CS4303 Geometry Wars Report

140011146

November 8, 2017

1 Introduction

In this practical we were tasked to implement a top-down shooter game, similar to games like *Robotron 2084* and *Geometry Wars* with the goal of implementing a variety of AI algorithms. In my game, I have implemented six different types of AI, including a flocking AI. I have also implemented multiple different pickups and the ability to play over a network.

To build the game, run `ant` in the submission directory.
To run the server, run `ant -Darg0="Server" server`
To run the client, run `ant -Darg0="Name" client` where "Name" can be any name.

2 Game features

asd

2.1 AIs

2.2 Weapons

2.3 Pickups

2.4 Networking

3 Design and Implementation

3.1 AIs

3.1.1 Basic chase

This is the most basic AI that always goes towards the current position of the player. If the player is always moving, this AI will always just be chasing behind the player, making it quite easy to run away from.

3.1.2 Circle

This AI is similar to the basic chase AI except they use acceleration in their movement rather than moving at a constant speed towards the player's position. This means that if the player moves past the quickly, they will take more time to turn around and follow the player again.

3.1.3 Ambush

This AI attempts to move towards where it predicts the player will be based on the direction the player is moving in. It will move to a position in front of the player.

I found when first creating this AI that if it implements this behaviour and there are a lot of them, they will all move synchronously when the player changes direction, making them easy to predict for a player. To deal with this, I added a random delay before they change their target position making each of them act more independent.

3.1.4 Patrol

These AI are spawn in with the generated pickups and will patrol in a square around the pickups. If the player comes near they will chase the player for a certain distance before moving back to their patrol locations.

3.1.5 Shoot

This AI will shoot back at the player, making it more difficult. I chose to make them shoot directly at the player position rather than randomly near the player so it is easier to dodge if the player is always moving. These AI will move randomly around the screen while shooting.

3.1.6 Flocking

I have also implemented the flocking algorithm following the example on https://processing.org/examples/flocki

Initially I had these enemies move towards the player position if there were no nearby flockmates but this made the game very hard as they would all flock towards the player. Instead, their initial target position is the player's current position and there is no other target direction or velocity when there are no nearby flockmates.

These AI are also smaller and more are spawned in the game to show off their flocking behaviour without filling the whole screen.

3.2 AI Director

The AI director uses [1].

3.3 Factories

3.4 Network

To implement the networking in the game, I use a processing UDP library (https://ubaa.net/shared/processing/udp/).

UDP packets are sent on every frame of the game. The server will send the entire game state and let the client render the state. Clients only send the player input to the server to process. This means the server is always correct and would prevent client players from cheating their attributes such as position and health. The trade off is the server has to do a lot more processing and so the game cannot scale to having many players at once.

3.5 Pickup

4 Conclusion

References

[1] Michael Booth. *The AI Systems of Left 4 Dead*. Valve Software, 2009.