Name: Ben Bonus Mark: /50

## Introduction \_\_/3

My feature for our game, The Shipwreck Protocol, is to handle the user’s input and translate it into movement for the player character.

When a player presses a key on a keyboard, presses a button or moves a joystick on a controller, or taps a specific area on a touchscreen, the game needs to be able to read it and perform the proper actions.

My job is to ensure that these actions performed by the user are properly read, and the player character responds accordingly by performing various movements.

## Use Case Diagram (with Scenario) \_\_/14

#### Use Case Diagrams

A diagram of a diagram

Description automatically generated

#### Scenarios

## **Name:** Player Jumps

## **Summary:** While the game is playing, the player presses the jump button.

## **Actors:** User, StateManager

## **Preconditions:** The player has been created and the game is running.

## **Basic sequence:**

## **Step 1:** The user presses the jump button.

## **Step 2:** The StateManager checks if the player is on the ground. If the player is not, check to see if the player has remaining jumps available.

## **Step 3:** The player character gains a burst of positive upwards velocity.

## **Exceptions:**

## **Step 1:** Wrong button is pressed: do nothing.

## **Step 2:** Player is in air and does not have remaining jumps: do nothing;

## **Post conditions:** Calculated value is displayed.

## **Priority:** 2

## **ID:** PJ1

## Data Flow Diagrams \_\_/14Data Flow Diagrams

A diagram of a diagram

Description automatically generated

#### Process Descriptions

**Subscribe to Inputs:**

Create variables for variables relevant to each input (Input comes from Input Asset)

Create functions for each input variable in which the input variable reads the input parameters

Subscribe the performed and started actions for each input type to their respective function

**Calculate Movements:**

Receive input from Input Manager

If received input is horizontal movement values: try to move player

If received input is attack button call: try to attack

If received input is dash button call and can dash: try to dash

If received input is jump button call:

If player is grounded: try to jump

If player is in air and has remaining jumps: try to jump

## Acceptance Tests \_\_/9

**When player moves:**

* Test to see if the player can move. If the path is blocked, the player should not move.

**When the player jumps:**

* Test to see if the player can jump. If the player is grounded, the player can jump. If the player is in the air, check to see if the player has jumps remaining. If the player does, jump and decrease the jumps remaining. If the player does not have jumps remaining, do not jump.

**When the player dashes:**

* Test to see if the player can dash. If the player does not have any required items to dash, or if dash is on cooldown, the player should not dash.

## Timeline \_\_/10

#### Work Items

|  |  |  |
| --- | --- | --- |
| Task | Duration (PWks) | Predecessor Task(s) |
| 1. Devise good control layout | 1 | - |
| 1. Code | 10 | 1 |
| 1. Gather Feedback | 2 | 2 |
| 1. Convert to mobile format | 8 | 3 |
| 1. Final feedback | 2 | 4 |
| 1. Documentation | 3 | 2 |
| 1. Deployment | 4 | 5, 6 |

#### Pert Diagram

#### A diagram of a diagram Description automatically generated

#### Gantt Timeline

A close-up of a chart

Description automatically generated