Endless Wasteland

By Splitz Productions

Who am I?

- Alex Davis, 22, 3rd Year of University.
- Sole member of Splitz Production.
- Solo Developer for Endless Wasteland.
- Managed the code, documentation, agile process, design ideas for the prototype.
- Created games/prototypes such as Another F***ing Zombie Game, Block Assault and Apartment Escape.

What is Endless Wasteland?

- Endless Wasteland is about a survivor in a desert wasteland accompanied by their trusty companion; a dog.
- The objective of the game is to survive multiple waves of enemies while trying not to die.
- You have a pistol with limited amount of ammo and a lazer pointer to tell you dog to attack enemies.
- The dog is controlled by an AI.

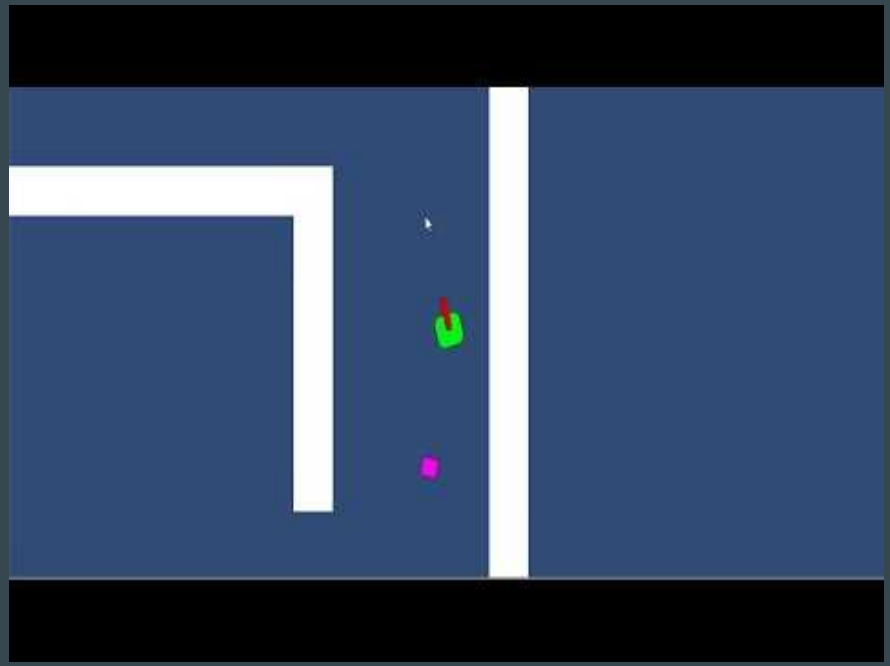
Companion AI

Research into Companion AI

- My individual research was based on the different types and approaches that developers employ for companion AI.
- My research explored different AI from games such as The Last of Us, Bioshock Infinite and God of War.
- My conclusion was that there was no wrong way to make a compelling companion AI and that there are multiple ways to create AI that engages with the player and helps them as well.

Dog AI Version 1

- Used trigger colliders to act as radars for the dog to follow explore points, attack enemies and collect items.
- The pathing was done via a A* Unity Package.
- There was no player control over the dog AI.
- Uses explore points inspired by Ellie's AI in The Last of Us.



Evaluation of Version 1

- The Player would be frustrated if they were attacking an enemy and the AI went off and did something else.
- Very barebones.

NO DIRECT PLAYER CONTROL!

Dog AI Version 2

- Based on the foundation of Version 1
- Uses States to accurately focus on one specific action that the player chooses.
- The Player can select which state the dog should be in via the number keys.
- 1 = Alert/Attack 2 = Search 3 = Fetch
- This gives the player more control over the AI which makes it less stressful for the player

```
void Update()
{
    if (findMe.target == null)
    {
        state = DogState.DEFAULT;
        findMe.target = player.transform;
    }

    if (Input.GetKeyDown(KeyCode.Alpha1))
    {
        if(state != DogState.DEFAULT)
        {
            return;
        }
        else
        {
            state = DogState.SEARCH;
            Debug.Log("search");
        }
    }
    else if (Input.GetKeyDown(KeyCode.Alpha2))
    {
        if (state != DogState.DEFAULT)
        {
            return;
        }
        else
        {
            state = DogState.FETCH;
            Debug.Log("fetch");
        }
    }
}
```

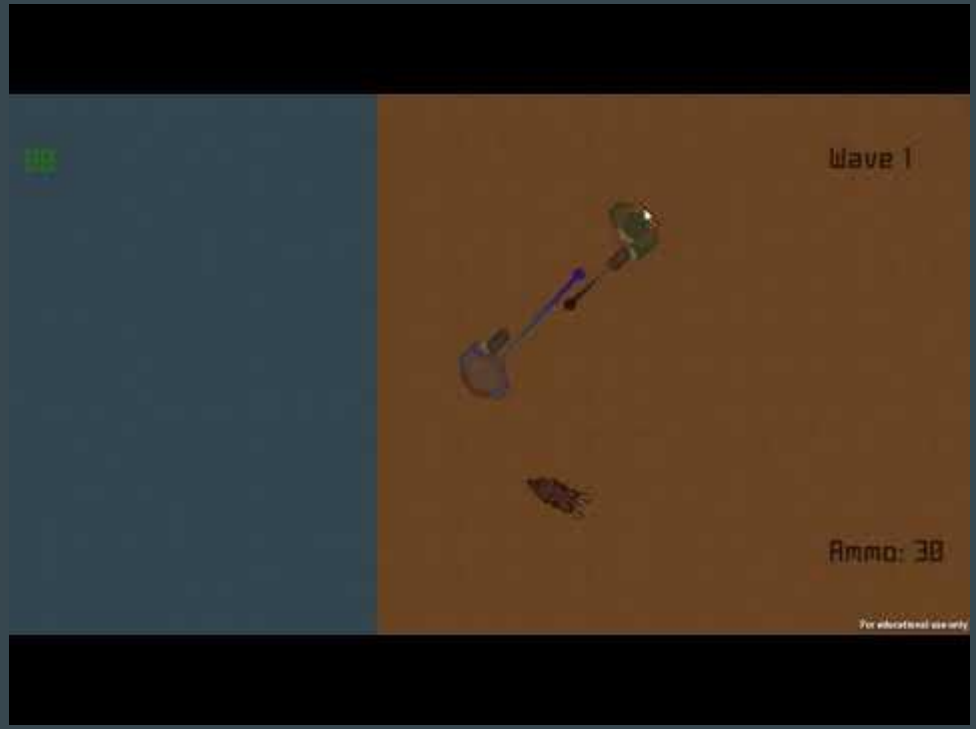
Evaluation of Version 2

- This evaluation was based on Usability Testing.
- With how the radars are set up, it doesn't work immediately and the dog has to walk into the radar for the dog to search/attack/fetch.
- Attacking didn't work 75% of the time.
- Once you pressed a state key, you can't choose another one unless the action has been completed.

NEEDS MORE DIRECT CONTROL.

Dog AI Latest Version

- Using raycasts, when you click the right mouse button, it'll shoot a laser that if it hits an enemy it would attack that enemy.
- Gives the player the ultimate control over how the dog attacks.
- Based on Version 2



Evaluation of Latest Version

- Raycast is finicky so sometimes the lazer doesn't work.
- The issues regarding the states are still persistent.
- Player has a decent level of direct control over the AI without the AI just being completely controlled by the AI
- The dog is helpful with attacking enemies
- Unsure if the explore point system is actually useful (potentially scrap the idea).

Gameplay

Gameplay of Endless Wasteland

- The player can shoot enemies.
- Enemies spawn via increasingly difficult waves.
- Enemies are either melee based, tanky based or ranged based.
- Enemies can drop bullets or health packs.
- If your health reaches 0, you lose the game.
- If you defeat all the waves, you win the game and can restart to play again.

Future Updates

- New Levels and more waves
- Fixing the raycasting for the Lazer Pointer
- Fixing the search and fetch modes
- New enemies
- Overarching story

Conclusion

Any questions?