Project 3 Report

High Level Description of Public-Member functions of Classes:

**Actor Class**:

In the actor class, I had all the functions that I wanted for all my derived classes. If a function will be called from the student world class and it refers to an actor function. It will most probably be from this class that I make most of the setter and getters that I specialize in my derived classes. It has the main function that I call during the move function in student world, called doSomething.

**Penelope Class:**

In the penelope class, we have one main function, do something. In this function we check for if the user inputted any characters. Depending on the input, we either move to a direction, if penelope’s not blocked, or tell the student world functions to make landmines, flames or vaccines. We use if-else if statement throughout this function and call appropriate functions in student world that either perform a move, check the bounding box, or make vomit, flame or acquire landmine objects.

**Zombie Class:**

In the zombie class, we also have setters and getters but the main function we have in it is doSomething. This is obviously a base class to dumb and smart zombies. In this function we first check for the paralyzed counter and if it’s even, then we make the function not do anything. After that we check if the function is currently infected by flame, then we delete it[[1]](#footnote-0). Then we follow it up with having temporary variables that checks for the possible different vomit locations. If the possible vomit location is in within the penelope object then we infect the penelope object.

**Dumb Zombie Class:**

Since we have already finished a big chunk of implementation in the zombie base class, we just call the zombie base class’s version of doSomething, which is the common part of the implementation between smart and dumb zombies. The dumb zombie’s doSomething implementation then proceeds to calculate the movement distance plan if the plan is zero. It calculates a random number between 3 and 10 as the distance plan and also gets a random direction. If the plan is not zero, we move one step in the direction we want to go to, if the bounding allows us to, and decrease the plan by one.

**Smart Zombie Class:**

**Citizen Class:**

In citizen, we first check if the citizen is in paralysis and if the is experiencing infection. If infection we increment the infection until it goes away and in paralysis we return. If not then we calculate the zombie and penelope distances away from citizen. If penelope’s distance is less than 80 and is smaller than the distance of zombie. If the citizen is on the same row or column as citizen then we move 2 steps in the direction of penelope. If it is not in the same row or column, then we look at the two directions that would get us closer to penelope and pick one of those at random and move forward 2 steps in that direction. If the citizen is blocked at the direction we chose at random then we pick the other direction and move 2 steps. If we get blocked by both or have the distance to penelope be higher than 80, then we look at the four direction that we have and see if a move in any four would get us farther than the closest zombie, we would then move that direction. If blocked by all directions, the nothing happens so we return.

**Gas Can Goodie:**

For this goodie, we just check if we overlap with penelope using a helper function in student world and then tell student world to add points, set the object’s state to false and add 5 flames.

**Landmine Goodie:**

For this goodie, we just check if we overlap with penelope using a helper function in student world and then tell student world to add points, set the object’s state to false and add 1 landmine.

**Vaccine Goodie:**

For this goodie, we just check if we overlap with penelope using a helper function in student world and then tell student world to add points, set the object’s state to false and add 1 vaccine that we can obtain.

**Wall:**

Nothing really complex, return immediately in do something method.

**Flame:**

In flame’s do something, it must check if it reached two ticks since the beginning of its life. Then, we check if flame overlaps with penelope, zombies, citizens, goodies, and landmines. If that returns true, then we make a flame boolean true in each of the classes that we overlap with.

**Pit:**

In pit’s do something function, we just make anything that overlaps with it die using an overlap student world class helper function.

**Vomit:**

In vomit’s do something function, we just check if we overlap with penelope or citizen. If overlapped, we infect each object that overlapped with it.

**Exit:**

For exit, we check if we overlap with either penelope or citizen. If there are no more citizens, then penelope wins the game.

**StudentWorld:**

Move:

Ask the actor pointer of penelope to do something and return finished level if the player’s state is false. Then, we loop through the actor pointer list and if we find an actor who’s state is dead we delete it. If not, we ask each actor pointer to do something. We also show the string streams game text in move.

Distance To Zombie:

In this function we calculate the distance to the zombie function. We have a boolean that determines if the object is in a zombie, so that we how we loop through the actor pointer list. We then set two temporary variables to compare the current distance with distance in the loop before. Then take the minimum of the two variables. (Distance to penelope function is practically the same, but with the penelope object)

Determine Block:

Determine block is the function that we determine if an object is blocked by another object. We first check if the actor passed in by the function and the player passed would overlap by however many steps the function specifies it to be, then we return false. Then, we go and check the same thing for every other object in the actor list. If none overlap by the parameter steps, then we return true.

Determine Overlap, Determine Exit, Determine Overlap with Flames and Determine Overlap with Zombies and Players:

All functions use the euclidean distance formula to check if either player or anyone of the actor pointer lists overlap with the given object passed in the function. The difference now is if the object we have is supposed to return true for exit, then we return true in the exit function. However, if the object we have is supposed to return true for either vomit or pit(given by the boolean getOverlapD) then it’ll return true. Overlap with flames checks for overlap but uses the boolean getFlame. For overlap with players and flames, then we return true if the object passed in overlaps with penelope, citizens or zombies. Else, return false.

Is A Person In The Same Direction:

Goes through player actor pointer and actor pointers that are players and checks if the actor pointer passed in is in the general same direction, and is less than the width and height of the SPRITE.

Failed Functionality:

* Landmine doesn’t correctly function if overlaps with flames
* Citizens don’t correctly run away from zombies(implemented but just doesn’t work)
* Did not fully implement smart zombies

Assumed Functionality:

Generally, I thought the spec was pretty specific with everything

* That I can’t make flames after walls.
* If penelope has vaccines and is not infected it wasted that vaccine

How I tested my classes:

**Wall:**

Made sure that penelope couldn’t go through walls

**Flames:**

Went through and checked if flame overlaps correctly with player, zombies or penelope and acts right. Made sure also that I can’t make any flames over or after blocks, so it doesn’t make any flames after walls or the designated areas.

**Vomit:**

Went through and made sure that the infection happens if I(penelope) get a vomit object and the infection actually happens. Given by the counter that is appeared on the game text I could see that the object is actually infected. For citizens, I could see that they are actually infected visibly. WILL CHECK FOR IF THEY’RE DEAD

**Exit:**

Made sure that nothing overlaps with it and nothing causes it to die. Unless all citizens overlap with the exit and are deleted from the game, penelope will only then be able to cross.

**Pit:**

Made sure that Penelope loses a life whenever it crosses with a pit, and that when players cross my scores decrease.

**Citizen:**

Made sure that they are able to be infected by vomit and die to turn into zombies. Made sure that it doesn’t cross any walls or doesn’t go into any persons or zombies. Follows penelope correctly as it always follows it’s track of motion, and stays next to it if they block each other. It also can fall and die into a pit and can get burned by a flame.

**Penelope:**

Made sure that they are able to be blocked by bounding boxes and able to go through pits and exits and correctly performs in those situations. Doesn’t go through walls and doesn’t overlap with other movable objects. Correctly gets vomited on and infected on. Gets deleted and level finishes if overlaps with flame.

**Landmines:**

Made sure that they are able to be triggered if zombies, penelope, or citizens overlap with it. I had my program crash if my flames overlapped with my landmines.

**Dumb Zombies and Smart Zombies:**

I made sure that dumb and smart zombies vomit on penelope or humans when getting really close from either object. Made sure that they don’t overlap with walls, penelope or other zombies and get other objects vomited on correctly.

**Vaccines:**

Make sure that when penelope is infected, it can correctly heal and set the timer and boolean that determines if she’s infected all to zero.

1. The reason we don’t delete objects right away in student world is because different objects have different implementations/reactions to being overlapped with a flame. [↑](#footnote-ref-0)