little functionality to the program aside from acting as a source to compile all other classes together into a working project. Much effort is being done to keep this class as concise as possible so that all other classes are doing the required backend.
- *Game Class:* This class functions as an umbrella and parent class to all other game sections. The data fields for this class include basic game settings that must be kept track of in game play, such as the number of players, the difficulty of the round, if the game is currently in play, the songs being played during the round, and a data structure to store the position history every player, etc. It simply provides the information required for each round of gameplay. Most methods within this class are accessor and mutator methods to retrieve information about the rounds settings.
- *Maps Class:* This class is purposed with displaying each play arena available in the game. The data structures for this class include the width and height of the play area, the level that has been selected, the font for text, and such. Many of the methods within this class are used

to display the maps for users to operate on. It also contains functions to indicate when cars have exceeding the boundaries of the wall or encountered obstacles that are placed in a specific map.

- *Menu Class:* This class is responsible for displaying all menus and screens when not in gameplay. It contains data fields to keep track of which screen is being displayed, the music to be played at each menu, the font for all text, and pictures to be used in the background of menus. Many of the methods within this class are display methods. Different methods are used for drawing the main menu, the settings screen, the instructions screen, and customization screen, the game mode screen, and map selection screen. It also contains mouse click methods corresponding to each of those screens so that the appropriate action is performed if a mouse is pressed on a specific screen.
- *Racer Class:* This class creates an object for each car used in the game. It's data fields contains basic properties such as the color of the car, the car's (x,y) location, the car's (x,y) velocity, and an indicator for if the car is still destroyed or not. The class has accessors and mutators for these fields. It also contains methods for displaying the car and updating the car's position. Finally, the class contains a key pressed function for deploying the appropriate action for when the arrow keys are used to navigate the car.