Custom Project: Zombie Smart Survival

Name: Wilbert Kruskie

Student ID: 104323659

Date: 13/06/2025

1. Introduction

The **Zombie AI** was developed to implement advanced AI techniques such as **pathfinding**, **steering behaviours**, and **decision-making** to create intelligent and dynamic agents within the game world. This project aligns with **Intended Learning Outcomes (ILO) 1-5**, covering aspects of game **AI design, optimization,** and **interaction mechanics**.

To achieve realistic AI movement, the **A* pathfinding algorithm** was utilized, allowing zombies to dynamically calculate routes when faced with obstacles or chasing the player. Additionally, Steering behaviours such as **Cohesion, Alignment,** and **Separation** ensure fluid and lifelike motion while maintaining group coordination.

Beyond navigation, the AI incorporates a **State Machine** system, enabling zombies to transition between **patrol, chase,** and **attack** states based on environmental triggers. The integration of **goal-oriented behaviours** further enhances the AI's adaptability, allowing zombies to respond dynamically to player actions, visibility checks, and strategic positioning.

A significant focus was placed on creating a **cohesive** and **responsive** system, ensuring AI components interact seamlessly to deliver engaging and immersive gameplay. By refining collision detection, velocity mechanics, and AI behaviour synchronization, the Zombie AI successfully replicates naturalistic movement and intelligent decision-making.

2. AI Design & Implementation

Zombie AI Behaviour

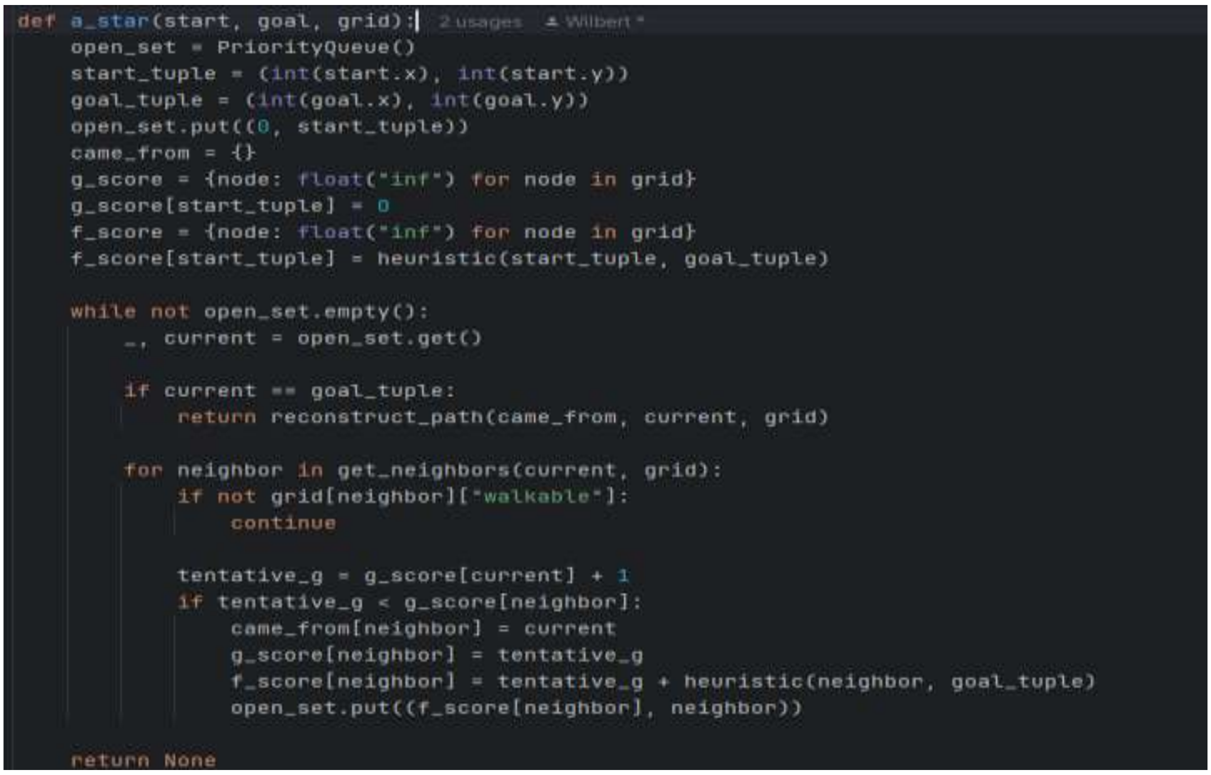
The **Zombie AI** operates with a **state-based system**, transitioning dynamically based on environmental conditions. The default state is **patrol**, where zombies follow predefined waypoints **until a player enters their vision cone**. Once detected, the AI triggers *A* pathfinding*, recalculating an optimal route to **chase** the player. Zombies only enter the **attack state** upon reaching a specified distance, simulating a **bite-based melee attack** without the need for additional weapons. A timer-based attack mechanism ensures **smooth animations and realistic combat pacing**.

Pathfinding & Navigation

To enable **intelligent movement**, the AI leverages *A* pathfinding*, implemented using **Python's PriorityQueue**. This system:

- Uses **tuple-based positioning** for efficient node representation.

- Accounts for **obstacles**, assigning movement costs (set to **1 per node**) to balance navigation complexity.
- Utilizes the **Manhattan heuristic**, ensuring zombies prioritize **the shortest path** to their target. Once the path is computed, the AI reconstructs the movement sequence, guiding zombies effectively toward the player.



```
def a_star(start, goal, grid):  2 usages  Wilbert *
    open_set = PriorityQueue()
    start_tuple = (int(start.x), int(start.y))
    goal_tuple = (int(goal.x), int(goal.y))
    open_set.put((0, start_tuple))
    came_from = {}
    g_score = {node: float("inf") for node in grid}
    g_score[start_tuple] = 0
    f_score = {node: float("inf") for node in grid}
    f_score[start_tuple] = heuristic(start_tuple, goal_tuple)

    while not open_set.empty():
        _, current = open_set.get()

        if current == goal_tuple:
            return reconstruct_path(came_from, current, grid)

        for neighbor in get_neighbors(current, grid):
            if not grid[neighbor]["walkable"]:
                continue

            tentative_g = g_score[current] + 1
            if tentative_g < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g
                f_score[neighbor] = tentative_g + heuristic(neighbor, goal_tuple)
                open_set.put((f_score[neighbor], neighbor))

    return None
```

Decision-Making & Vision Mechanics

Zombies rely on **a vision cone** to detect targets, considering obstacles as **visibility blockers**. If the player is behind an obstacle, **detection fails**, preventing unrealistic tracking. The AI dynamically reassesses the environment, adapting movement accordingly.

Player AI & Goal Planning

The **Player AI** follows a **goal-driven behaviour system**, incorporating state machines for strategic movement:

- **Safe Spot System:** The player can retreat to a **safe zone** for **recovery and reloading**, but only if they are in a **survival state**.
- **Objective-Based Navigation:** Once the player is ready and equipped with ammunition, the AI calculates a route to the **end goal**, transitioning into an **objective-seeking state**.

Performance Optimizations

To maintain efficiency:

- **Pathfinding calls** are minimized, preventing excessive calculations while ensuring fluid AI movement.

- **Collision detection** uses **predefined distance-based triggers**, activating interactions only when the player or zombie reaches **a defined goal threshold**.

Debugging & Iterations

Key Fixes

1. **Preventing Zombies from Walking Through Obstacles**
    a. Initially, zombies were able to pass through obstacles due to incorrect collision checks.
    b. To fix this, I implemented future position prediction, where the AI checks the upcoming coordinates for obstacles before proceeding.
    c. If an obstacle is detected, the zombie adjusts its heading by scanning for an open path and moves accordingly

2. Building a Reliable A* Pathfinding System
    a. Early versions of A* failed to reconstruct paths properly even when reaching the end goal.
    b. Debugging revealed that world grid checks required tuples, ensuring proper node tracking within the pathfinding algorithm.
    c. With this fix, zombies correctly calculate and follow paths, avoiding unnecessary wandering.

Behaviour Refinements

1. State Switching Optimization:
    a. Transitioning between patrol, chase, and attack states posed early challenges in balancing realism and performance.
    b. Through multiple iterations, I refined state switching logic to ensure smooth behaviour transitions.
2. Patrol Mechanism Improvements:
    a. Originally, zombies wandered aimlessly, lacking structured movement patterns.
    b. Implementing a waypoint-based patrol system, with randomized paths, created a more lifelike zombie behaviour.
    c. Waypoints were adjusted to maintain at least a 100-unit distance, ensuring strategic movement.

AI Coordination Issues

1. Integrating AI Components Exposed Hidden Bugs:
    a. During integration, zombies failed to navigate correctly due to coordinate mismatches.
    b. Debugging revealed that Pyglet handles floating-point coordinates, while my grid was based on integers, causing misalignment.
    c. The solution involved adjusting all positional calculations to use integers before passing arguments, ensuring precise movement tracking.

Bug Found

1. While integrating obstacle with image, it causes bug with the obstacle location where it doesn’t exactly location on the coordinate pointed but it starts placing from the bottom

left on the coordinate, which I tried setting with anchor and alter the position to make sure it matches the position. But the bug is that it still isn't fully functioning as player can pass through a bit of the image because the coordinates is offset.

2. Also for the avoid obstacle, found bug when it wants to go to a destination right behind the obstacle, as it turn heading for avoiding obstacles and it calls the state behaviour which causes it to spin on place.

Supporting Artefacts

Game Design Document

1. Overview

The **Zombie Survival Game** focuses on building a player-driven experience where survival depends on strategic decisions and smart AI interactions. The player must **complete tasks while staying alive**, while the **zombies aim to eliminate them**, navigating obstacles and executing realistic movement patterns.

2. AI Integration

The AI system **enhances gameplay dynamics**, providing intelligent behaviours for both:

- Zombie AI: Implements patrolling, chasing, attacking, and obstacle avoidance.
- Player AI: Uses goal tracking, safe-zone mechanics, and path planning to progress toward survival objectives.

3. Environment Interaction

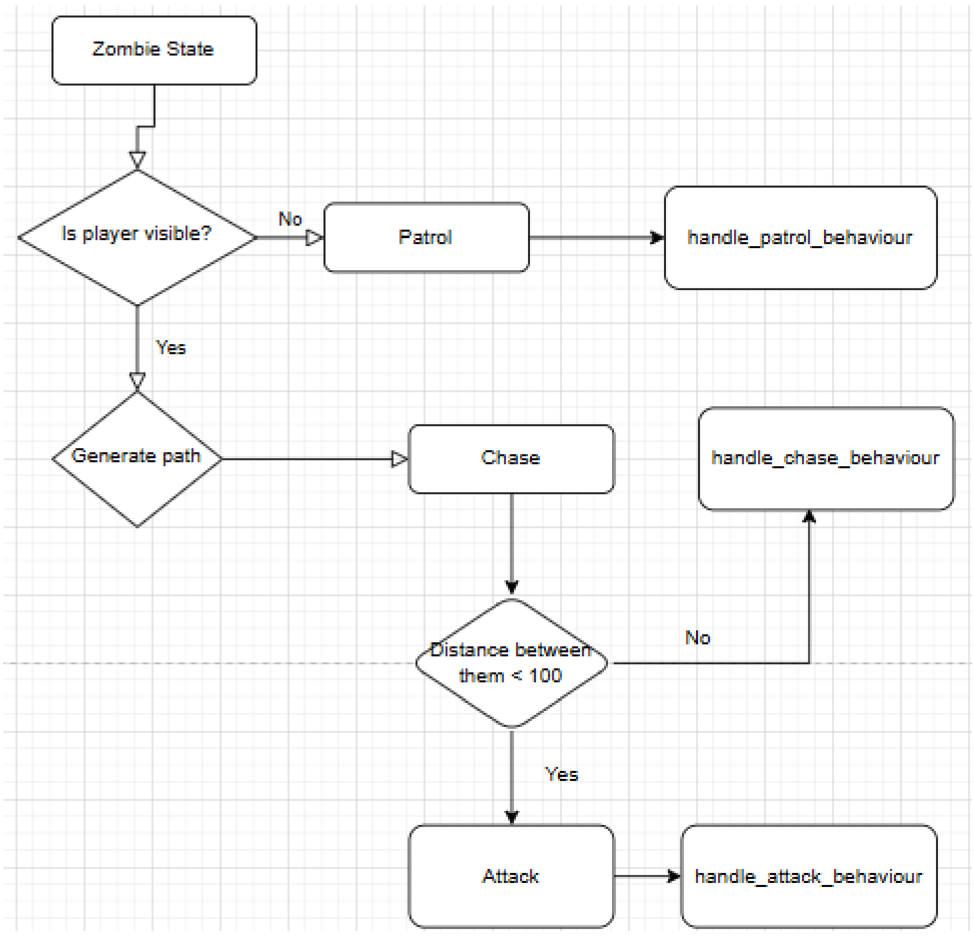
- Obstacles: Non-walkable areas force zombies to recalculate paths using A pathfinding*.
- Safe Spot: The player can heal and reload only in designated safe spots.
- End Goal: Completing key objectives triggers the winning conditions for survival.

4. Development Challenges

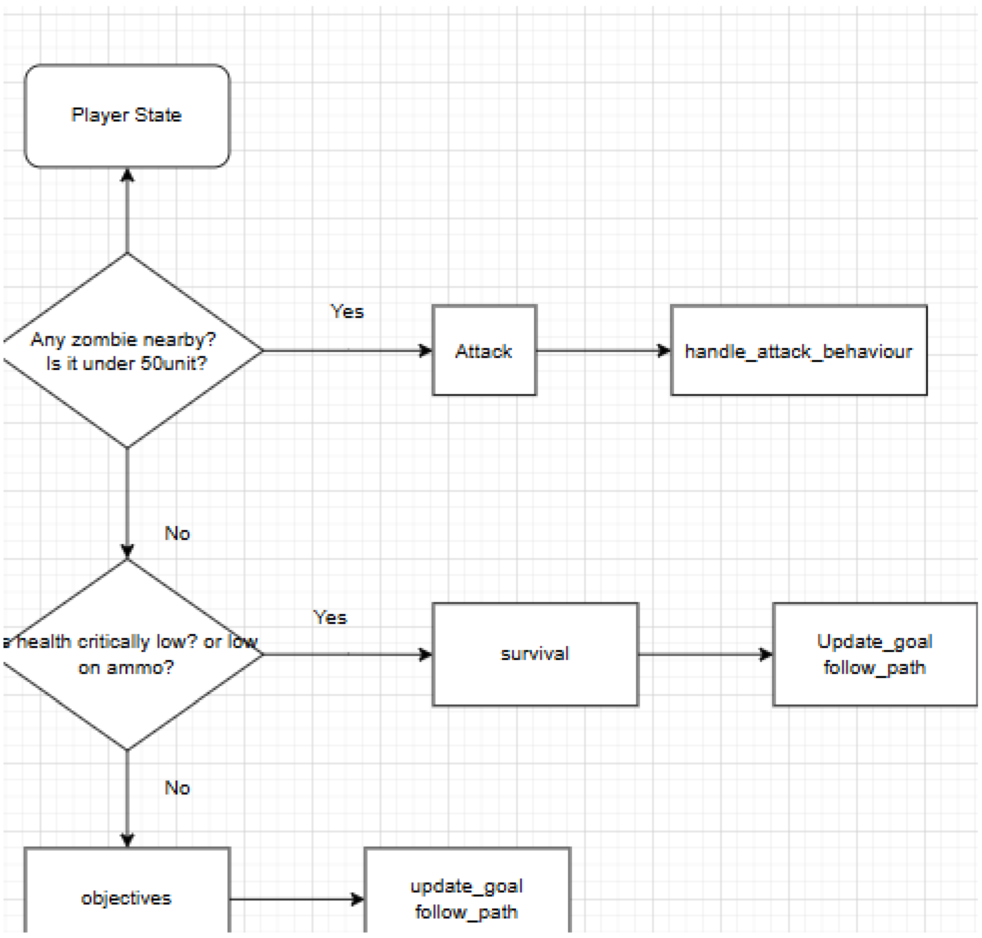
During integration, **AI behaviour became difficult to refine**, often conflicting with other rules, leading to **unexpected results**. Several iterations were required to **synchronize movement, improve decision-making, and enhance overall system stability**.

State Machine Diagram

Zombie State Mechanism



Player State Mechanism



References

- Course Materials: Reusing Task 12 class as a base.
- Online Resources
  - Vision cone:visioncones/main.py at main · endo4x/visioncones · GitHub
  - Pyglet Library: pyglet Documentation — pyglet v2.1.7