Light Cycle

Kole Cralley

September 10, 2019

1 Introduction

1.1 How the game is played

Light cycle is a game played between two opponents on an n by n grid. The goal of light cycle is to stay alive longer than your opponent. When a cycle moves it can move right, forward, or left. When it moves it leaves a wall behind in the space it moved from that can be crashed into by either light cycle. Each light cycle must move one space each round and move at the same time each round. The game continues with light cycles moving until:

- At least one light cycle moves out of bounds
- At least one light cycle crashes into a wall
- The two light cycles try to move to the same spot in the same round

If both light cycles die on the same round it is considered a tie.

1.2 Grid Properties

The grid is an n by n discrete grid. The top left of the grid has a zero x and a zero y position. The bottom right of the grid has an x position of n - 1 and a y position of n - 1. If a bike tries to move out of the n by n grid it is considered out of bounds and it is considered dead.

1.3 Light Cycle properties

The light cycles have three properties. The first property is the direction the light cycle is facing. This can be North, South, East, or West.

- North refers to the top of the grid and is facing forwards row zero.
- South refers to the bottom of the grid and is facing forwards row n - 1.
- East refers to the right of the grid and is facing towards column n - 1.
- West refers to the left of the grid and is facing towards column 0.

The second and third properties of the light cycle are the x position and y position of the cycle. To move the bike you give it a command using the notation left, forward, or right. Please note that this is relative to the direction the bike is facing. For example, if a bike is facing south and moves left it will be facing east.

1.4 Start Configuration Example

When a game starts the two light cycles start in the top left and bottom right corners. Which corner a bike starts is random each game. If the bike spawns in the bottom right it will start facing north. If a bike spawns in the top left it will be facing south.

B				
				R

Table 1: Example start configuration

1.5 Example Game

Table 2 is an example of a game on round 2. On round zero the light cycles were placed in the top left and bottom right represented by B and R respectively. On round two both cars moved forward. When they move forward a wall forms behind them represented by W. Then each car turned left. When they turned left they still moved to a new space, leaving a W behind them. Notice that left is relative to the direction the light cycle is facing.

W				
W	B			
			R	W
				W

Table 2: Round two game

2 Server Client Interaction

2.1 Server Design

There are two parts to the design of the game: the server and the two clients. Each of the two clients control one of the light bikes. At the start the server forks both of the client executable files into two sub processes. These sub processes run for the whole game, so it is important to make sure your client does not close before the game is over. If your client closes (i.e crashes) before the game is over then it counts as a death.

2.2 Client Setup

When the server starts the client process the client should write the line "ready" to its standard out and then expect a line for grid size on its standard in. The server will wait for a predefined amount of time for the "ready" response and if it does not receive it that client will be considered dead. The grid size will be a string containing an integer. This set up will only happen once at the start of a game.

2.3 Client Ongoing Communication

At the start of each round the server expects a line containing "ready" to be sent. If it is not sent in a predefined amount of time the server will count that client as dead. After the client says it is ready the server sends the client the current state of the board. How the state is transferred is in the following string format: "your direction, your x position, your y position, opponent direction, opponent x position, opponent y position". So if Grid size is 10 and your client spawned in the top left the first state string would be "s,0,0,n,9,9". After the state information is sent the server expects a response in a predefined time. The response should be how the light cycle should move in the form of a line "left", "right", "forward".

2.4 Example Communications between a client and the server

- (client to server) ready
- (server to client) 10
- (client to server) ready
- (server to client) "s,0,0,n,9,9"
- (client to server) forward
- (client to server) ready
- (server to client) "s,0,1,n,9,8"
- (client to server) right

2.5 Python example of a client that only moves forward

```
#!/usr/bin/python3
gridLength = int(input("ready"))
while True:
    state_string = input("ready")
    print("forward")
```

3 Where to start

3.1 Make a simple client

Make a client that will only move forward and see if it works with the server. This will help you understand how to have your client communicate with the server.

3.2 Keep track of the map

Most clients will need to keep track of the what has happened on the board. This can be done in an array or a list. Keeping track of the board can be used to keep track of where the opponent is and where walls are.

3.3 Plan a strategy

Think about how your client is going to make decisions. Is it going it be a hard coded set of rules or will it use reinforcement learning to learn to play on its own? Will it play aggressively trying to block its opponent or will it play defensively trying to maximize the distance it can travel in the future.