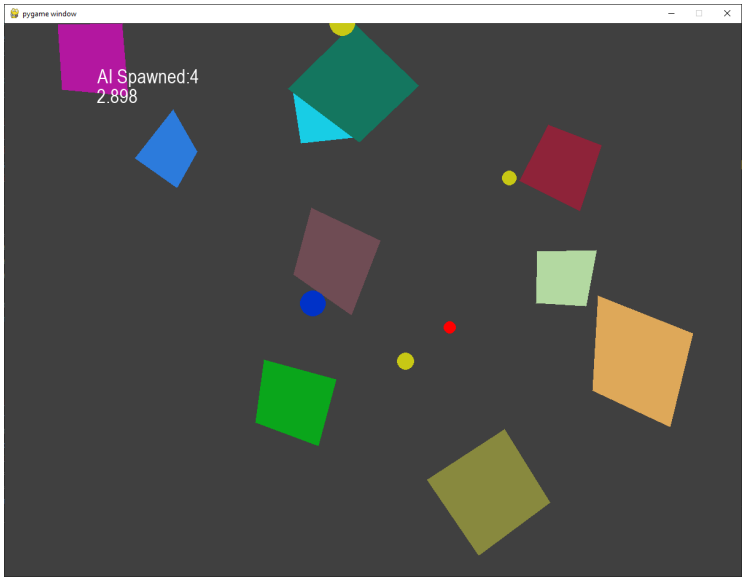
Final Project

Jacob Justice

https://github.com/JacobJustice/HideAndSeek

Overview

For my Final Project I decided to implement a Hide and Seek game, with obstacles of an arbitrary shape and AI agents that try to chase the player down. When an enemy agent collides with you, you lose, and are given the option to replay with the same obstacles.



Approach

When the game is loaded, 10 random obstacles are generated. These are generated using random angled steps along the circumference of a circle. Two parameters “irregularity” and “spikiness” allow me to alter the jaggedness of these polygons.

AI Agents spawn every 10 seconds at the edge of the board. This way the player is incentivized to stay near the middle of the board, where it is most difficult to avoid getting surrounded.
The AI Agents have 3 types:

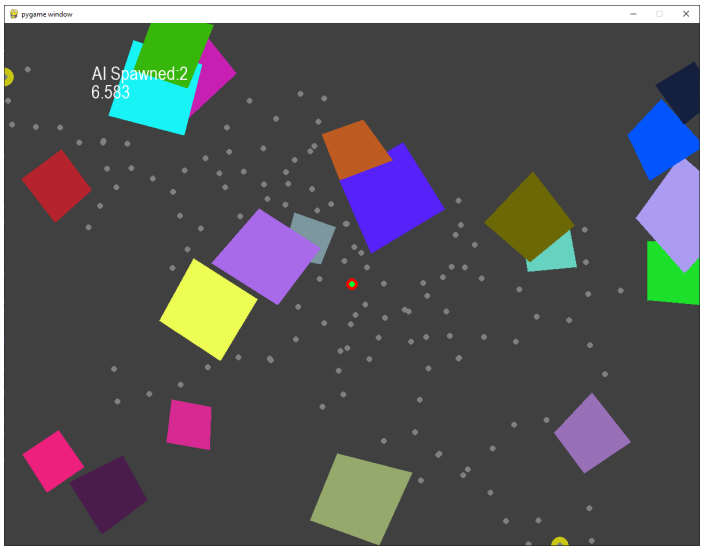
- SimpleAI: moves towards you in a straight line, sliding along obstacles (in blue)
- RandomAI: at each time step samples 20 random possible moves at maximum speed, and picks the one that moves closest to the player(in yellow)
- RRTAI: uses rapid random trees, with a new node being added at each time step, to learn ways to get around obstacles (yellow, shown in figure 2)

SimpleAI and RandomAI work as intended, although SimpleAI has a tendency to get stuck on corners. RRTAI is very close to working, but unfortunately is not in the current state.

When colliding with an object, entities slide along the projection of their velocity vector, with the nearest edge. This makes movement feel much more fluid and responsive, as opposed to simply stopping once you touch an obstacle.

Current Bugs and Limitations

Unfortunately, RRTAI never move, as once they reach their first node, they never start moving toward the next node. I was unable to fix this due to time constraints.



Collisions can be a little buggy around the vertices of the polygons. If the game thinks the nearest edge is as an edge on the other side of the polygon, then the player will not be able to move in their desired direction at all. This doesn't really affect gameplay, as the player can just move in any other direction.

I tried to implement a Probabilistic RoadMap AI as well, but it led to an extremely long loading screen at the very beginning to compute the roadmap, so I went with the RRT approach instead, since the tree would grow per timestep.

Future Work

The first thing I would do is fix the RRTAI, as that is the most interesting and would enhance gameplay significantly. Other gameplay features I would add are settings to change the number of obstacles, the size and shape of obstacles and the rate of new AI spawning into the board. I would also like to add a feature where AI have a small cone of vision, allowing the player to truly hide from the AI.