TREVIAN JENKINS
ENPM690 – HW3
ROBOT SIMULATION

SUMMARY:

This project is a simulation of a "car" that can move around in its environment by driving forward, backward, and by turning its orientation. The robot has two degrees of freedom of movement: steering and driving. The steering is performed at a rate of 180 degrees per second in either direction, while the robot's speed, while it is moving, is constant at 192 pixels per second. Control of the vehicle can be toggled between the human user and the robot. While the robot is human-controlled, it is blue, and when operating autonomously, it is red. The map is a construct of 64 x 64 pixel blocks that the robot cannot drive through. The robot also has a circular collision mask with a radius of 32 pixels.

CONTROLS:

- <up, down>: drive forward and backward
- <left, right>: steer
- <T>: toggle control between human and robot

DESCRIPTION OF AUTONOMY:

The robot is allowed to perform the same movements as the human, with the minor exception that while the robot may physically turn and drive simultaneously, there is not currently a need to do so, nor is there logic in place for the robot to perform this action. The robot can also sense the presence of obstacles directly in front of it, within approximately 100 pixels. The robot moves straight until it senses an obstacle. At this point, the robot turns toward a random direction. If it is free to move in that direction, it will continue to go straight. Otherwise, it will turn toward another direction, and this process repeats. In this mode, the robot has a visible line in front of it indicating its range of sight.

DEVELOPMENT:

This project was developed using GameMaker Studio 2, an engine primarily purposed for the development of 2D games. I drew the sprite myself, a simple polygon for a chassis with two elliptical wheels.

I will consider updating this project to Pygame to utilize Python for development if we use this project as a basis for future application of robot learning in this class. Initially, I wanted to use python for this simulation. However, I have used GameMaker for quite some time now and have gotten very familiar with it, so it was my preferred approach to create a proof-of-concept. I am also familiar with python, but I have never used Pygame, or any python graphical interface involving keyboard/joystick input, so such a transition, while feasible, would involve learning how to use such a new tool, and I am looking forward to it. As I become more familiar with ROS, it will become my primary tool for such proof-of-concept simulations later this semester.

SCREENSHOTS:

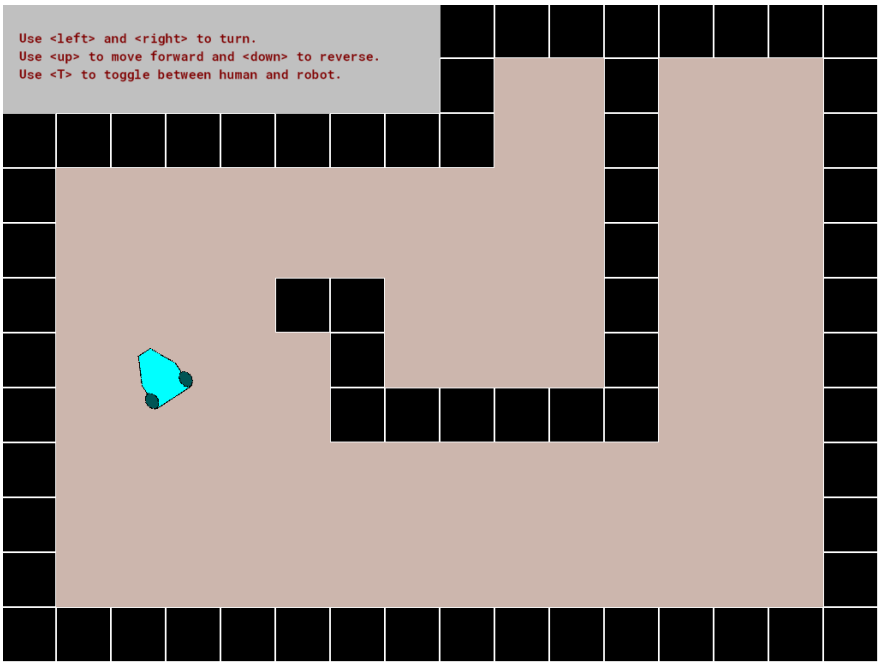

Fig. 1: Human-controlled mode.

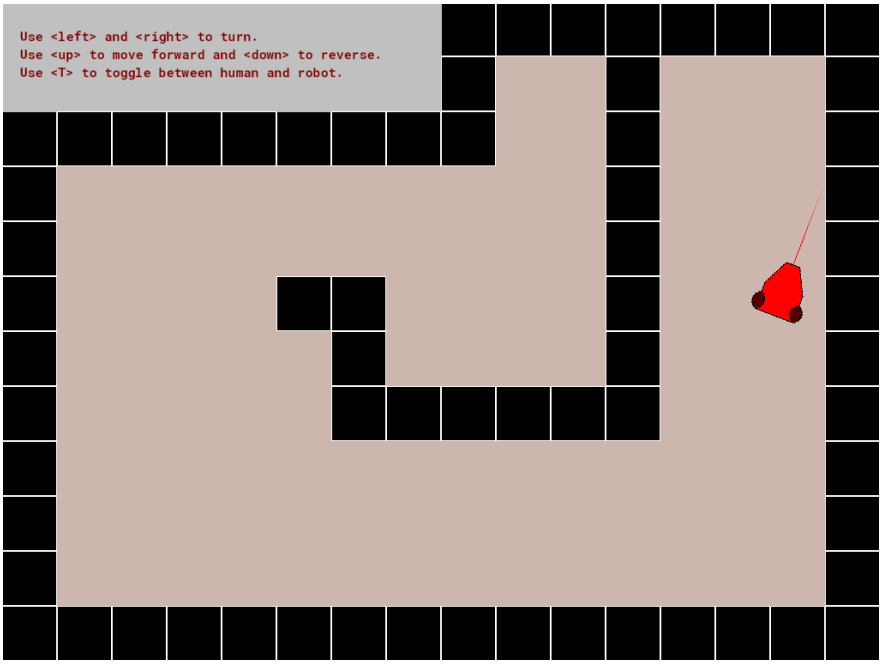

Fig. 2: Autonomous mode. Note the visible line-of-sight.

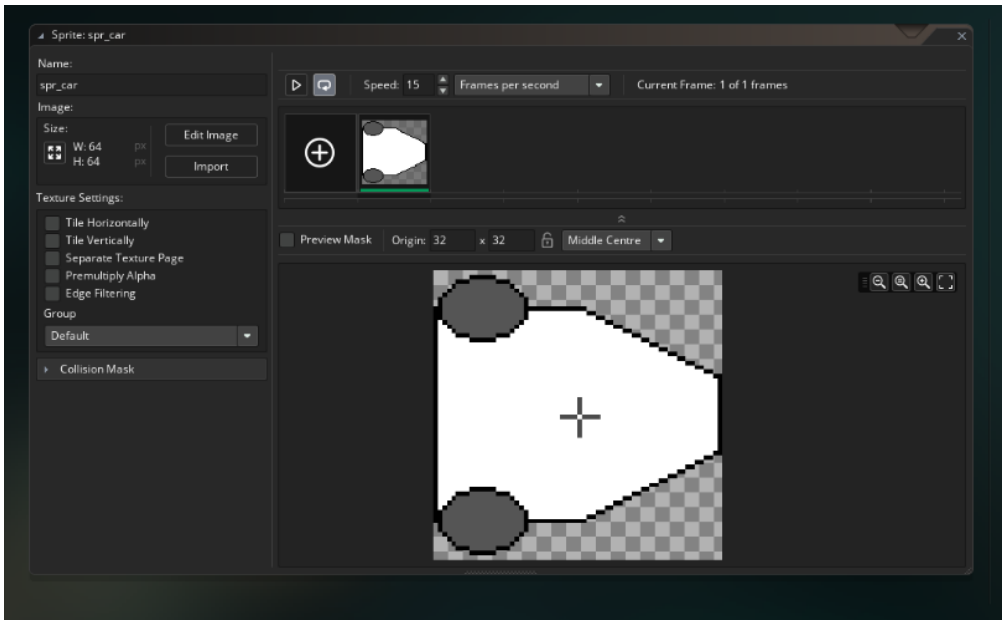

Fig. 3: GameMaker Studio IDE for sprite development.

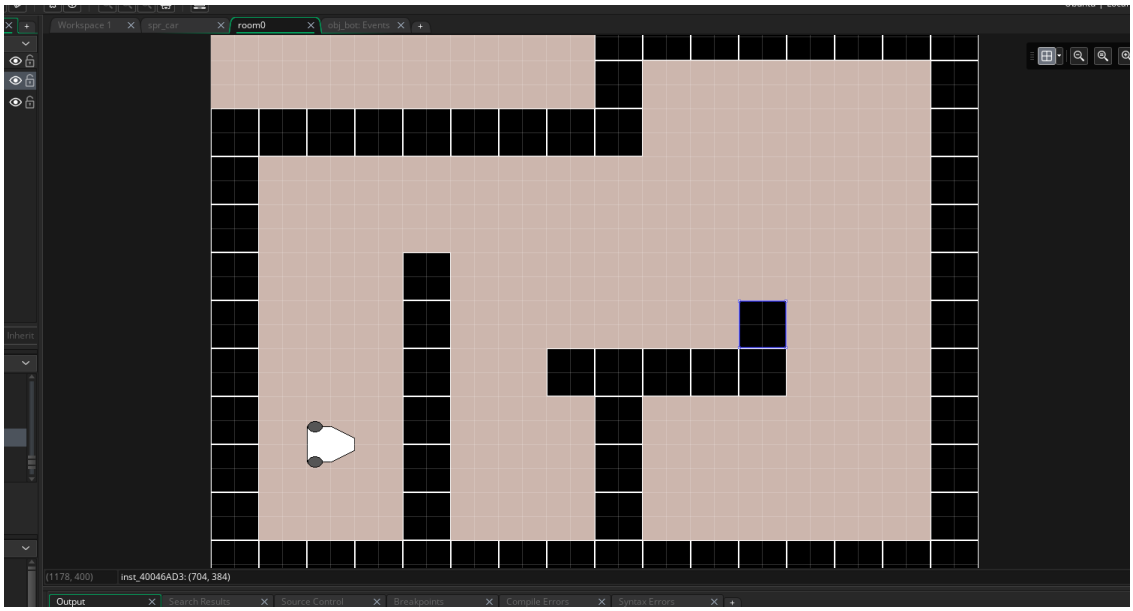

Fig. 4: GameMaker Studio IDE for room design
(slightly different design from final room, as in Fig. 1 and Fig. 2).

ISSUES AND IMPROVEMENTS:

Since the robot can only sense in a straight line directly in front of the middle of its heading, it is prone to clipping the walls. At first, the robot would get stuck at the wall and cease to move. I handled this by having the robot perform the same action as it would normally when detecting an obstacle, so that the simulation could continue. This could be improved by setting a rectangular or triangular range within which the robot can sense obstacles, rather than a line.

The robot also moves rather inefficiently since it stops to turn, rather than driving and steering simultaneously. This prevents the robot from colliding with walls needlessly, but an additional layer of sophistication of the robot's logic could help it learn to navigate spaces quicker without needing to stop. The robot's logic is also limited to moving forward. Since it can turn, it currently has no particular need to reverse, but to save the time it takes to turn 180 degrees (about a second), the robot could reverse in the future. Some other more complicated maneuvers could be constructed with reversing, and additional logic to help the robot learn when to reverse would save a significant amount of time.

LINKS:

- GameMaker Studio 2 manual: https://docs2.yoyogames.com
- Github link: https://github.com/trevian2345/ENPM690_HW3
- Video link: https://drive.google.com/open?id=1yMM2B5jCFRXXwWkWceE66nqeV5y_NoL-