Gesture Based UI Development Project

-

A Unity Application developed to incorporate Mobile Gestures

By John Groves G00367771

David O'Loughran G00362038

https://github.com/johngroves1/Gesture-Based-UI-Development

Contents

Purpose of the application

The purpose of this application is to explore and experiment with several gestures provided by a mobile device. This application incorporates appropriate gestures into a mobile application to enhance the user's experience. The mobile application is a game called "Space Jump" inspired heavily by the popular mobile game Doodle Jump. The application utilizes several different gestures available from the hardware of a mobile device. The goals of this project are:

- To incorporate practical gestures into various aspects of the application which give a natural feel to the overall experience.
- Provide a clean responsive user experience with easy to learn gestures.
- To further our learning of the usage and practices of gestures.
- Provide highly precise and accurate functionally in relation to gesture recognition.

The User Interface.

The user interface is designed for a mobile application, specifically on an Android platform. All UI features are in portrait mode and scaled to the size and resolution of the mobile device. All interfaces follow a pixel space theme referring to the title "Space Jump". Below each interface and its functionality is described in detail as well as a visual graphic showing how each gesture could interact with the interface.

Main menu scene

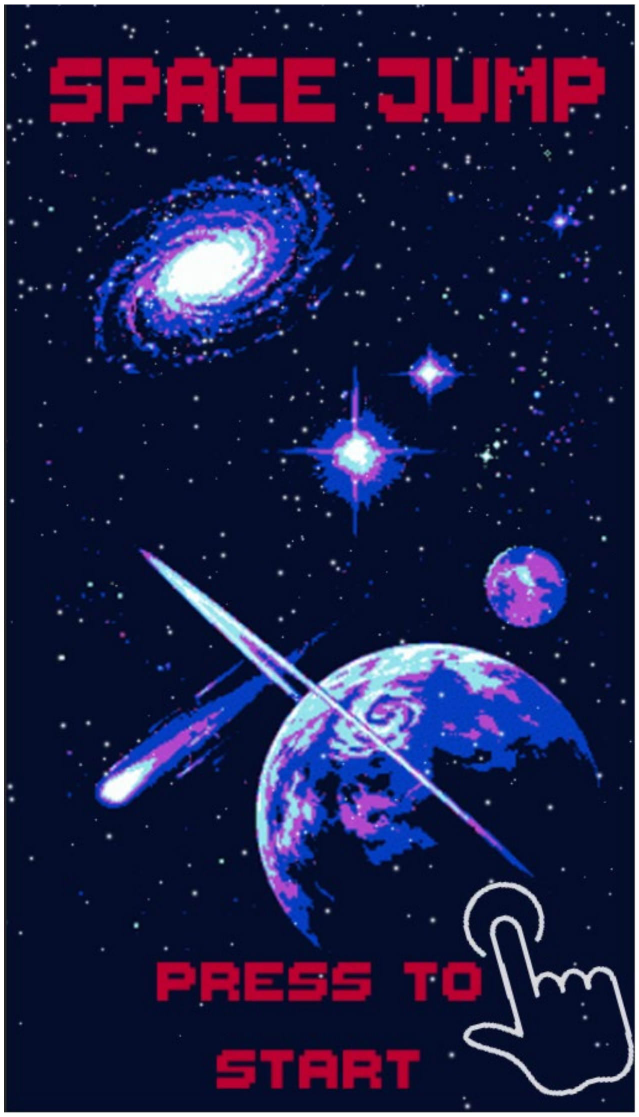


Figure 1: Home screen

The main menu scene sets the premise of the space theme of the application. An ambient dystopian soundtrack is played on loop on the title page, with moving stars in the background. The user has only one option to press the start button to further navigate into the application.

Character selection

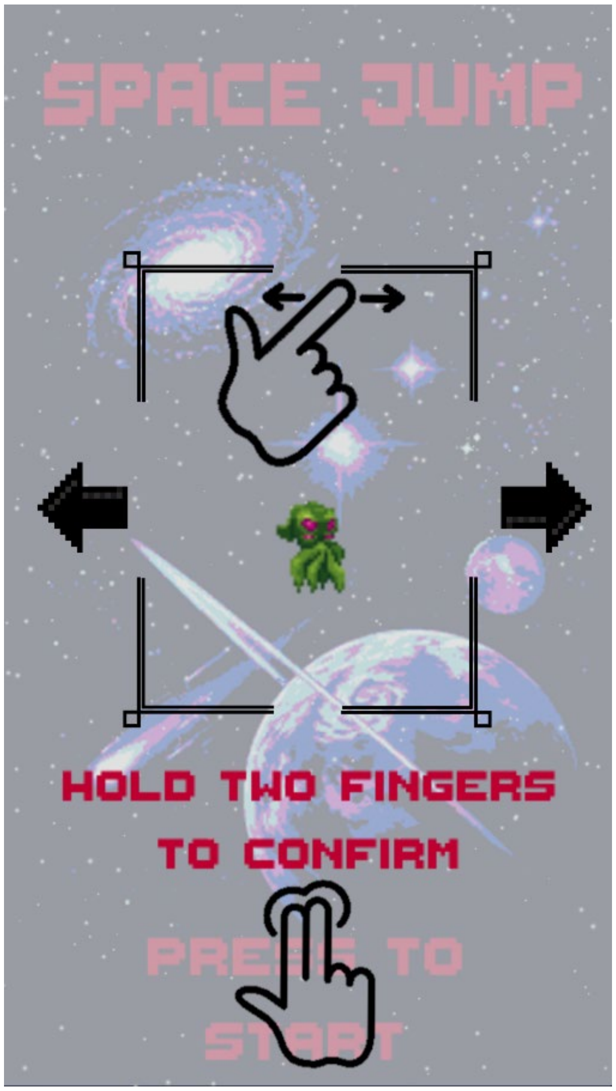


Figure 2: Character selection menu

After starting the application, the character selection interface appears. From this interface the user can select between four unique sprites with their own animations, each character follows the common theme. The user can cycle left or right between the playable characters and once happy with their selection they can confirm to navigate to the main gameplay.

Gameplay Scene

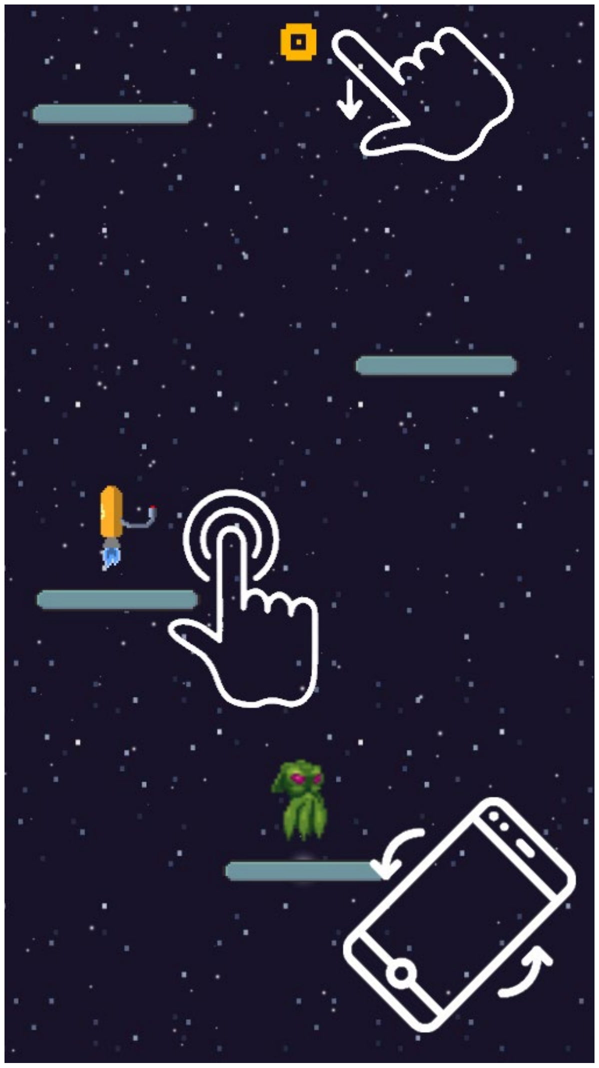


Figure 3: Gameplay screen

The gameplay scene has a more upbeat soundtrack that gradually gets more intense as the game goes on, it follows the space theme and really adds to the immersion of the gameplay. The user's score is displayed at the top of the screen, and the fuel bar of the jetpack once collected is displayed at the top left of the screen.

There are light blue platforms that spawn and bounces the character if the user lands on one. The camera follows the character so as the character bounces upwards from platform to platform, more will spawn in a random location above and the platforms below will be destroyed by a collider off screen. There is a dark blue platform which has a 1 in 7 chance to spawn, this special platform has a much larger bounce than the default one. If the user collects the jetpack power-up they can get an additional boost to their bounce. If the user times it right after bouncing off a special platform the character can jump a huge distance.

Pause Screen

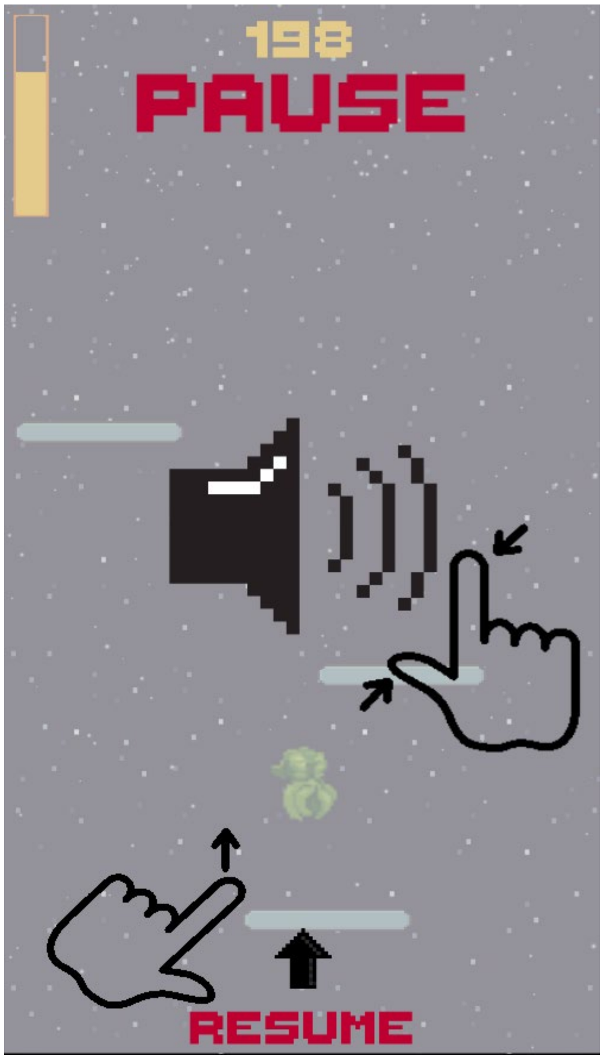


Figure 4: Pause menu

The pause menu is accessible from the gameplay scene at any point during the gameplay, when the menu is activated, all gameplay is stopped. While paused the user can raise or lower the volume of the soundtrack and all in game sound effects.

Game over scene

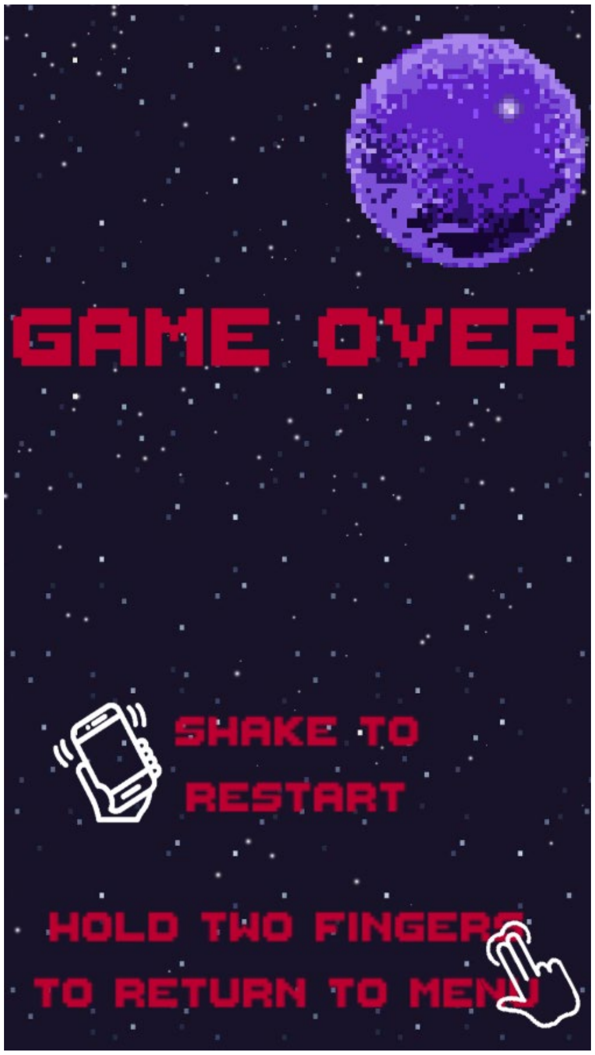


Figure 5: Game over screen

The game over screen appears when the character falls off a platform. The soundtrack changes to a more ambient theme and the user is prompt with two menu options. To restart the gameplay and try again, or to return to the main menu.

Gestures Identified

There are several various gestures available in mobile device, these gestures are the motions made by the user to activate a specific control within the mobile interface. The majority of mobile gestures are performed by the users' fingers such as tap, swipe, and press. Mobile gestures can also involve the users' hand movements such as tilting and shaking.

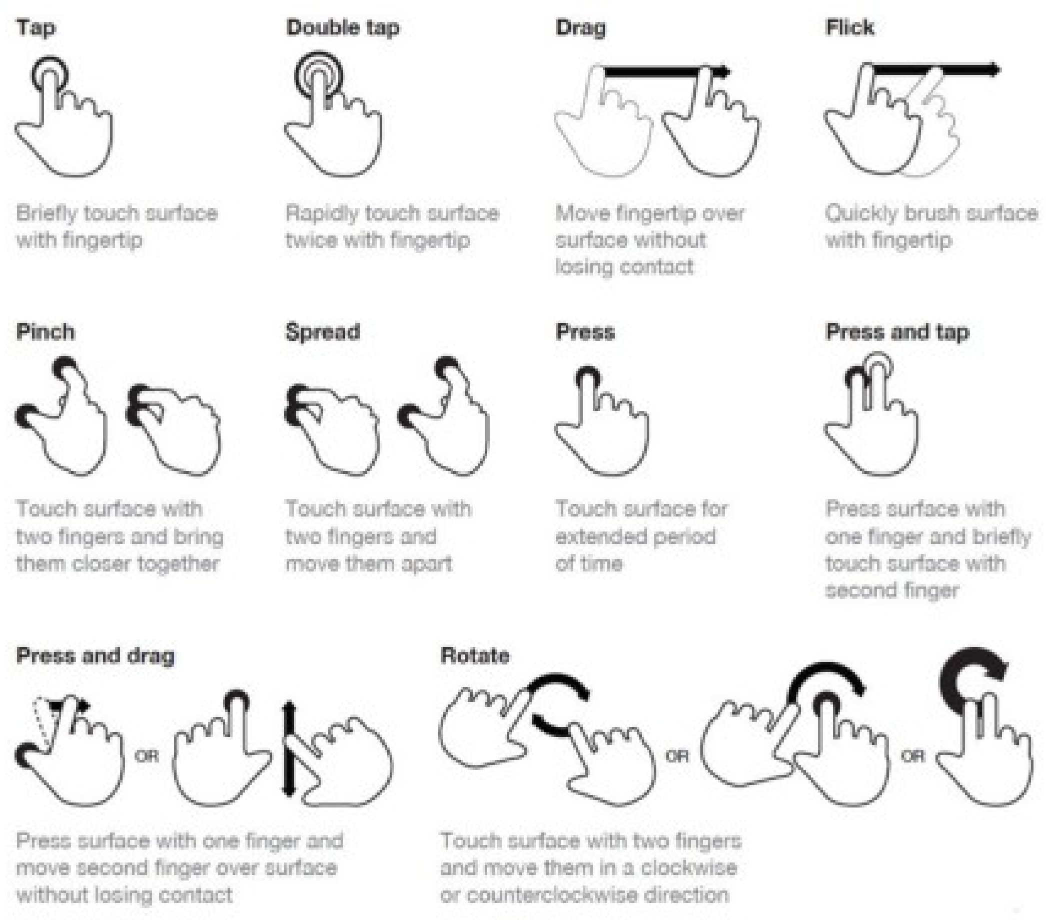

Tap
Briefly touch surface with fingertip
Double tap
Rapidly touch surface twice with fingertip
Drag
Move fingertip over surface without losing contact
Flick
Quickly brush surface with fingertip
Pinch
Touch surface with two fingers and bring them closer together
Spread
Touch surface with two fingers and move them apart
Press
Touch surface for extended period of time
Press and tap
Press surface with one finger and briefly touch surface with second finger
Press and drag
OR
Press surface with one finger and move second finger over surface without losing contact
Rotate
OR
OR
Touch surface with two fingers and move them in a clockwise or counterclockwise direction

Figure 6: Examples of different types of mobile gestures with the use of fingers

The gestures implemented into this application were chosen to make the game feel as natural as possible to the user. Below each gesture used is described in detail and how they interact with each interface and scene within the application

Tap

- A tapping gesture was used to navigate from the main menu to the character selection menu

Swipe

- A left or right swiping gesture was used to swipe back and forth between each character available in the game.
- Using a downward swipe gesture anywhere on the gameplay screen brings down the pause menu.
- When the pause menu is activated, the gameplay is paused. To resume the game, use an upward swipe gesture.

Multi-touch

- A multi-touch gesture using two fingers is implemented to confirm the character selected and navigate to the gameplay.
- A multi-touch gesture using two fingers during the game over screen navigates back to the main menu scene.

Tilt

- A tilting gesture is incorporated to control the movement of the character. If the user tilts their phone in a leftward direction, the character moves left. If the user tilts their phone in a rightward direction, the character moves right.

Long Press

- When the character collects the jetpack power-up, the user can do a long press gesture to activate the jetpack. Once the long press is stopped, the jetpack comes to a halt.

Pinch

- While the game is paused, a pinching gesture with two fingers can be used on the volume icon to shrink the icon. This lowers the volume of the game itself, including the gameplay music and all sound effects.

Spread

- While the game is paused a spreading gesture with two fingers on the volume icon expands the image, this raises the volume of the game and all sound effects.

Shake

- The user can perform a shaking gesture of the mobile device to restart the gameplay while in the game over scene.

In total there are 8 unique gestures implemented into this application. To summarize the use of each gesture:

- **Tap** – For navigation.
- **Swipe** – To switch selected character and to pause the game.
- **Multi-touch** – To confirm a character selection and to navigate back to main menu.
- **Tilt** – Controls the characters movement direction.
- **Long press** – Controls the usage of the jetpack power-up.
- **Pinch** – Lowers the game volume.
- **Spread** – Raises the game volume.
- **Shake** – Restarts the game.

Hardware considered

Two technologies were first considered for this project. These were Vuforia for augmented reality applications or a mobile device for touchscreen and movement gestures.

Vuforia

Vuforia is an augmented reality SDK for mobile devices that enables the creation of augmented reality applications. Developers can easily add advance computer vision functionality to any application, allowing it to recognize images and objects, and interact with spaces in the real world. The SDK supports a variety of 2D and 3D target types which offers a broad range of features and targets to make the AR development process more flexible. Vuforia provides an API in C++, Java and the Unity game engine, the SDK supports both native development for iOS and Android.

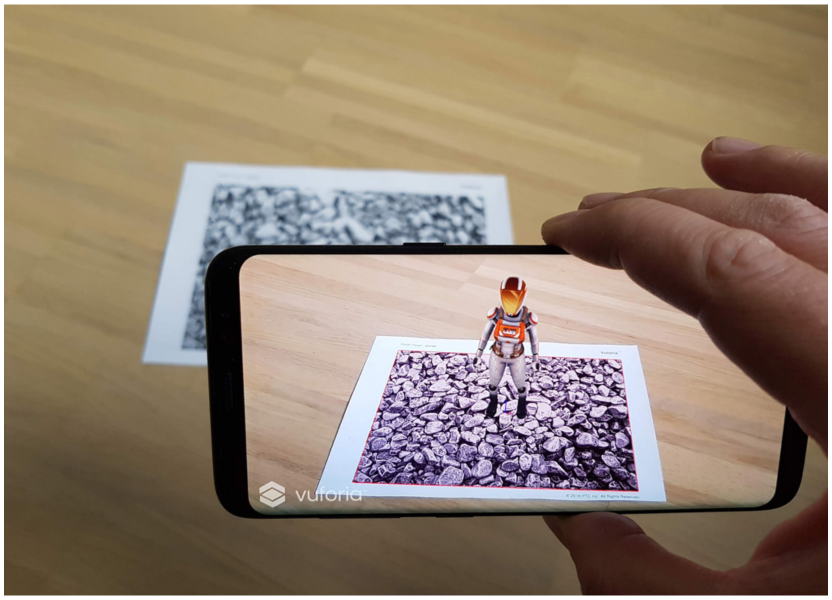


Figure 7: Vuforia Engine Example

Mobile device

The mobile application has several sensors that could potentially be used for a gesture-based application. The accelerometer is an in-built sensor that tracks the different motions of the device, such as shaking, tilting, and rotating. The accelerometer calculates and detects motions using an XYZ axis. The gyroscope is another in-built sensor which tracks all types of rotations, it measures this using angular velocity. The accelerometer senses the axis orientation whereas the gyroscope senses the angular orientation.

Another mechanism to the device which could be used for gestures is the touchscreen. The use of a touchscreen can accurately detect several different finger gestures, some of these gestures are tap, swipe and pinch. The camera of a mobile device could also be used for gesture recognition with the implementation of body and facial detection an example of this would be camera filters that detect facial expressions as a gesture.

Hardware selected

The chosen hardware for this project is the mobile device for its sensors and touchscreen gestures. However, Vuforia was strongly considered for the Doodle Jump clone. The approach for Vuforia would be by using a 2D image to control the character movement by physically sliding the target image in a direction and having a separate target image that spawned the gameplay scene and platforms. After further research this option wasn’t viable as an endlessly scrolling game was awkward to implement into augmented reality.

Both team-members had access to an Android device and an interest in the development of a mobile game. Especially with utilizing the various gestures provided by the mobile hardware. On top of using an Android device as the hardware for the application, the gameplay itself was developed using the Unity Editor. The application was built using native Unity and deployed as an APK for testing on an android device.

Architecture for the solution

Core Technologies

- Unity Engine
- Java
- Android SDK
- Android mobile device

Architecture Design

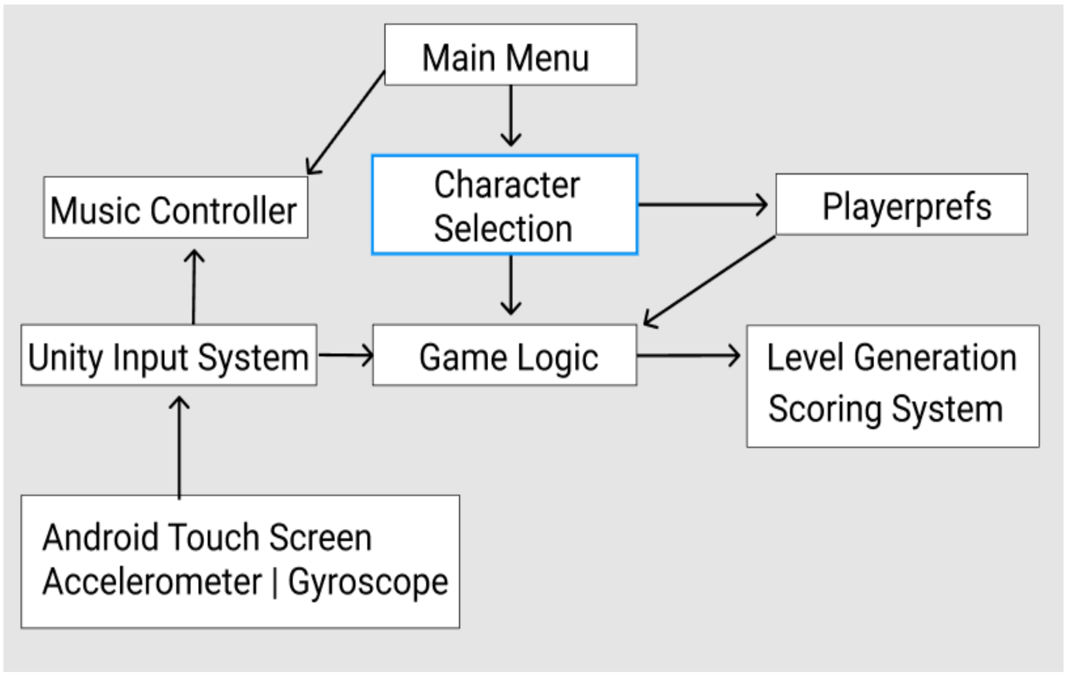


Figure 8: System architecture of components

Gesture Scripts

Multi-touch

Multi-touch gestures are implemented by utilizing Unity’s Input system to detect when two or more fingers are touching the screen.

Press/Hold

This gesture is simple to detect so does not have its own script but instead is utilized throughout different scripts in the application. We simply check to see if the touchCount from Unity’s input system is equal to one on each frame update.

Tilt Controls

Tilt controls utilize the Unity Input System to check the devices acceleration on the x axis and using this value to apply force to our player game object based on how much the device is being tilted.

Swipe Detection

The SwipeDetection script also uses Unity’s Input System to detect when one or more fingers are touching the screen. It starts by getting the users starting touch position and the position of where this touch was ended. Before doing any further calculations the magnitude/distance of this swipe is then checked to ensure the possibility of accidental swipes being detected is minimal. It then compares the position of the start and end touches on both the x and y axis to determine what direction the user has swiped.

Shake Detection

The ShakeDetection script utilizes the phone's accelerometer through Unity’s Input System. We start by getting the acceleration of the device and comparing the accelerations square magnitude and comparing it to a shake threshold to ensure slight movement of the phone is not detected when shaking the device. This threshold was manually tested until the shake detection was not too passive or aggressive to keep a natural feel for the user.

Pinch and Spread to Scale

The PinchScaleController script once again utilizes Unity’s Input system and uses pointer events to detect if the user's finger is being held down on the screen. If this is the case, it then uses the touchCount function from the Input System to make sure two fingers are touching the screen before performing any calculation on the touch positions. We then calculate the distance between each touch on every frame to check if the distance is increasing or decreasing. If the distance between both touches is decreasing, it is clear a pinch gesture is being performed and if it is increasing a spread gesture is being performed by the user. It then changes the scale of the selected image and the volume of the game by multiplying the delta time of 60 frames per second by a predetermined scaling rate as we cannot use the games delta time since this script is used while the game has been paused.

In-game systems

Scoring

The scoring system within this application works by taking the current y position of the player. As the player bounces upwards from platform to platform, the score increases with the y position. The score is displayed at the top of the screen.

Character Selection

The character selection menu cycles through each playable sprite and the subsequent animation. The player can cycle back or forward between characters and once the player decides on a character, they can hold a long press gesture to confirm their selection. A character selection manager script using Unity Engines PlayerPrefs finds the confirmed sprite and loads it into the gameplay scene.

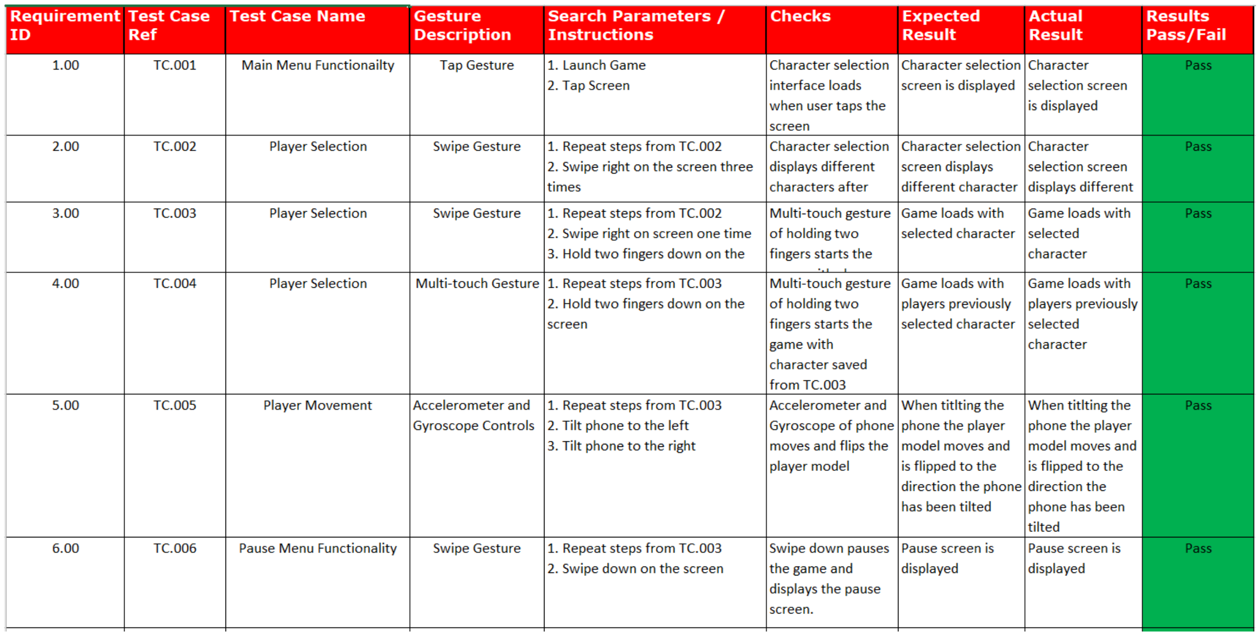
Game Over

The game over mechanic is calculated by getting the current y transform of the player. If the transform is less than 50 units compared to the current score of the player, the game is over and the game navigates to the game over scene.

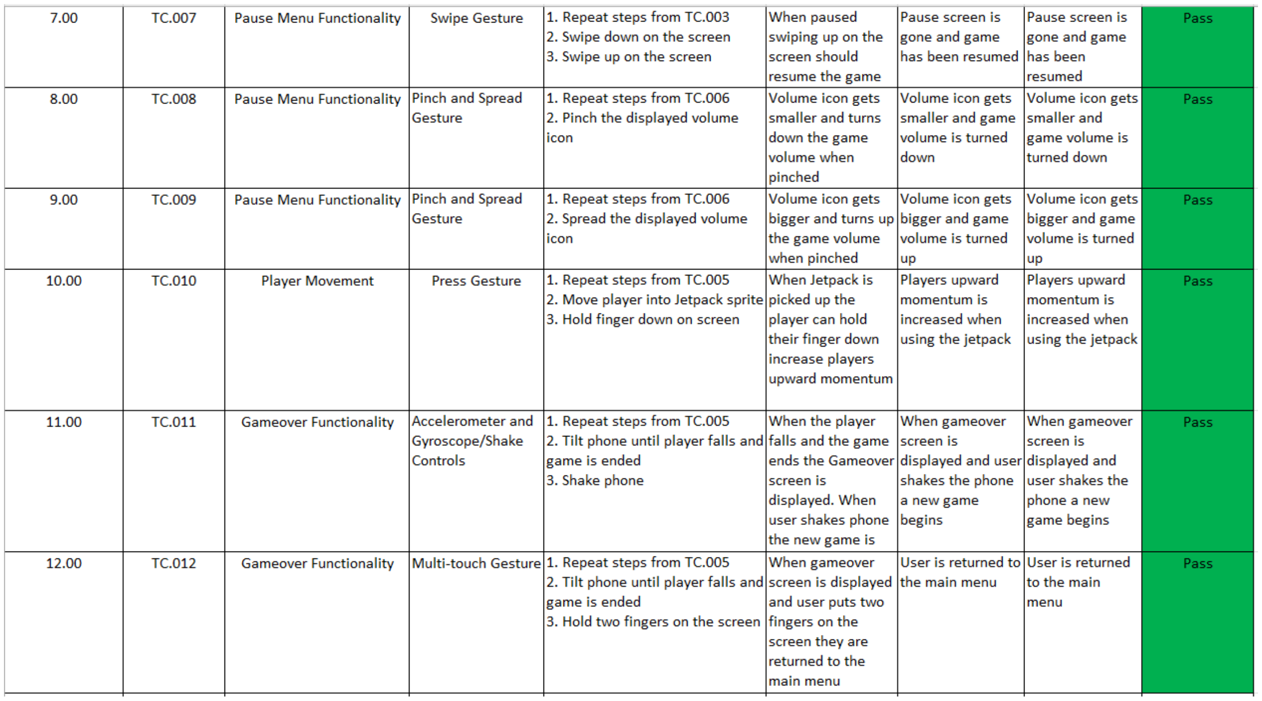
Testing

Testing Approach

Testing played a key role throughout the development process to ensure all gesture functionality was implemented and working as intended. Unit Testing was utilized as each different gesture was manually tested after completion which was a massive help in not just finding issues with the gestures, but also in relation to elements of the user interface.



Requirement ID	Test Case Ref	Test Case Name	Gesture Description	Search Parameters / Instructions	Checks	Expected Result	Actual Result	Results Pass/Fail
1.00	TC.001	Main Menu Functionailty	Tap Gesture	1. Launch Game 2. Tap Screen	Character selection interface loads when user taps the screen	Character selection screen is displayed	Character selection screen is displayed	Pass
2.00	TC.002	Player Selection	Swipe Gesture	1. Repeat steps from TC.002 2. Swipe right on the screen three times	Character selection displays different characters after	Character selection screen displays different character	Character selection screen displays different	Pass
3.00	TC.003	Player Selection	Swipe Gesture	1. Repeat steps from TC.002 2. Swipe right on screen one time 3. Hold two fingers down on the	Multi-touch gesture of holding two fingers starts the	Game loads with selected character	Game loads with selected character	Pass
4.00	TC.004	Player Selection	Multi-touch Gesture	1. Repeat steps from TC.003 2. Hold two fingers down on the screen	Multi-touch gesture of holding two fingers starts the game with character saved from TC.003	Game loads with players previously selected character	Game loads with players previously selected character	Pass
5.00	TC.005	Player Movement	Accelerometer and Gyroscope Controls	1. Repeat steps from TC.003 2. Tilt phone to the left 3. Tilt phone to the right	Accelerometer and Gyroscope of phone moves and flips the player model	When titlting the phone the player model moves and is flipped to the direction the phone has been tilted	When titlting the phone the player model moves and is flipped to the direction the phone has been tilted	Pass
6.00	TC.006	Pause Menu Functionality	Swipe Gesture	1. Repeat steps from TC.003 2. Swipe down on the screen	Swipe down pauses the game and displays the pause screen.	Pause screen is displayed	Pause screen is displayed	Pass



7.00	TC.007	Pause Menu Functionality	Swipe Gesture	1. Repeat steps from TC.003 2. Swipe down on the screen 3. Swipe up on the screen	When paused swiping up on the screen should resume the game	Pause screen is gone and game has been resumed	Pause screen is gone and game has been resumed	Pass
8.00	TC.008	Pause Menu Functionality	Pinch and Spread Gesture	1. Repeat steps from TC.006 2. Pinch the displayed volume icon	Volume icon gets smaller and turns down the game volume when pinched	Volume icon gets smaller and game volume is turned down	Volume icon gets smaller and game volume is turned down	Pass
9.00	TC.009	Pause Menu Functionality	Pinch and Spread Gesture	1. Repeat steps from TC.006 2. Spread the displayed volume icon	Volume icon gets bigger and turns up the game volume when pinched	Volume icon gets bigger and game volume is turned up	Volume icon gets bigger and game volume is turned up	Pass
10.00	TC.010	Player Movement	Press Gesture	1. Repeat steps from TC.005 2. Move player into Jetpack sprite 3. Hold finger down on screen	When Jetpack is picked up the player can hold their finger down increase players upward momentum	Players upward momentum is increased when using the jetpack	Players upward momentum is increased when using the jetpack	Pass
11.00	TC.011	Gameover Functionality	Accelerometer and Gyroscope/Shake Controls	1. Repeat steps from TC.005 2. Tilt phone until player falls and game is ended 3. Shake phone	When the player falls and the game ends the Gameover screen is displayed. When user shakes phone the new game is	When gameover screen is displayed and user shakes the phone a new game begins	When gameover screen is displayed and user shakes the phone a new game begins	Pass
12.00	TC.012	Gameover Functionality	Multi-touch Gesture	1. Repeat steps from TC.005 2. Tilt phone until player falls and game is ended 3. Hold two fingers on the screen	When gameover screen is displayed and user puts two fingers on the screen they are returned to the main menu	User is returned to the main menu	User is returned to the main menu	Pass

Testing Results

Volume Controls: While testing the volume controls it was noticed that the script responsible for scaling the image utilized the game's delta time. However, the game is paused while performing this action so this was changed to the delta time of 60 frames per second as most phones are capped to 60 frames per second.

Player Selection: While testing the player selection functionality it was realized that when the game is loaded for a second time the player's saved model is not displayed in the UI but will be loaded when the user starts the game.

Pause Menu: While testing the functionality of the pause menu it was noticed that the player's fuel will still be decreased when they press down on the screen. This was fixed by setting and checking a Boolean value when the pause menu is active to not update the players amount of fuel. This same set up was also used to stop the player model from flipping directions when tilting the phone in the pause menu.

Conclusions

What we've learned

Prior to this project neither of us had any previous experience with developing games for a mobile device, especially using C++ and incorporating a variety of mobile gestures. Throughout testing and developing this project we both felt we learnt a lot when it came to gesture based UI within a mobile application.

Recommendations for future development

If we managed our time better, we would have developed a scaling difficulty into the application. As well as enemies that spawned throughout the gameplay and potentially more power-ups to expand on the gameplay aspect of the project.

Conclusion – John

I personally believe the approach we took into developing the mobile application with the incorporation of the accelerometer and gyroscope provides an interesting take on the popular game Doodle Jump. I am happy with how the implementation of the various gestures went; I felt we achieved a natural simplistic feel to the user experience. I learned of all the different gestures available within a mobile application, the practically behind them and how to make acceptable use of each.

Conclusion – David

Overall, I am happy with the final product as it allowed me to apply the various skills, I have gained over the years to utilize different components of a mobile device that I had no previous experience with such as the accelerometer, gyroscope, and touch-based gestures. I learned more about the importance of creating a natural and intuitive user experience by utilizing these different gestures to interact with the user interface. While the game itself is simplistic in its design, I feel it takes full advantage of all the different technologies utilized to create a fully gesture-based application.

References

- Doodle Jump Wiki https://doodlejumpsky.fandom.com/wiki/Doodle_Jump_(game)
- Touch Gesture Reference Guide https://www.lukew.com/ff/entry.asp?1071
- Character Selection Screen Tutorial https://www.youtube.com/watch?v=hZdPs1e9bz0&ab_channel=GregEads
- Pixel Space asset: https://assetstore.unity.com/packages/2d/characters/free-pixel-space-platform-pack-146318
- Slime asset: https://assetstore.unity.com/packages/2d/environments/free-pixel-effect-113529
- Alien squid asset: https://assetstore.unity.com/packages/2d/characters/gothicvania-cemetery-120509
- Cosmic asset: https://assetstore.unity.com/packages/2d/characters/cosmicman-167448
- Cosmic character asset https://assetstore.unity.com/packages/2d/characters/cosmicman-167448
- SynthWave Loop pack: https://assetstore.unity.com/packages/audio/music/electronic/synthwave-loop-pack-190942
- Pixel Art FX: https://assetstore.unity.com/packages/2d/textures-materials/free-pixel-art-fx-package-185612
- Tutorial to Sprite Animations https://learn.unity.com/tutorial/introduction-to-sprite-animations#
- Unity Input Documentation https://docs.unity3d.com/ScriptReference/Input.html
- Unity Pointer Events Documentation https://docs.unity3d.com/2018.3/Documentation/ScriptReference/EventSystems.PointerEventData.html