Name\_Zachary Squires\_ Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

My feature for CS Survivor, a product by Teal Team 6, is a functioning fully automated demo that requires no player input until such time as a player may interact with the game and start their own playthrough. In addition to the demo, I will oversee the implementation of the AI enemies within CS Survivor.

Before the game even starts there will need to be an initial overlay of the system featuring our preview demo. Upon user interaction the demo ends, and the Players game begins. In essence I will be implementing a version of the game where AI plays it until the user starts.

Generation of AI and path mapping is the other task in my charge. This will require having our functional map environment as well as our characters. The enemies in CS survivor are bits of bad code such as segment faults, syntax errors, buffer overflow, that will need to ‘hunt’ down the player character, and attack/hit them reducing user hit points until they reach the end of their life, ending the game. This of course will also need to flow in the opposite direction in our demo. The AI played character will also need to be able to track enemy zombies to best avoid or attack them before they can get too close.

## Use case diagram with scenario \_\_14

### Use Case Diagrams

A diagram of a diagram

Description automatically generated

A diagram of a program

Description automatically generated

### Scenarios

**Scenario** **1** (1st Use Case Diagram):

**Name**: Demo Start

**Summary**: User approaches the system then starts/boots the game up. AI starts to play the demo, showing and simulating the gameplay. User interaction stops the demo and brings them to a menu where the user can either start their own player run game, visit other potential settings, or quit the game and exit.

**Actors:** User/Player

**Preconditions:** Someone approaches the system

**Basic Sequence:**

**Step 1:** User/Player Starts the game

**Step 2:** Demo Starts to Play

**Step 3:** Additional user input happens

**Step 4:** Demo gameplay ends

**Step 5:** Menu appears and displays possible user options including start player game, options, and exit game.

**Exceptions:**

**Step 1:** Upon game start AI immediately begins to play the demo

**Post conditions:** Demo is played automatically by AI until user interacts and starts their own game or exits.

**Priority:** 2\*

**ID:**  C01

**Scenario 2 (2nd** Use Case Diagram)

**Name:** Enemies

**Summary:** While the player plays the game, ‘bad code’ will spawn in to try and track down the player and attack them. Covers action interactions between player and the enemy.

**Actors:** Player,

**Preconditions:** Successful game start

**Basic Sequence:**

**Step 1:** Game starts

**Step 2:** Enemy AI begins spawning enemies

**Step 3:** Enemies begin to move around the environment

**Step 4:** Enemies detect player

**Step 5:** Enemies start to path towards and chase down the player

**Step 6**: Enemies attack and attempt to defeat the player.

**Post Conditions:** Enemies spawn in and begin to patrol for and hunt down the player character

**Priority:** **1\***

**ID:** C02

## Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_\_\_\_14

[Get the Level 0 from your team. Highlight the path to your feature]

Example:

### Data Flow Diagrams

A diagram of a level data

Description automatically generated

A diagram of a flowchart

Description automatically generated

### Process Descriptions

**Create Enemy:**

LoadEnemyTemplate()

{ If (GameStart)

load enemy template, basic || advanced;

return enemy template

}

**Assign Attributes:**

EnemyAttribute()

{ If (Template == basic)

Assign health = health.value, speed = speed.value, attack = attack.value;

}

Else {

If template == advanced;

Assign health = Ahealth.valuue, speed = Aspeed.value, attack = Aattack.value;

Return enemy

}

**Initialize AI:**

StartAI()

{ If enemy == basic

Set behavior patrol;

} else {

If enemy == advanced

Set behavior chase;

Return enemy

}

**Place Enemy in map:**

SpawnEnemy()

{ check valid spawn = map.getspawnpoint()

ChosenSpawnPoint = random(valid spawn)

Enemy.position = ChosenSpawnPoint

Return enemy

}

**Update Enemy Data:**

Update\_HostileAI(enemy)

{//Update system with current enemies state, and position

Update enemy in system(enemy)

}

## Acceptance Tests \_\_\_\_\_\_\_\_9

[Describe the inputs and outputs of the tests you will run. Ensure you cover all the boundary cases.]

Test for whether the correct enemy template is loaded based on the given type.

Test1: load enemy templates

Inputs:

Enemy\_type = basic

Enemy\_type = advanced

Enemy\_type = invalid

Expected outputs:

basic type: health = 100, speed = 2, attack = 10

advanced type: health = 150, speed = 3, attack = 20,

invalid type: NULL or ERROR

|  |  |  |  |
| --- | --- | --- | --- |
| # of Tests | Return Type Success | Invalid Returns | Successful Return Rate: |
| 10 | 9 | 1 | 90% |
| 50 | 48 | 2 | 96% |
| 100 | 96 | 4 | 96% |
| 150 | 148 | 2 | 98% |

## Timeline \_\_\_\_\_\_\_\_\_/10

[Figure out the tasks required to complete your feature]

Example:

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (hours) | Predecessor Task(s) |
| 1. Enemy Templates | 6 | - |
| 2. Enemy Design | 6 | 1 |
| 3. Develop Enemy AI | 15 | 2 |
| 4. Spawning System | 6 | 2 |
| 5. Enemy Actions | 10 | 4 |
| 6. Enemy State Manager | 6 | 3 |
| 7. AI behavior testing | 5 | 5,7 |
| 8. Debug and Optimize AI | 5 | 7 |

### Pert diagram

A diagram of a diagram

Description automatically generated

### Gantt timeline

A red blue and black line

Description automatically generated