CS4303 Geometry Wars Report

140011146

November 9, 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. There is also an AI director that controls the difficulty and pacing of the game. 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 and design

2.1 AI Enemies

2.1.1 Basic chase

Figure 1: Basic Chase Enemy

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 be chasing behind the player, making it quite easy to run away from.

2.1.2 Circle

Figure 2: Circle Enemy

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 reverse and follow the player again.

2.1.3 Ambush

Figure 3: Ambush Enemy

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.

2.1.4 Patrol

Figure 4: Patrol Enemy

These AI 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.

2.1.5 Shoot

Figure 5: Shooting Enemy

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.

2.1.6 Flocking

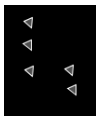

Figure 6: Flocking enemy

The flocking algorithm was implemented following the example on https://processing.org/examples/flocking.htı
[2].

Initially these enemies moved 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.

2.2 AI Director

2.2.1 Pacing

The AI director uses an adaptive pacing algorithm following the system used in Left 4 Dead [1]. The idea is to have bouts of high intensity and low intensity, the durations of which change dynamically. The "intensity" of players is roughly estimated:

- Increase intensity when the player takes damage, proportional to the damage.

- Increase intensity when the player kills an enemy, proportional to the score value of killed enemy.
- Decrease intensity over time.

If the intensity is too high, no shooting enemies are spawned to make the game a little easier for the players.

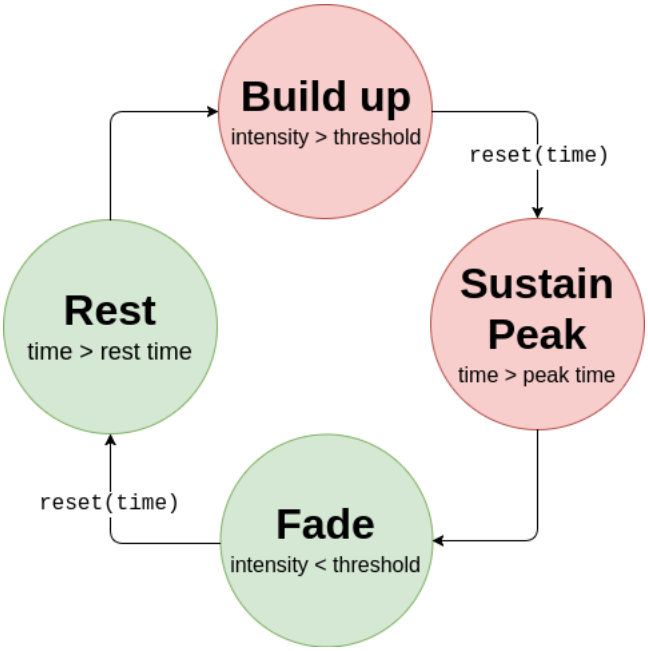

Figure 7: State machine of the AI director.

The **Build up** and **Sustain Peak** states have a higher spawn rate of enemies. The director moves from build up to sustain peak when the intensity is greater than the threshold. The sustain peak state is there to ensure there is some minimum time spent with high spawn rates of enemies.

Fade and **Rest** states are the opposite side to build up and peak. The director moves out of fade when the intensity is below the threshold and rest acts the same as peak, ensuring a minimum period of rest before the next build up.

2.2.2 Difficulty

At regular intervals, the state of the game is sampled to see how well the players are doing and adjust the difficulty accordingly. By default, the difficulty slowly increases at each interval to progressively make the game harder.

An increase in the difficulty value of the game affects the following parameters:

- Increased enemy health
- Increased enemy damage to players
- Increased enemy score value
- Increased enemy spawn rate
- Decreased pickup spawn rate
- Decreased pickup lifespan

2.3 Weapons

The player is equipped with three different weapons that can be quickly swapped at any point. Each weapon has a separate ammo and reload system, making it convenient if the player runs out of ammunition on one weapon to switch to another rather than wait for the reload.

2.3.1 Pistol

Figure 8: Player with pistol equipped

The pistol is a simple weapon with low clip size and low fire rate and faster reload.

2.3.2 Rocket

Figure 9: Player with rocket equipped

The rocket is a powerful weapon that creates an explosion on impact, dealing lots of damage but it comes with extremely low clip size, low fire rate and slow reload.

2.3.3 Machine gun

Figure 10: Player with machine gun equipped

The machine gun shoots bullets quickly with a larger clip size and high fire rate but takes longer to reload.

2.4 Pickups

2.4.1 Health pickup

Figure 11: Health pickup

The health pickup gives the player more health. A player is able to go over the maximum 100 health when picking up these pickups to act as a buffer for more health, but this will signal to the AI director that the player is doing well.

2.4.2 Ammo pickup

Figure 12: Ammo pickup

The ammo pickup increases the ammo of each of the player's weapon by one clip. It is important to pick up these, otherwise the players will eventually run out of ammunition.

2.4.3 Speed pickup

Figure 13: Speed pickup

This pickup temporarily increases the speed of the player who picked it up. This is useful to run away or reach a far pickup quickly, but it is a double-edged sword as the player can easily run head-on into a group of enemies.

2.4.4 Pierce pickup

Figure 14: Pierce pickup

The pierce pickup will allow the player's bullets to pierce through all enemies. This does not apply to the rocket weapon.

2.4.5 Damage pickup

Figure 15: Damage pickup

This pickup permanently increases the player's damage. This helps the player deal with the increased health of enemies from increasing difficulty.

2.4.6 Bullet radius pickup

Figure 16: Bullet radius pickup

This pickup permanently increases the player's bullet's radius. This helps the player with easily aim as the larger bullets will catch enemies easier, especially when trying to shoot the flocking enemies. This increased radius also affects the rocket's explosion size, as it scales off the rocket bullet's radius.

2.4.7 Bomb pickup

Figure 17: Bomb pickup

This is a powerful pickup that will clear the entire screen of enemies, giving players a brief respite.

2.5 Networking

The game can be played over the network, allowing multiple players to play cooperatively.

3 Implementation details

3.1 Factories

3.2 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.

3.3 Pickups

4 Conclusion

References

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

[2] Daniel Shiffman. *Flocking*. `https://processing.org/examples/flocking.html`.