Purpose

UAT (user acceptance testing) is important for this mobile game as application testing with users ensures that the mechanics and features that have been implemented are functional and meets the needs of the users.

Scope

* Player movement
* Platform spawning
* Platform jumping
* Score system

Objective

Player movement should function as the user expects it to. The player game object should move fluidly with the movements of the user at their command

## Test

### User 1

**Test Owner** -

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| User | Screen size | Operating System | Phone Model |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
|  |  |  |  |

Player Movement

**Test Case** - Verify that the user can interact with the player game object and that it functions as they expect it to.

**Expected Outcome** - The user should touch and hold the player game object with one finger on the touchscreen and drag it left and right. The game object should follow the movements of the user along the X axis exclusively.

1. When in game scene, touch the chicken sprite
2. Drag it back and forth

**Result** –

**Test Case** – Verify that the game has high performance and does not lag.

**Expected Outcome** – The game should run smoothly across all devices and should have any considerable amount of lag.

1. Touch the player object
2. Drag it back and forth
3. Jump on platforms

**Result** –

**Test Case** – Verify that the user can’t escape the boundary of the play area/screen.

**Expected Outcome** – Due to the boundary colliders in the scene and the player object collider, the two should collider and the player object should stay within the screen edge.

1. Enter game scene
2. Play normally
3. Erratically drag the player object off screen

**Result** –

Platform Spawning

**Test Case** - Verify that the platforms spawn continuously as the user is playing.

**Expected Outcome** - Every time the user jumps on a platform, a new platform should spawn outside of the screen view to create a seamless illusion that there are endless platforms.

1. Play normally
2. Advance upwards
3. Note the platforms spawning

**Result** –

**Test Case** – Verify that the user can reach each platform within distance of each other.

**Expected Outcome** – Every platform should spawn within relative distance of each other and the user should find no issue with moving between platforms.

**Result** –

**Test Case** – Verify that the platforms get destroyed after being landed on.

**Expected Outcome** – There should be a one second delay after the cloud has been landed on before it destroys itself.

1. Play the game
2. Play up to a certain height
3. Land on a platform and stay still
4. Platform should destroy itself

**Result** –

Platform Jumping

**Test Case** – Verify that the user can jump to and from every platform.

**Expected Outcome** – The player object should be continuously bouncing so that the user can jump and reach every platform in the scene.

1. Advance through the game
2. Ensure player is constantly bouncing on each platform

**Result** –

**Pass/Fail** –

**Test Case** – Verify that the user can phase through the platforms from underneath to then sit on top.

**Expected Outcome** – The platforms should allow the player object to phase through it from underneath to land on it with no collision when moving through.

1. Play normally
2. Aim for the next platform
3. Phase through the platform

**Result** –

**Test Case** – Verify that the user can enter the platform on an angle for side movements.

**Expected Outcome** – The platforms should allow the player object to phase through it from underneath to land on it with no collision when moving through.

1. Play normally
2. Identify the next platform is on the opposite side of the screen
3. Jump to it from the side (not directly underneath)

**Result** –

Score System

**Test Case** – Verify that the user’s height gets updated in game.

**Expected Outcome** – Upon entering the main scene of the game, the height UI should increment by one with every platform the user lands on.

**Result** –

**Pass/Fail** –

**Test Case** – Verify that the high score is saved and displayed on the title screen.

**Expected Outcome** – When a user plays, then exits back to the title screen (or exits and enters the game), the player preferences should save the high score and get displayed on the title screen.

**Result** –

**Test Case** – Verify that the high score UI is visible on the user’s phone

**Expected Outcome** – When the user plays, the UI in the top right corner should not be hidden off screen,

**Result** –

### User 2

**Test Owner** -

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| User | Screen size | Operating System | Phone Model |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
|  |  |  |  |

Player Movement

**Test Case** - Verify that the user can interact with the player game object and that it functions as they expect it to.

**Expected Outcome** - The user should touch and hold the player game object with one finger on the touchscreen and drag it left and right. The game object should follow the movements of the user along the X axis exclusively.

1. When in game scene, touch the chicken sprite
2. Drag it back and forth

**Result** –

**Test Case** – Verify that the game has high performance and does not lag.

**Expected Outcome** – The game should run smoothly across all devices and should have any considerable amount of lag.

1. Touch the player object
2. Drag it back and forth
3. Jump on platforms

**Result** –

**Test Case** – Verify that the user can’t escape the boundary of the play area/screen.

**Expected Outcome** – Due to the boundary colliders in the scene and the player object collider, the two should collider and the player object should stay within the screen edge.

1. Enter game scene
2. Play normally
3. Erratically drag the player object off screen

**Result** –

Platform Spawning

**Test Case** - Verify that the platforms spawn continuously as the user is playing.

**Expected Outcome** - Every time the user jumps on a platform, a new platform should spawn outside of the screen view to create a seamless illusion that there are endless platforms.

1. Play normally
2. Advance upwards
3. Note the platforms spawning

**Result** –

**Test Case** – Verify that the user can reach each platform within distance of each other.

**Expected Outcome** – Every platform should spawn within relative distance of each other and the user should find no issue with moving between platforms.

**Result** –

**Test Case** – Verify that the platforms get destroyed after being landed on.

**Expected Outcome** – There should be a one second delay after the cloud has been landed on before it destroys itself.

1. Play the game
2. Play up to a certain height
3. Land on a platform and stay still
4. Platform should destroy itself

**Result** –

Platform Jumping

**Test Case** – Verify that the user can jump to and from every platform.

**Expected Outcome** – The player object should be continuously bouncing so that the user can jump and reach every platform in the scene.

1. Advance through the game
2. Ensure player is constantly bouncing on each platform

**Result** –

**Pass/Fail** –

**Test Case** – Verify that the user can phase through the platforms from underneath to then sit on top.

**Expected Outcome** – The platforms should allow the player object to phase through it from underneath to land on it with no collision when moving through.

1. Play normally
2. Aim for the next platform
3. Phase through the platform

**Result** –

**Test Case** – Verify that the user can enter the platform on an angle for side movements.

**Expected Outcome** – The platforms should allow the player object to phase through it from underneath to land on it with no collision when moving through.

1. Play normally
2. Identify the next platform is on the opposite side of the screen
3. Jump to it from the side (not directly underneath)

**Result** –

Score System

**Test Case** – Verify that the user’s height gets updated in game.

**Expected Outcome** – Upon entering the main scene of the game, the height UI should increment by one with every platform the user lands on.

**Result** –

**Pass/Fail** –

**Test Case** – Verify that the high score is saved and displayed on the title screen.

**Expected Outcome** – When a user plays, then exits back to the title screen (or exits and enters the game), the player preferences should save the high score and get displayed on the title screen.

**Result** –

**Test Case** – Verify that the high score UI is visible on the user’s phone

**Expected Outcome** – When the user plays, the UI in the top right corner should not be hidden off screen,

**Result** –

### User 3

**Test Owner** -

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| User | Screen size | Operating System | Phone Model |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
|  |  |  |  |

Player Movement

**Test Case** - Verify that the user can interact with the player game object and that it functions as they expect it to.

**Expected Outcome** - The user should touch and hold the player game object with one finger on the touchscreen and drag it left and right. The game object should follow the movements of the user along the X axis exclusively.

1. When in game scene, touch the chicken sprite
2. Drag it back and forth

**Result** –

**Test Case** – Verify that the game has high performance and does not lag.

**Expected Outcome** – The game should run smoothly across all devices and should have any considerable amount of lag.

1. Touch the player object
2. Drag it back and forth
3. Jump on platforms

**Result** –

**Test Case** – Verify that the user can’t escape the boundary of the play area/screen.

**Expected Outcome** – Due to the boundary colliders in the scene and the player object collider, the two should collider and the player object should stay within the screen edge.

1. Enter game scene
2. Play normally
3. Erratically drag the player object off screen

**Result** –

Platform Spawning

**Test Case** - Verify that the platforms spawn continuously as the user is playing.

**Expected Outcome** - Every time the user jumps on a platform, a new platform should spawn outside of the screen view to create a seamless illusion that there are endless platforms.

1. Play normally
2. Advance upwards
3. Note the platforms spawning

**Result** –

**Test Case** – Verify that the user can reach each platform within distance of each other.

**Expected Outcome** – Every platform should spawn within relative distance of each other and the user should find no issue with moving between platforms.

**Result** –

**Test Case** – Verify that the platforms get destroyed after being landed on.

**Expected Outcome** – There should be a one second delay after the cloud has been landed on before it destroys itself.

1. Play the game
2. Play up to a certain height
3. Land on a platform and stay still
4. Platform should destroy itself

**Result** –

Platform Jumping

**Test Case** – Verify that the user can jump to and from every platform.

**Expected Outcome** – The player object should be continuously bouncing so that the user can jump and reach every platform in the scene.

1. Advance through the game
2. Ensure player is constantly bouncing on each platform

**Result** –

**Pass/Fail** –

**Test Case** – Verify that the user can phase through the platforms from underneath to then sit on top.

**Expected Outcome** – The platforms should allow the player object to phase through it from underneath to land on it with no collision when moving through.

1. Play normally
2. Aim for the next platform
3. Phase through the platform

**Result** –

**Test Case** – Verify that the user can enter the platform on an angle for side movements.

**Expected Outcome** – The platforms should allow the player object to phase through it from underneath to land on it with no collision when moving through.

1. Play normally
2. Identify the next platform is on the opposite side of the screen
3. Jump to it from the side (not directly underneath)

**Result** –

Score System

**Test Case** – Verify that the user’s height gets updated in game.

**Expected Outcome** – Upon entering the main scene of the game, the height UI should increment by one with every platform the user lands on.

**Result** –

**Pass/Fail** –

**Test Case** – Verify that the high score is saved and displayed on the title screen.

**Expected Outcome** – When a user plays, then exits back to the title screen (or exits and enters the game), the player preferences should save the high score and get displayed on the title screen.

**Result** –

**Test Case** – Verify that the high score UI is visible on the user’s phone

**Expected Outcome** – When the user plays, the UI in the top right corner should not be hidden off screen,

**Result** -

### User 4

**Test Owner** -

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| User | Screen size | Operating System | Phone Model |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
|  |  |  |  |

Player Movement

**Test Case** - Verify that the user can interact with the player game object and that it functions as they expect it to.

**Expected Outcome** - The user should touch and hold the player game object with one finger on the touchscreen and drag it left and right. The game object should follow the movements of the user along the X axis exclusively.

1. When in game scene, touch the chicken sprite
2. Drag it back and forth

**Result** –

**Test Case** – Verify that the game has high performance and does not lag.

**Expected Outcome** – The game should run smoothly across all devices and should have any considerable amount of lag.

1. Touch the player object
2. Drag it back and forth
3. Jump on platforms

**Result** –

**Test Case** – Verify that the user can’t escape the boundary of the play area/screen.

**Expected Outcome** – Due to the boundary colliders in the scene and the player object collider, the two should collider and the player object should stay within the screen edge.

1. Enter game scene
2. Play normally
3. Erratically drag the player object off screen

**Result** –

Platform Spawning

**Test Case** - Verify that the platforms spawn continuously as the user is playing.

**Expected Outcome** - Every time the user jumps on a platform, a new platform should spawn outside of the screen view to create a seamless illusion that there are endless platforms.

1. Play normally
2. Advance upwards
3. Note the platforms spawning

**Result** –

**Test Case** – Verify that the user can reach each platform within distance of each other.

**Expected Outcome** – Every platform should spawn within relative distance of each other and the user should find no issue with moving between platforms.

**Result** –

**Test Case** – Verify that the platforms get destroyed after being landed on.

**Expected Outcome** – There should be a one second delay after the cloud has been landed on before it destroys itself.

1. Play the game
2. Play up to a certain height
3. Land on a platform and stay still
4. Platform should destroy itself

**Result** –

Platform Jumping

**Test Case** – Verify that the user can jump to and from every platform.

**Expected Outcome** – The player object should be continuously bouncing so that the user can jump and reach every platform in the scene.

1. Advance through the game
2. Ensure player is constantly bouncing on each platform

**Result** –

**Pass/Fail** –

**Test Case** – Verify that the user can phase through the platforms from underneath to then sit on top.

**Expected Outcome** – The platforms should allow the player object to phase through it from underneath to land on it with no collision when moving through.

1. Play normally
2. Aim for the next platform
3. Phase through the platform

**Result** –

**Test Case** – Verify that the user can enter the platform on an angle for side movements.

**Expected Outcome** – The platforms should allow the player object to phase through it from underneath to land on it with no collision when moving through.

1. Play normally
2. Identify the next platform is on the opposite side of the screen
3. Jump to it from the side (not directly underneath)

**Result** –

Score System

**Test Case** – Verify that the user’s height gets updated in game.

**Expected Outcome** – Upon entering the main scene of the game, the height UI should increment by one with every platform the user lands on.

**Result** –

**Pass/Fail** –

**Test Case** – Verify that the high score is saved and displayed on the title screen.

**Expected Outcome** – When a user plays, then exits back to the title screen (or exits and enters the game), the player preferences should save the high score and get displayed on the title screen.

**Result** –

**Test Case** – Verify that the high score UI is visible on the user’s phone

**Expected Outcome** – When the user plays, the UI in the top right corner should not be hidden off screen,

**Result** –

## Evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| User | Functionality | Usability | Overall Experience |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Reporting

Summary (testing process, coverage, overall findings)

Open issues and bugs

Recommendations for improvements based on test results and user feedback