Name: **Andrew Bonilla** Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

My feature in Hot Dog Jones the Video Game is I will be implementing a state machine that handles the sprites and animations of the main character and the enemies in the game.

**Movement:** We will be working with a top-down environment in our game. This means when the player moves left, right, up, or down, I will be responsible for dealing with the animations changing to a different state depending on the user’s input. When it comes to enemy AI they will also be moving around. I will be responsible for the animations and sprite states as well.

**Attacking/Taking Damage:** Along with regular movement from enemies and the player, I will also be dealing with the animations and different states of our player when they are attacking or take damage from an enemy. This also means I will be doing the same thing for an enemy when they attack or are taking damage as well.

**Powerups:** In Hot Dog Jones the Video Game there will be different powerups that the player can collect. When the player gets a power up, there will be a state and animation for when the power up is picked up and then there will be a different attack animation.   
  
**Death/Game Over:** On top of movement and attacking/taking damage. I will also be looking at and working with different states of the animation and sprites when there is a death of the player or enemies. When the player runs out of health or the enemy is defeated, there will be different animations for these as well.

## Use case diagram with scenario \_\_14

Ensure you have at least one exception case, and that the <<extend>> matches up with the Exceptions in your scenario, and the Exception step matches your Basic Sequence step.

Also include an <<include>> that is a suitable candidate for dynamic binding]

Example:

### Use Case Diagrams

A diagram of a company

Description automatically generated

### A diagram of a diagram Description automatically generated

### Scenarios

**[Scenario 1]**

**Name:** Interacts with powerup

**Summary:** The player moves over an interactable power up

**Actors:** Player

**Preconditions:** Player is moving over powerup

**Basic Sequence:**

**Step 1:** Player inputs movement command

**Step 2:** Player walks over powerup

**Step 3:** Animation plays

**Step 4:** Player has new attack

**Exceptions:**

**No exceptions**

**Postconditions:** Player now has new attack animation

**Priority:** 2

**ID:** C06.1

**[Scenario 2]**

**Name:** Player Dies

**Summary:** When player takes to much damage they die

**Actors:** Player

**Preconditions:** Player takes fatal damage

**Basic Sequence:**

**Step 1:** Enemy attacks player

**Step 2:** Player takes damage

**Step 3:** Damage animation plays

**Exceptions:**

**Step 2:** Player takes fatal damage

**Step 3:** Death animation plays

**Postconditions:** Player is dead

**Priority:** 1

**ID:** C06.1

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

### Data Flow Diagrams

A diagram of a process

Description automatically generatedMy process, is process number 5, highlighted in Red

### A diagram of a computer Description automatically generated with medium confidence

### Process Descriptions

If player interacts with Power up:

function playerInteractWithPowerUp(player, powerUp):

if player collides with powerUp:

playPowerUpAnimation(powerUp)

player.enableNewAttackAnimation()

function playPowerUpAnimation(powerUp):

// Play animation for power-up

// This could involve visual effects or sound effects

// Example: display a sparkle effect or play a power-up sound

function Player.enableNewAttackAnimation():

// Enable the new attack animation for the player

// This could involve changing the player's sprite or animation state

// Example: switch the player's attack animation to a new one

// Example usage:

player = getPlayer() // Get reference to player object

powerUp = getPowerUp() // Get reference to power-up object

// Check for player interaction with power-up

playerInteractWithPowerUp(player, powerUp)

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

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

**Player Animation:**

This acceptance test will check the main contents related to the main player when it comes to health, powerup collection, attacking status, as well as taking damage or dying. First this test will require user input to ensure the animations are playing for all movement and ensure that the state machine is capturing different states of the player and animation. This will also play a role in the attacking animation as well. There will also be a test with the powerups in the game and the player will walk over them to ensure the state changes to the powerup pickup animation. Then the players attack animations should be different as well. Lastly, we will test damage and death animation states with having an enemy attack the player to ensure animations and states are being updated as player is damaged, as well as when the player’s health reaches 0.

**Enemy Animation:**

This acceptance test will work similarly to the player except we will have an enemy npc state machine that will check enemy states and animations as well. We will test the enemies attacking the player, as well as the enemy taking damage and dying. This could be simply spawning an enemy in front of the player and testing the states and animations when the enemy is attacking or being attacked. And we want to ensure the states are being updated accordingly.

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

[Figure out the tasks required to complete your feature]

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (PWks) | Predecessor Task(s) |
| 1. Player State Machine | 6 | - |
| 2. Enemy State Machine | 6 | 1 |
| 3. Sprites | 4 | 1 |
| 4. Player Animations | 5 | 3 |
| 5. Enemy Animations | 5 | 2,3 |
| 6. Programming | 5 | 4,5 |
| 7. Testing | 3 | 6 |
| 8. Integration | 4 | 6,7 |

### Pert diagram

A diagram of a number

Description automatically generated

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  | 2,3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 4,5 |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6,7 |  |  |  |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |

### Gantt timeline

|  |  |
| --- | --- |
| Wrk Hrs | Slack |
|  |  |