Name\_\_Devon Bryce\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

The feature I’m working on is player movement. I’m in charge of making sure the player can move their character forward, backward, left and right using 4 inputs. The player must also move only if the player is pressing down a key and not if the key is up.

## Use case diagram with scenario \_\_14

### Use Case Diagrams

A diagram of moving and moving

Description automatically generated

### Scenarios

**Name:** Move Left

**Summary:** The player gives an input that moves their character slightly left.

**Actors:** Player

**Preconditions:** The player character has been initialized.

**Basic sequence:**

**Step 1:** Accept input “a”.

**Step 2:** Move the player character slightly left.

**Step 3:** Once there is no more input, stop moving the character left.

**Exceptions:**

**Step 1:** The player character hits an obstacle they cannot pass through: the character does not move left.

**Post conditions:** The player character is somewhere to the left of their previous position.

**Priority:** 2\*

**ID:** M01

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

**Name:** Move Right

**Summary:** The player gives an input that moves their character slightly right.

**Actors:** Player

**Preconditions:** The player character has been initialized.

**Basic sequence:**

**Step 1:** Accept input “d”.

**Step 2:** Move the player character slightly right.

**Step 3:** Once there is no more input, stop moving the character right.

**Exceptions:**

**Step 1:** The player character hits an obstacle they cannot pass through: the character does not move right.

**Post conditions:** The player character is somewhere to the right of their previous position.

**Priority:** 2\*

**ID:** M01

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

**Name:** Move Forward

**Summary:** The player gives an input that moves their character slightly forward.

**Actors:** Player

**Preconditions:** The player character has been initialized.

**Basic sequence:**

**Step 1:** Accept input “w”.

**Step 2:** Move the player character slightly forward.

**Step 3:** Once there is no more input, stop moving the character forward.

**Exceptions:**

**Step 1:** The player character hits an obstacle they cannot pass through: the character does not move forward.

**Post conditions:** The player character is somewhere in front of their previous position.

**Priority:** 2\*

**ID:** M01

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

**Name:** Move Backward

**Summary:** The player gives an input that moves their character slightly backward.

**Actors:** Player

**Preconditions:** The player character has been initialized.

**Basic sequence:**

**Step 1:** Accept input “s”.

**Step 2:** Move the player character slightly backward.

**Step 3:** Once there is no more input, stop moving the character backward.

**Exceptions:**

**Step 1:** The player character hits an obstacle they cannot pass through: the character does not move backward.

**Post conditions:** The player character is somewhere behind their previous position.

**Priority:** 2\*

**ID:** M01

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

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

### Data Flow Diagrams

A diagram of a diagram

Description automatically generated



A computer screen shot of a computer

Description automatically generated with medium confidence

### Process Descriptions

Create player(){

If (the level data is accessed && the level is loaded properly){

Spawn the player character;

Load player data;

}

}

Access movement system(){

If (the player character has been properly loaded in && the player gives valid input){

Execute the corresponding movement;

}

}

Check obstacle collision(){

Access necessary level data;

If (the player hits an obstacle){

Prevent movement in the direction of the obstacle;

}

}

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

**Moves test**

Run Moves 100 times using inputs “w”, “a”, “s”, and “d”.

**Moves cases**

|  |  |  |
| --- | --- | --- |
| Output | Input | Notes |
| Move forward | w | The player moves forward unless there is an obstacle in front of them. |
| Move  left | a | The player moves left unless there is an obstacle to the left of them. |
| Move right | s | The player moves backward unless there is an obstacle behind them. |
| Move backward | d | The player moves right unless there is an obstacle to the right of them. |

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

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (Hrs) | Predecessor Task(s) |
| 1. Requirements Collection | 1 | - |
| 2. Programming | 5 | 1 |
| 3. Testing | 2 | 2 |
| 4. Installation | 1 | 2, 3 |

### Pert diagram

A diagram of a number

Description automatically generated with medium confidence

### Gantt timeline

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |