Design Document

This document outlines the iterative process that is being used to develop the mobile game *Island Hopper*. The game's development has been divided into 3 distinct phases with each phase having its own predefined milestones. The first phase of the development includes research, planning and the creation of the game's core mechanics whilst the second phase includes the development of a method of progression within the game and lastly the third phase integrates User Interface into the game along with elements to enhance game feel.

Phase 1

Research

The development of *Island Hopper* began with the establishment of central design goals that would serve as the foundation for the rest of the game's development. The primary design goal was to create a game that appeals to the largest audience possible. According to Newzoo Research[(1)] the mobile gaming market makes up almost 50% of the global gaming market when compared with PC and Console games. Statisca also forecasted that the number of mobile gamers will reach 2.6 billion by 2021[(2)]. It was evident that to ensure the largest audience was reached the game would have to be a mobile game.

Further research into the most popular mobile games of all time also yielded useful information that was used to formulate secondary design goals for *Island Hopper*'s development. At the beginning of *Island Hopper'*s development, the 5 most-played mobile games of all time were: Pokémon Go[(3)], Subway Surfers[(4),] Despicable Me: Minion Rush[(5)], PUBG Mobile[(6)] and Candy Crush Saga[(7)]. Playing and analyzing these games proved that while they each have elements that make them unique, there are a few common traits that can be found in all of them. The first of these traits is that each of them contains a gameplay loop in which the player performs some actions and then once that loop is finished the player gets rewarded, progresses in some way and then does it all again. This led to the establishment of the next two design goals for *Island Hopper*: The game should have a satisfying and replayable gameplay loop as well as a progression system. The other noticeable trend amongst these games is that 4 out of the 5 of them are played in a portrait orientation. This may seem intuitive but mobile expert, Steve Hoober confirmed that this is preferable when he conducted a study of 1333 people and found that 90% of them use their mobile device in portrait orientation[(8)]. This lead to the final design goal for *Island Hopper*: The game should be developed in portrait orientation.

Development

With the design goals established the next step was to start developing the games core mechanics. Instead of having three channels which players move between like Subway Surfers and Dispicable Me: Minion Rush, *Island Hoppers* contains small platforms the player can jump between. The first iteration was solely focused on mechanics, so it was developed using standard 3D Unity Objects like cubes and spheres. The platforms were made into prefabs and a GameManager Game Object was created to store the necessary scripts required to run the game. The first of these scripts was the CreateNodes script, which uses a Random.Range() function to calculate a random x-coordinate for a platform within the bounds of the screen and the Instantiate() method to instantiate a platform at that position, a for loop then repeats this process numerous times to create a path of platforms the player can move between. Conditional statements were also added to the code to ensure the player can only move to the platform in front of them preventing them from skipping platforms and from moving backwards. A MovePlayer() script was attached to the Player object, enabling the player to move. This movement was done using a Ray which is cast from the position the player touches on the screen utilizing ScreenPointToRay(). If the Ray collides with a platform the player's character will move to that platform's location using Vector3.Lerp().

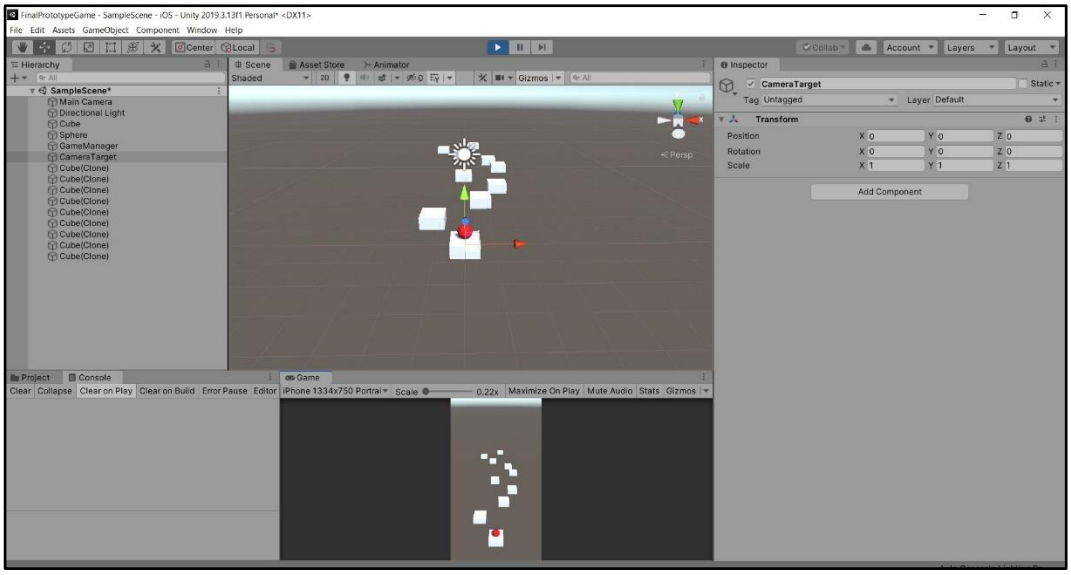


Figure 1: Creation of Nodes and Player Movement

At this stage the camera was stationary as the player moved off into the distance so the next thing that needed to be developed was a Camera Controller script. Simply attaching the camera as a child to the Player object would result in it moving in all directions according to the Player's movement so instead a CameraTarget object was created with a MoveCamera script. This object would move forward at a progressively increasing speed as the player moved and if the player did not move fast enough from platform to platform it would overtake the player resulting in the player losing. Code was added to the MovePlayer script to ensure the player was always a certain distance in front of the CameraTarget .

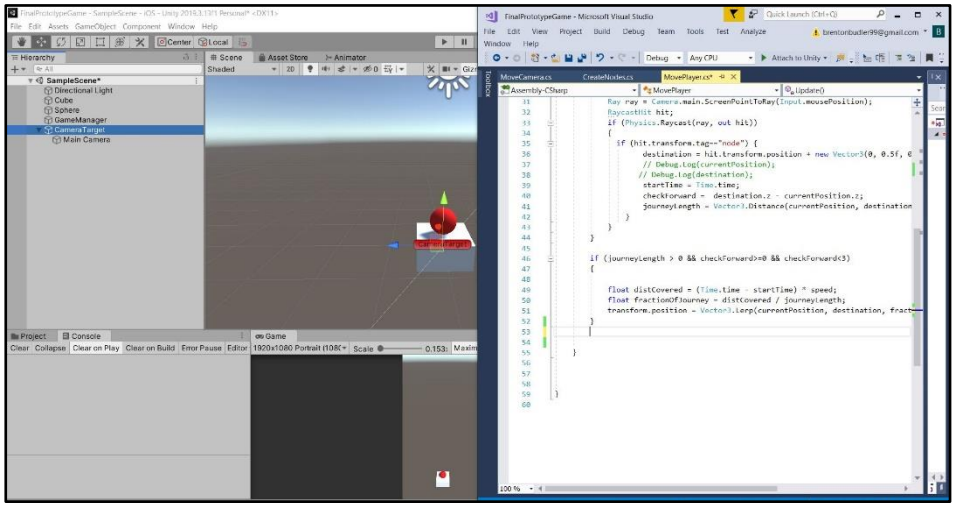


Figure 2: Camera Target and Script

After completing the above scripts, the first round of playtests was conducted. These were simply concept playtests and the play-testers were made aware of this. The play-testers were asked the following questions after playing: "Does the concept of the game appeal to you?", "Would you want to play it again, if not why?" and "Did you experience any issues with the gameplay?". The feedback was mostly positive from a concept standpoint but there were issues in the gameplay and replayability of the game that were identified resulting in several changes being made. An issue that almost all of the play-testers experienced was that the game was not responsive enough and that if they tapped on a platform the character would not always jump to that platform. This was largely due to the fact that the Box Collider on the platform prefab was the exact same size as the platform so if the player just slightly missed the platform the character would not move there. To combat this, the collider's size

was increased to be slightly largely than the platform allowing for some leeway. The next issue pertained to replayability, with play-testers noting that having the same movement mechanic for long periods of time became monotonous and deterred them from wanting to play again. To combat this, a new movement mechanic was introduced into *Island Hoppers*. After moving from platform to platform for a while the player would reach a larger platform which they would jump to and then have drag on the screen from side to side to dodge obstacles.

The development of these larger platforms involved the creation of three distinct areas. The Start Area; the area the player would jump to from a smaller platform, changing the movement controls. The Central Area; the area the player would move on to avoid obstacles. The End Area, the area in which the player would jump from the larger platform to a smaller platform changing the movement controls back. The new movement controls were created using the TouchPhase.Moved method and the .deltaPosition attribute. Checking the tags of the object the Player had collided with was used to determine which movement controls were currently in effect.

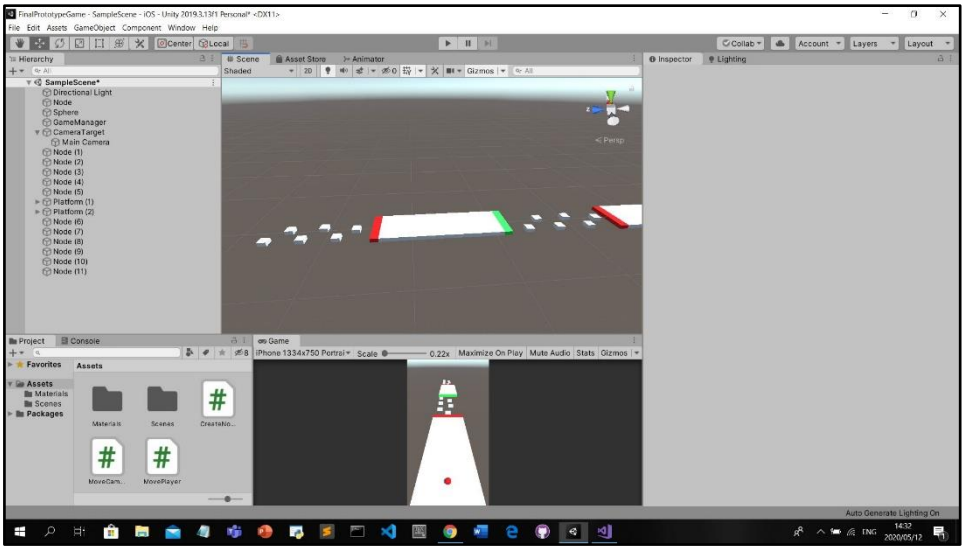


Figure 3: Start, Central and End Areas of Larger Platforms

Enemy and Environment prefabs were also created for the larger platforms, these were different coloured cube objects with colliders that were given specific tags. If the player collider with an Environment object they would be able to move around it and carry on going whilst if they collided with an Enemy object, the gameplay loop would end, and they would have to try again. Procedural generation using random numbers and the Instantiate() method was used to create versions of these prefabs along the players path on the larger platforms.

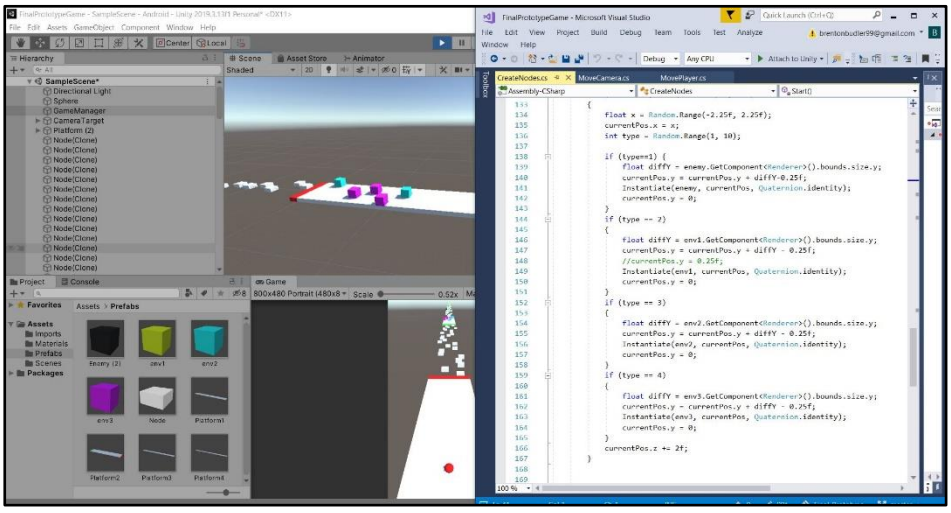


Figure 4: Procedural Generation Code for Environment and Enemies

After the larger platforms were integrated into the game, second round of playtests were conducted using new playtesters. These play-testers were asked the same questions as the first group once they had finished playing. Their feedback showed that the incorporation of a second movement mechanic had increased the likelihood of players wanting to play the game again, but it also resulted in several new technical issues. The first of these issues occurred when a player moved to the side of one of the larger platforms, instead of falling off the edge, the Player object would freeze and then move back to the Start Area of that platform. This was happening because the colliders that determine which movement mechanic was currently in effect were set up using the OnCollisionExit() method and when the Player object moved over the edge, it left the platform's collider resulting in it freezing. To combat this, new Box Collider objects were placed on either side of the larger platforms, these colliders did not have any Mesh Renderers and were set as Triggers. If the Player object collided with these colliders their movement would be changed to a third mechanic resulting in them falling over the edge and the gameplay loop ending. There other technical issues were due to misplaced colliders or incorrectly assigned collider tags which could easily be rectified.

With the core mechanics in place and the technical issues fixed, the next step was to begin replacing the standard 3D objects with more visually stimulating assets. The theme that was chosen for the game was Island's as the different sized platforms resembled islands that the player could move between and the name *Island Hopper* was chosen as a play on words for the popular expression "Island hopping" which is used to describe travelers who move from one island to the next[9].

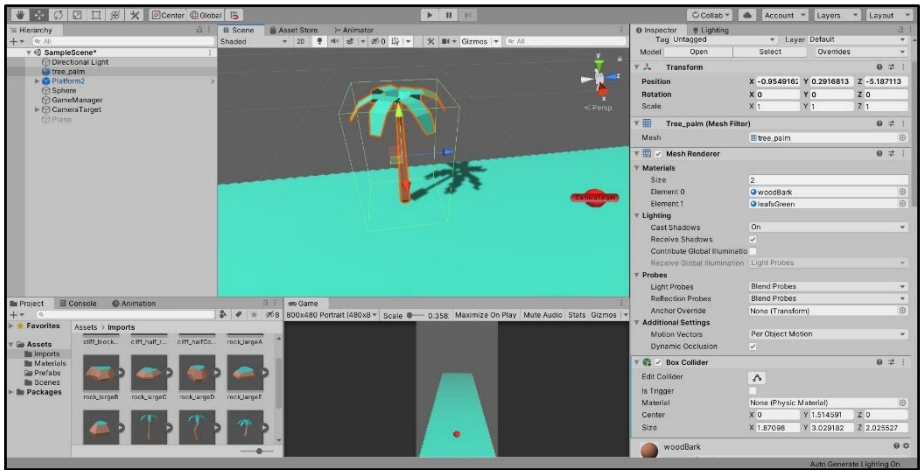


Figure 5: Incorporation of Visual Elements

After the game was reskinned, the third round of playtesting was conducted. Up until this point, not much attention had been payed to who the play-testers were, as the focus was solely on the mechanics but because the core mechanics were complete that focus needed to shift. As the primary design goal of the game's development was to appeal to the largest audience possible, play-testers of different ages, genders and experience levels all had to be included in the process. It is important to note that *Island Hopper* was developed during the worldwide pandemic COVID19 which resulted in people not being able to leave their homes, so the diversity of people that could playtest the game was limited. This round of playtesting involved builds of the game being sent to other developers and then those developers asking individuals they live with to test it and fill in a questionnaire consisting of the following questions:

1. Does the concept of the game appeal to you? If no, why?
2. Would you share it with your friends and family? If no, why?
3. Does the aesthetic of the game appeal to you? If no, why?
4. Do you think the game appeals to a specific audience?
5. Is the gameplay loop satisfying and rewarding?

The feedback from these questionnaires included a lot of interesting insight into the way player's were perceiving the game, but the most noticeable trend in the feedback was that the game still wasn't engaging enough and that the gameplay loop got boring resulting in players not being motivated to play again. One of the primary factors research attributes to a game being engaging is that it should be challenging but not frustrating as easy games are boring and games that are too difficult become frustrating[10]. Reflection on where *Island Hopper* was at this stage in its development made it evident that the game was just too easy. The enemies were stationary so could be easily avoided and the only way a player could lose was by falling off the edge or falling behind. To improve engagement in the game, the first change that was made was to give the enemies movement, this was done using a Coroutine and the transform.translate() method which would move the enemies up and down a predefined path at a certain rate.

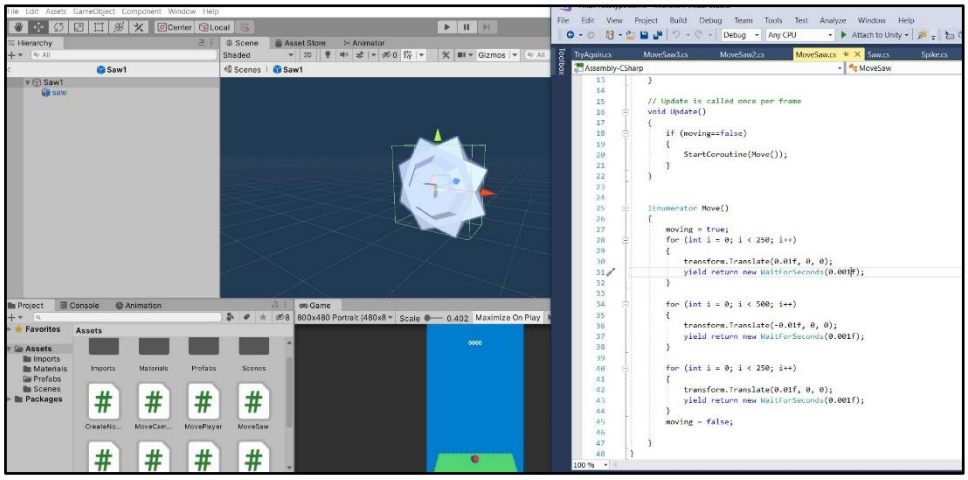


Figure 6: Enemy Movement

At this point, all of the engagement occurred on the larger platforms and there weren't any elements that improved engagement along the path of smaller platforms. As a result, a new Enemy object was introduced into the game. These enemies were spikes that would spawn randomly on smaller platforms which would stay up for a certain amount of time before going down, allowing the player to move passed easily. This forced the player to concentrate more, instead of just aimlessly moving from one platform to another, as they would need to time their jumps correctly in order to survive.

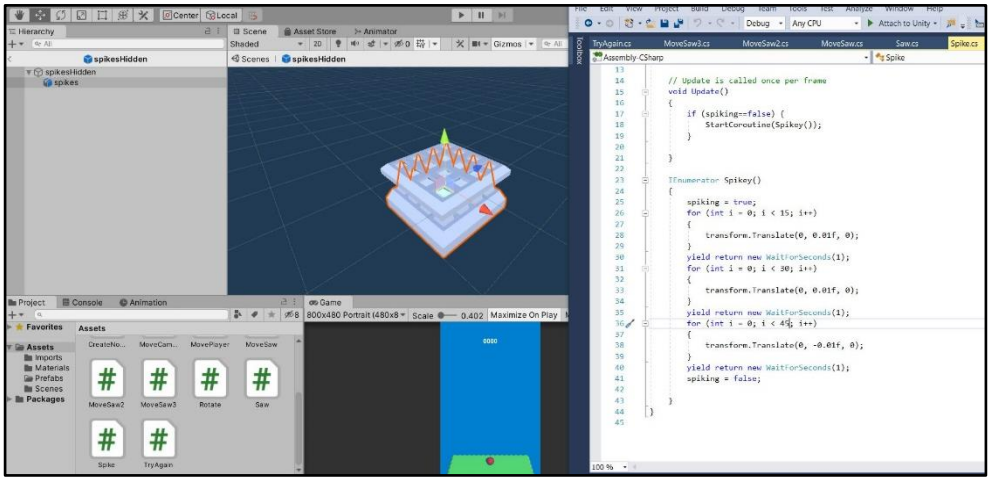


Figure 7: Development of Spikes

The next factor that research attributes to creating engagement in a game is knowing your target audience and what motivates them. This motivation is said to come from 4 different factors: Friends, Leaderboards, Goals and Discoveries[10]. Due to *Island Hopper*'s large target audience, Phase 2 of its development will focus on how all of these motivations can be incorporated into the game.

Phase 2

Research

Whilst the first development phase focused on the creation of a satisfying and engaging game loop, the second phase was focused on developing progression in the game. Josh Bycer[11] explains that progression in a video game can be divided into two categories: player and game. Player Progression is the player learning the rules and mechanics while developing their skills. On the other hand, Game Progression refers to the unlocking of new elements over the course of the game. Bycer[11] explains that Meta-Game Progression is a progression technique that falls perfectly between these two types of progression.

Meta-Game Progression Systems are game which feature a main game system and a secondary one that adds permanence between play sessions of the main game. The primary benefit of using the meta-game design is that the act of adding permanence goes a long way towards improving replay-ability. A key factor that *Island Hopper* still lacked.

The main game system was the game loop that had been developed in Phase 1, but now a secondary game system had to add the required permanence to the game. In order to continue satisfying the design goal of appealing to the largest target audience of mobile gamers possible, further research was done to find a popular game system that could be incorporated alongside the game loop. Base building games proved to be extremely popular among mobile users[12]. It was therefore decided that the secondary game system required for the Meta-Game Progression would be a base builder.

Development

The development of the base builder began with the creation of node Game Objects the Player could interact with. A large map of these nodes was created along with a script which allowed the Player to select a specific node.

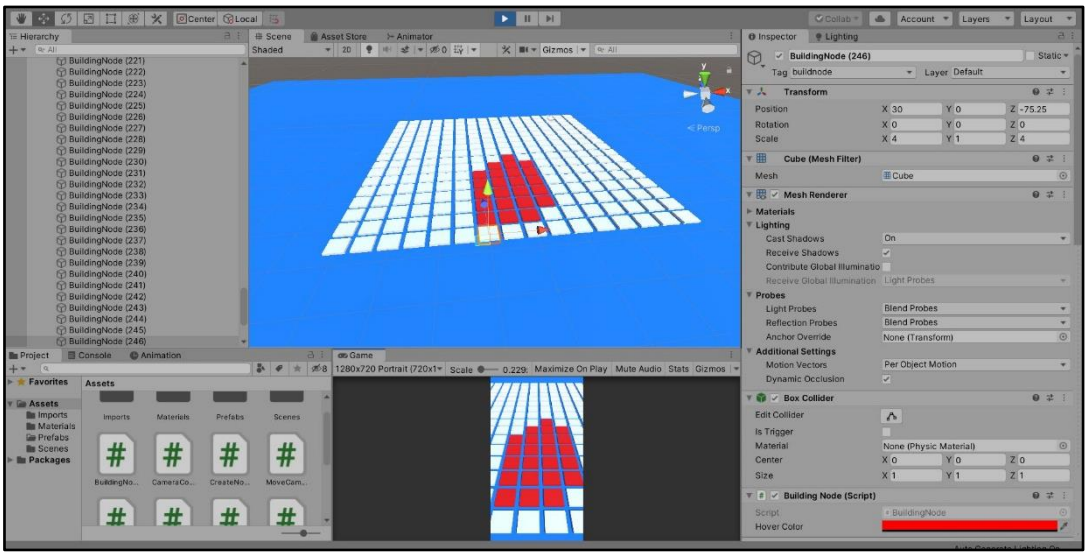


Figure 8: Building Nodes being selected

The next important element for the base builder was a Shop which would allow the Player to view the different available buildings and select which one they wanted to construct. The Shop was created using a simple Panel UI element with several buttons on it. Due to the next phase, Phase 3, being solely focused on the development of the UI elements, the UI that was implemented in this phase was purely to test various mechanics.

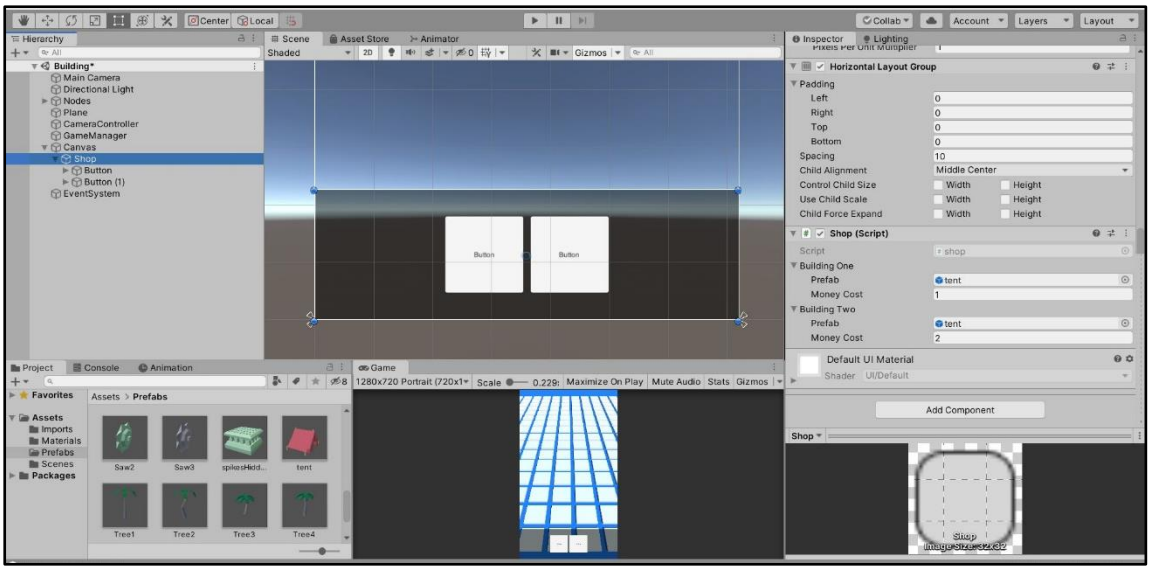


Figure 9: Incorporation of Shop UI

To ensure the secondary game system was aligned with the theme established in the primary game loop of hopping from one island to another, instead of having numerous positions on the map where the player could place buildings , there would only be one central island which the Player could expand upon. This expansion would be modular with each node being a new section of the island. Instead of using a square based node with only four sides, hexagons were used instead; giving the player multiple directions to expand their island. These hexagonal shapes did however clash with the square based nodes in the main game loop, so for continuity reasons the nodes which the player jumps between in the main game loop were also changed from squares to hexagons.

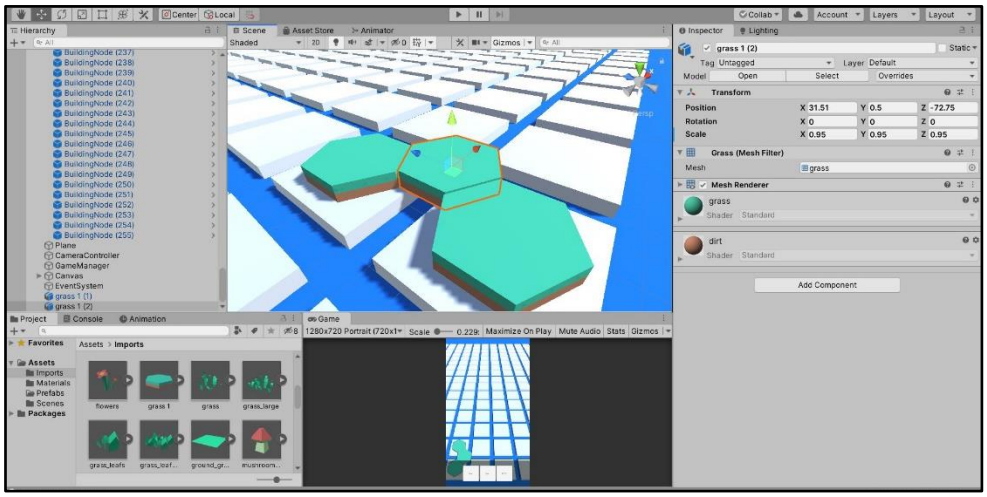


Figure 10: Replacement of Squares With Hexagons

A script then had to be coded to control the connection between the button selected in the shop UI and the building that would be built on the node. The Player would only be allowed to build a specific building if they had enough money, so a check was also put in to ensure the Player could afford the selected building. Once a building was selected a second UI element would appear in place of the shop which would allow the Player to confirm their selection, rotate the building or cancel their selection entirely.

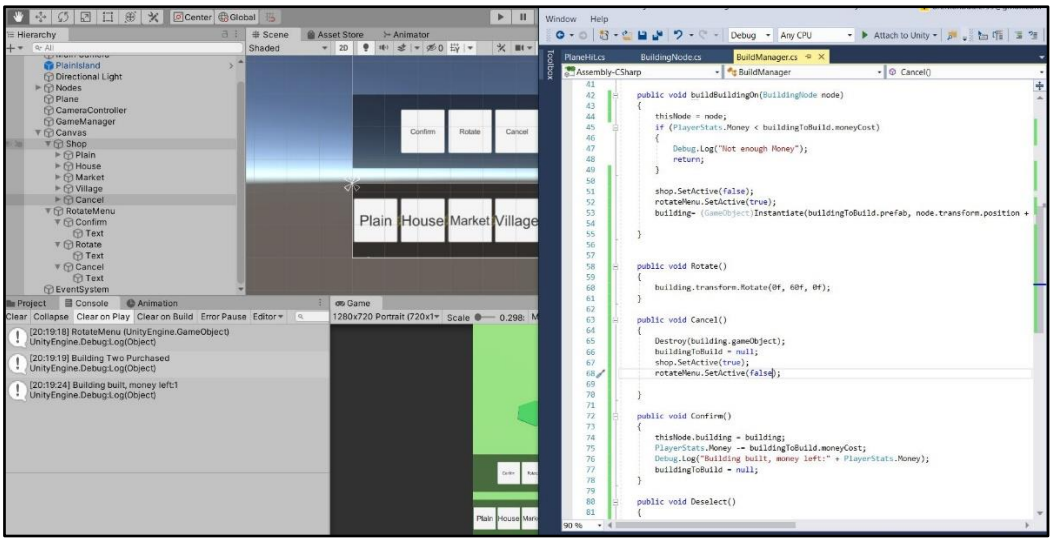


Figure 11: Linking UI with Building

The next step was to find a way of linking the permanent progress made in this new secondary mode with the main game loop. Initially the link was made through currency, as the Player could spend the money they made in the main game loop on building to expand their island in this mode. To further integrate the two modes, resources such as wood and stone would be incorporated into the main game loop, these resources would then be used to construct building in the base builder. The amount of money, wood and stone the player had would be tracked using a PlayerStats scripts and would be reduced by the respective values when a new building was built.

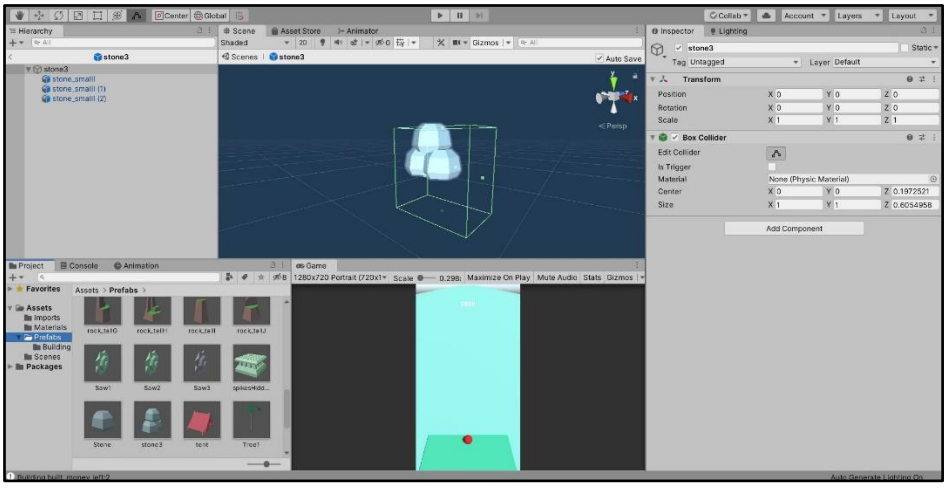


Figure 12: Incorporation of Resources into Main Game Loop

With the necessary calculations to modify the values in each mode, the first round of playtests was conducted for Phase 2. For this playtest however, the player was given an unlimited amount of each resource and asked to build whatever they want. They were then asked questions pertaining directly to the base builder mechanics. The main feedback that was received was that although the building mechanics worked well, the base builder was not very immersive as the Player could only zoom in and out and could not move the camera around. To combat this a Camera Controller script was developed to allow the Player to zoom in and out but to also pan around the map.

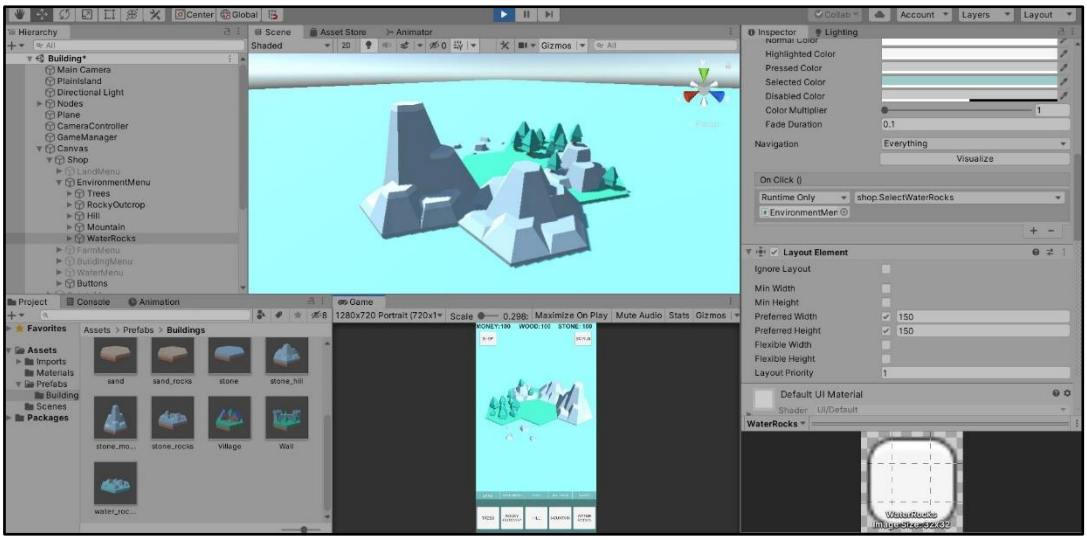
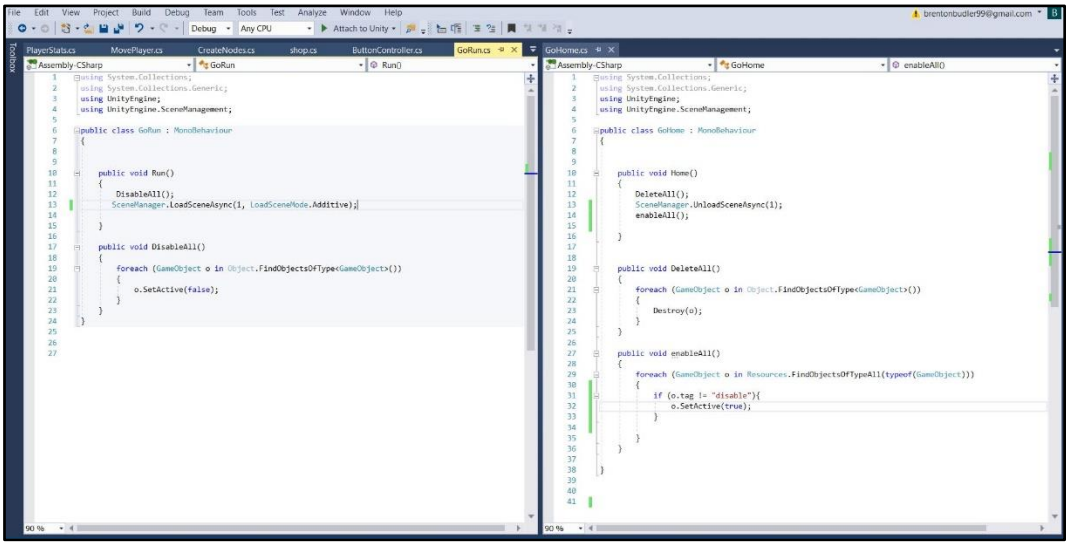


Figure 13: Game State during a Play Test

Before conducting the next round of playtests the two Scenes that had been created, One containing the main game loop and one containing the Base Builder, had to be linked. Unity has a way a Scene can be loaded in the background of the current scene, namely LoadSceneAsync, however when unloading the scene any game objects that were Instantiated during run time would not be deleted. Below is a screenshot of the workaround solution which deactivates all of the Game Objects in the base builder before loading the main game loop and then once the main game loop is over, deletes all game objects from that scene and then proceeds to activate the objects from the initial base builder scene.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GoRun : MonoBehaviour
{

    public void Run()
    {
        DisableAll();
        SceneManager.LoadSceneAsync(1, LoadSceneMode.Additive);

    }

    public void DisableAll()
    {
        foreach (GameObject o in Object.FindObjectsOfType<GameObject>())
        {
            o.SetActive(false);
        }
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GoHome : MonoBehaviour
{

    public void Home()
    {
        DeleteAll();
        SceneManager.UnloadSceneAsync(1);
        enableAll();

    }

    public void DeleteAll()
    {
        foreach (GameObject o in Object.FindObjectsOfType<GameObject>())
        {
            Destroy(o);
        }
    }

    public void enableAll()
    {
        foreach (GameObject o in Resources.FindObjectsOfTypeAll(typeof(GameObject)))
        {
            if (o.tag != "disable"){
                o.SetActive(true);
            }

        }
    }
}
```

Figure 14: Solution for Concurrent Scenes

At this point, the second round of playtests for Phase 2 was conducted, with Players needing to collect resources in the main game loop and then using these resources to construct building in the base builder. The feedback was mostly positive, but the biggest question Player's had was "What is the point of the buildings?" They understood that they needed to collect resources and then construct buildings, but they did not understand how the buildings benefited them in any way. Each building was then re-evaluated to see if it could be transformed into something that could give it purpose or if it was purely being used for visual appeal. All environmental elements that could be built such as land, trees, hills and mountains were still included for visual purposes but each other building was

either given a new purpose or removed from the building menu. Whilst Houses had an implied purpose, other buildings had none.

To give these other buildings purpose a new subsystem was implemented into the game; farming. Players could construct farmland on their island upon which wheat would grow. This would could then be taken to the windmill and turned into flour, this flour could then be taken to a bakery and used to make bread. This sub-system not only gave several buildings purpose but also allowed for more progression to be included as players need to get the resources to construct the necessary buildings in order to turn the wheat into bread.

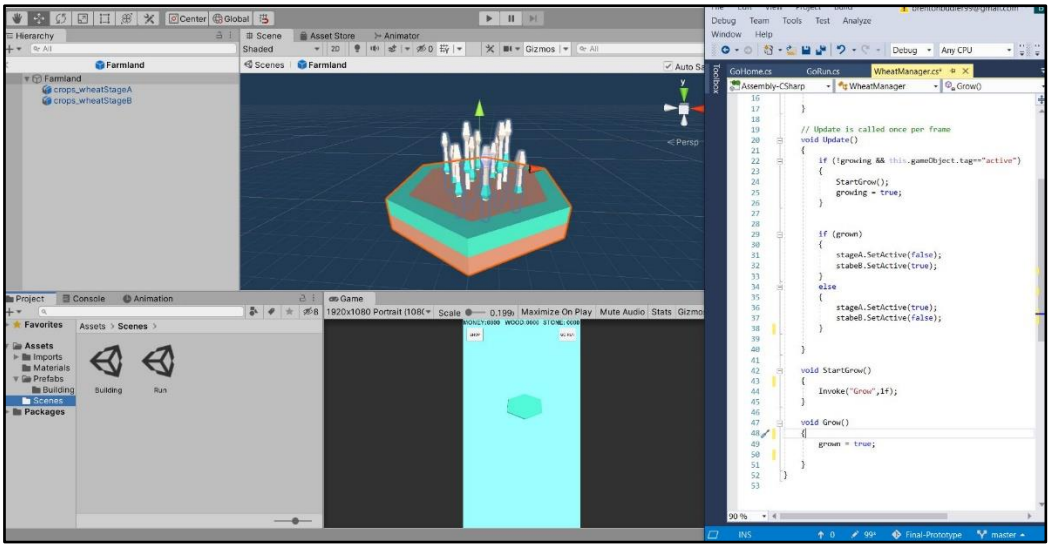


Figure 15: Script causing wheat to grow on Farmland

Each of these resources; wheat, flour and bread were added to the temporary UI to allow the Player to see how much of each they had with the end goal being them taking the bread to the market where they can sell it for money which they can reinvest into other buildings. The market UI as well as UI which indicates that the wheat is fully grown, or that the windmill has made flour, or that the baker has made bread will be developed in the third phase however the mechanics have been implemented and this process can be accomplished by constructing the necessary element and then tapping the building to begin the process, provided you have the required resources (you need wheat to make flour and flour to make bread.)

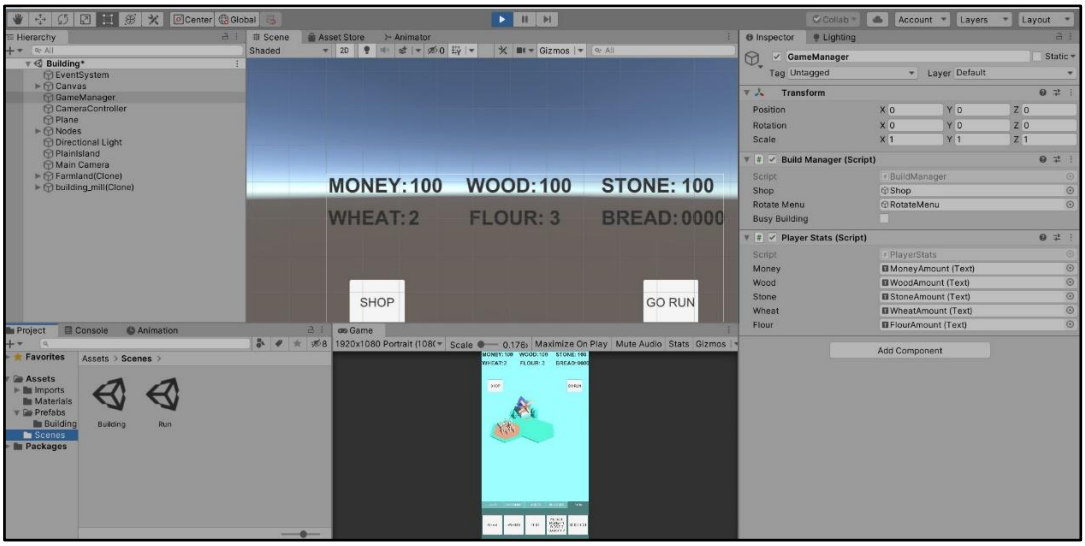


Figure 16: Incorporation of new resources into temporary UI

References:

1- Warman, P., 2018. Newzoo Cuts Global Games Forecast For 2018 To $134.9 Billion; Lower Mobile Growth Partially Offset By Very Strong Growth In Console Segment | Newzoo. [online] Newzoo. Available at: <https://newzoo.com/insights/articles/newzoo-cuts-global-games-forecast-for-2018-to-134-9-billion/> [Accessed 20 May 2020].
2- Gough, C., 2018. Number Of Global Mobile Gamers 2021 | Statista. [online] Statista. Available at: <https://www.statista.com/statistics/748089/number-mobile-gamers-world-platform/> [Accessed 20 May 2020].
3- Webster, A., 2019. Pokémon Go Spurred An Amazing Era That Continues With Sword And Shield. [online] The Verge. Available at: <https://www.theverge.com/2019/2/28/18243332/pokemon-go-sword-shield-franchise-history-niantic-nintendo-switch> [Accessed 20 May 2020].
4- Luton, W., 2019. What We Can Learn From... The Future. [online] GamesIndustry.biz. Available at: <https://www.gamesindustry.biz/articles/2019-12-19-what-we-can-learn-from-the-future> [Accessed 20 May 2020].
5- App Store. 2017. Minion Rush. [online] Available at: <https://apps.apple.com/jp/app/minion-rush/id596402997?l=en> [Accessed 20 May 2020].
6- Naik, K., 2019. PUBG Mobile Gets 600 Million Downloads. [online] IGN India. Available at: <https://in.ign.com/playerunknowns-battlegrounds-mobile/142341/news/pubg-mobile-gets-600-million-downloads> [Accessed 20 May 2020].
7- Alderman, N., 2014. Why Candy Crush Saga Likes To Play On Your Sweet Tooth. [online] the Guardian. Available at: <https://www.theguardian.com/technology/2014/jun/23/candy-crush-saga-freemium-games> [Accessed 20 May 2020].
8- Hoober, S., 2014. How Do Users Really Hold Mobile Devices? :: Uxmatters. [online] Uxmatters.com. Available at: <https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php> [Accessed 20 May 2020].
9- www.dictionary.com. 2020. Definition Of Island-Hopping | Dictionary.Com. [online] Available at: <https://www.dictionary.com/browse/island-hopping?s=t> [Accessed 21 May 2020].
10- Ollero, C., 2015. What Makes Games Engaging?. [online] Ennomotive. Available at: <https://www.ennomotive.com/the-power-of-engaging-games/> [Accessed 21 May 2020].
11- Bycer, J., 2013. The Procession Of Progression In Game Design. [online] Gamasutra.com. Available at:<https://www.gamasutra.com/blogs/JoshBycer/20130523/192906/The_Procession_of_Progression_in_Game_Design.php> [Accessed 10 June 2020].
12- Apps for Android and iOS. 2017. Base Building Games For Android & Ios. [online] Available at: <https://freeappsforme.com/base-building-games/> [Accessed 11 June 2020].