## 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.

UAT is important for recording the feedback given by the users to improve the game. The testers are a sample of a wider audience, so their feedback, and observation of how they respond and interact with the game is a good indicator for how the game can improve and develop further for more people to enjoy.

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.

Platform spawning should be user-friendly for different mobile screen sizes, as it needs to spawn outside of the user’s view for an immersive experience to seem like the platforms already exist in the world and are entering the player’s view when they move up them.

Platform jumping should move and flow with the user’s input. For the jumping to feel user-friendly, it must feel like they are in control of the chicken to guide it. Unlike player movement for input and movement testing, jumping is how the chicken acts with the platforms. The chicken must move through the bottom of the platform then land on top and not feel like the platforms themselves are obstacles.

Score system should be displayed in a user-friendly way that makes it readable on all mobile devices. Size should be big enough to read and should be readable against the background. The data should also be reliable and consistent so the user can trust that every platform increases their score, and that it gets saved on every session of gameplay.

## Test

### User 1

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Owner** | **Screen size** | **Operating System** | **Phone Model** |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
| Alex | 18.5:0 | 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.

**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.

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

**Pass/Fail** – Pass.

**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.

**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.

**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.

**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.

**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.

**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 an angle or the side with no collision until landing solidly on top.

**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.

### User 2

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Owner** | **Screen size** | **Operating System** | **Phone Model** |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
| Lorna | 19.5:9 | 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.

**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.

**Result** – No lag, but scene loading has graphical loading glitch.

**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.

**Result** – Successful at being unable to get outside of the 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.

**Result** – Successful, clouds spawn off screen.

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

**Pass/Fail** – Fail.

**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.

**Result** – Most of the time unless the player object stops jumping on the platform.

**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.

**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** – 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.

**Result** – User is able to phase through platforms.

**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 an angle or the side with no collision until landing solidly on top.

**Result** – The user can pass through on the sides of the platform.

**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** – Successful, UI updated constantly

**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** – Yes, high score saves 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, colour can merge when in front of a cloud, size is fine, not hard not read.

**Pass/Fail** – Pass.

### User 3

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Owner** | **Screen size** | **Operating System** | **Phone Model** |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
| Grayson | 19:9 | 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.

**Result** – Touches around the player object at first but understands quickly to touch the object directly. Movement does function.

**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.

**Result** – No lag, except for the scene load graphical glitch.

**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.

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

**Result** – Successful with platforms spawning out of view.

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

**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.

**Result** – Platforms destroy, comments that it takes a while to destroy itself.

**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.

**Result** – Mostly successful.

**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.

**Result** – Is able to phase through the platforms from underneath.

**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 an angle or the side with no collision until landing solidly on top.

**Result** – Boundary and player colliders touching can make the camera shake. Is able to phase through.

**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** – Successfully gets updated.

**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** – Successfully saves data.

**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** – Mobile camera is fitted on the screen in the right corner, obscuring the UI.

**Pass/Fail** – Fail.

### User 4

**Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Owner** | **Screen size** | **Operating System** | **Phone Model** |
| Example | 16:9 Portrait | Android | Samsung Galaxy S10 |
| Chase | 19:9 | 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.

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

**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.

**Result** – No lag.

**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.

**Result** – Successful, aside from object jittering.

**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.

**Result** – Successful spawning off screen.

**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** – Randomised and reachable.

**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.

**Result** – Previous clouds all destroy themselves.

**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.

**Result** – Most of the time player continuously jumps.

**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.

**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 an angle or the side with no collision until landing solidly on top.

**Result** – User can enter platforms from an angle.

**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** – Successfully increase by one.

**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** – Yes, it saves and updates.

**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** – No issue with concentration or usability

## Evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Owner** | **Functionality** | **Usability** | **Overall Experience** |
| Alex | Player jumping can sometimes stop, high score being added to the main scene’s score. | Understands the controls quickly, but the user touches everything around the chicken to move it. | Minimal bugs, non-intrusive bugs, good visual design. |
| Lorna | High score added to normal score in game, stops bouncing when at equal level to top of platform. | Expects the chicken to be in front of the cloud sprite when phasing through, clouds are off-screen when spawned at the edges, colour of main scene UI is too close to the colour of the clouds and can be obscured at brief points, platforms can spawn too far apart from each other along the x axis. | Engaging |
| Grayson | High score being added to main scene score, jumping can stop, scene loading assets visual glitch. | Responds well to the game, aside from trying to touch the areas around the chicken to move it, rather than the sprite directly. | Replayable |
| Chase | Chicken stops jumping, and high score is added to main scene score. | No problems. | Enjoyable gameplay. |

## Reporting

For the testing process, I first recorded the device that the tester would be using, noting the operating system, aspect ratio, and phone model. Half of my testers ended up using my phone for either efficiency for setting up or a phone issue blocking the installation of external applications. After that, I got them to play the game normally, observing their behaviours while they played so I could see how they responded to different aspects of the game. Then I went through my test cases and either asked them to do an action (e.g. “try to escape the screen”) or their opinion on a certain element (e.g. “does the score update?”).

The testing process was successful, having users to test the current state of the game proved to be very beneficial. Many issues that I had not noticed or was desensitised to due to doing personal testing often, were exposed by users who has never played the game before.

I tried doing a range of test cases for the mechanics for as much coverage as possible, mostly it was for functionality, but some were used for special cases that a user might do that causes unforeseen outcomes (like platform phasing at an angle and trying to glitch out of bounds). UI is also something that has multiple cases to test, as there are design elements to it as well as functional uses.

I also tried to get a range of users with different phone sizes, but half of the testers used my mobile device instead. The testers didn’t have an aspect ratio that was too different to what I was using, so it would have been interesting to see how a large screen like a tablet would respond to the game.

My test cases were designed to be explicitly pass or fail. Results explained how the test case passes or fails compared to the expected outcomes of a test case. My test cases were used to look for the core functionality of an element, so any issue that arises but does not affect the functionality of a mechanic is noted in my evaluation.

The jumping player test case was mostly successful in the results of the tested users but was still given a pass because the jumping was functional, except for specific scenarios that I have listed in my bugs. In the instances where the player would stop jumping, could be fixed by moving the player until the jumping restarted, therefore not halting the progression of the game.

**Bugs and Issues**

I found that most issues that were brought up by the testers where often minor issues, nothing that hugely impacted the progression of the game.

High score being added to the score in-game

* If the player plays for the first time, session score starts at 0 and increments properly and saves their high score properly on the title screen.
* If the player leaves and enters the main game again, the score will still start at 0 as intended, but as soon as the player lands on a platform, the high score is added to the current score, continuing to increment and increasing the high score.
* This score is still saved to the high score on the title screen.

Player stops bouncing

* From observation and testing, this seems to be caused by the player landing on the platform at the same level as their position while jumping.
* This means that there was not enough distance between the player and the top of the platform for the player to fall to then jump back up normally.
* The player can manoeuvre themselves back to jumping by moving the player side-to-side on the platform until it starts again.

Scene load glitch

* This is possibly caused by the game loading the sprites and assets when loading the game scene from the title screen.
* This does not affect the gameplay.

Camera shakes when hitting the edge of screen

* Camera follows the player, and the player’s collider against the boundary’s collider begin to collide and because of the persistent movement against each other, shake the camera.

**Recommendations For Improvements**

I would fix the issues and bugs that commonly surfaced during my testing, like the high score properly updating only when the in-game current session score reaches it, as well as change how I approached the continuous jumping on the player to resolve the stopping.

I think the scene loading could be fixed by adding a buffer or a waiting screen until the assets fully load in so that users don’t need to see the behind-the-scenes of the game scene. This would be useful for potential users whose devices may not perform as well as my testers’ devices did.

I would also update the main scene’s score and timer by adding a background to contrast it from the clouds that might briefly pass behind it as the player moves up.

Adding onto this UI, I was informed that devices hold information about where obstacles are on the screen so the UI can morph around it, depending on the device. This would fix the issue where the camera on my device obscured the score text and could be applied to other devices for potential users.

I would also increase the collider on the player object so that users can have more freedom to click wherever on the screen they like to move the player, while also making sure this collider doesn’t affect the other colliders in the scene.