## Introduction

Purpose

User acceptance testing (UAT) 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. At the end of the development process, user testing is performed with real users to check the validity of functionality with the mechanics of the application.

Scope

The type of testing conducted is beta testing as there is a limited number of users testing the application. The mechanics/main features of the UAT conducted are as follows:

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

These mechanics all use functionality testing to ensure that the features work as intended. In my user testing, my test case expected outcomes outline the desired functionalities for the specific mechanic.

User experience (UX) testing is also conducted. When I am testing my users, I use UX techniques like observation to note the behaviours of the users and how they interact with different elements and mechanics of the application.

Performance testing is also used in a test case of the player movement mechanic to ensure that the user’s input and the applications output is responsive.

Objective

Player movement should be user-friendly as it should respond to the user’s input in a responsive manner and react the way the user expects it to. Through testing, player movement should receive the user’s input and be responsive with high performance.

## Test

### User 1

**Test Owner** -

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| User | Screen size | Operating System | Phone Model |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
| Alex |  | Android | Samgsumg S9+ |

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** – User touches area around the player game object. Expects to be able to touch anywhere on the screen to interact with the player.

**Pass/Fail** - Pass

**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** – No lag, camera shake when touching player.

**Test Case** – Verify that the user cannot 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** – Successful at not being able to escape the screen but camera jitters when colliding with boundary.

**Pass/Fail** - Pass

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** – Spawns out of view, which is successful.

**Pass/Fail** - Pass

**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** – Spaced evenly, enough time to jump across the screen.

**Pass/Fail** - Pass

**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** – Platforms get destroyed after bouncing on them.

**Pass/Fail** - Pass

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** – Most of the time the player continuously bounces on the platform, except when entering a platform from the side or at an angle.

**Pass/Fail** - Pass

**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** – Successful at phasing through the platforms then landing on the solid top.

**Pass/Fail** - Pass

**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** – Successful at phasing through it on the side.

**Pass/Fail** - Pass

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** – Gets updated on every platform.

**Pass/Fail** – Pass

**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** – Saves high score on the title screen.

**Pass/Fail** - Pass

**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** – Good positioning in the corner, doesn’t go off screen, White against the clouds, not distracting or hard to look at

**Pass/Fail** – Pass

Overall Experience – minimal bugs, non-intrusive bugs, good, visual design very good,

Bugs – jumping stopping, camera jittering at boundary, high score being added to main score

### User 2

**Test Owner** -

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| User | Screen size | Operating System | Phone Model |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
| Lorna |  | Android | Samsung A15 |

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** – Presses the screen around the player object. Camera shake when first touch of player object. Should have a delay. Sensitivity is fine. Need to increase collider area.

**Pass/Fail** – Pass

**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** – No lag, scene loading delay for graphical loading.

**Pass/Fail** – Pass

**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** – Successful unable to get off the sides.

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** – Successful, clouds spawn off screen.

**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** – Too far apart when spawning horizontally. Especially when clouds are off screen.

**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** – Most of the time unless the player object stops jumping on the platfrom

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** – Most of the time the player is bouncing continuously unless the jumping animation is in line with the top of the cloud which means there is less to bounce from.

**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** – Phasing is a success.

**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** – Passes through on the side.

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** – Successful, ui updated constantly

**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** – Yes high score saves on the title screen

**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** – Good positioning, colour can merge when in front of a cloud, size is fine, not hard not read.

Overall Experience – Engaging

Expect to in front of the clouds. Clouds are off screen

Bugs – High score added to normal score in game, stops bouncing when at equal level to top of platform, scene load glitch, camera shakes when hitting the edge of screen.

### 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** – Touches around the player object

**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** – No lag, except for the scene load

**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** – Successfully can’t escape the boundary.

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** – Successful with platforms spawning out of view

**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** – Success no issue with vertical or horizontal.

**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** – Platforms destroy, takes a while

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** – Mostly successful,

**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** – Successful

**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** – Bound from the sides moves the camera

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** – Successfully gets updated.

**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** – Successfully saved

**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** – Camera issue

Overall – replayable, and good.

Bugs – scene loading, score, jumping

### User 4

**Test Owner** -

**Environment**

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

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** – Touch around the player object before realising to touch the chicken.

**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** – No lag

**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** – Successful, aside from object jittering

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** – Successful spawning off screen

**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** – Random, reachable,

**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** – Successful

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** – Most of the time player continuously jumps

**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** – Successful

**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** – Successful

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** – Successfully increase

**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** – Yes it saves and updates

**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** – No issue with concentration or usability

Overall – Enjoyable,

Bugs – No chicken jump, high score on main,

## 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