Design

Introduction

The application for this project is called Animal Simulator. It’s an open world sandbox simulator game where you can choose a wild animal, such as a deer, and explore the world at your own pace. We encourage creativity and independence in this game and give you the freedom to play in whatever way suits.

Animal Simulator is developed using Unity and C# programming language. In this chapter we will describe the process of how we designed Animal Simulator. It will contain details about the fundamental design factors of the game, such as the user interface, fonts and colour schemes, block diagrams of the user navigation, flow charts of the game mechanics and technologies used.

The Program Design will convey what technologies are used to help implement the game mechanics and a general overview of how it works. The User Interface Design will convey the design and layout of Animal Simulator’s user interface, including references to similar game interfaces and how we utilised them in Animal Simulator.

Program Design

Technologies

Unity

The application that is used to create this application is Unity. It’s website can be accessed from this link: <https://unity.com/>.

Unity is an excellent application for game development that uses C# and offers a graphical interface. It has different plans for students, businesses, and enthusiasts. Most of their plans are royalty-free. Unity focuses more on complex 3D game development. Many successful games like Fall Guys, Rust and Fire Watch were created with Unity. It provides a sophisticated GUI workspace which makes it easier to create 3D worlds and adjust 3D assets. Users can see the changes as they go along in writing their code.

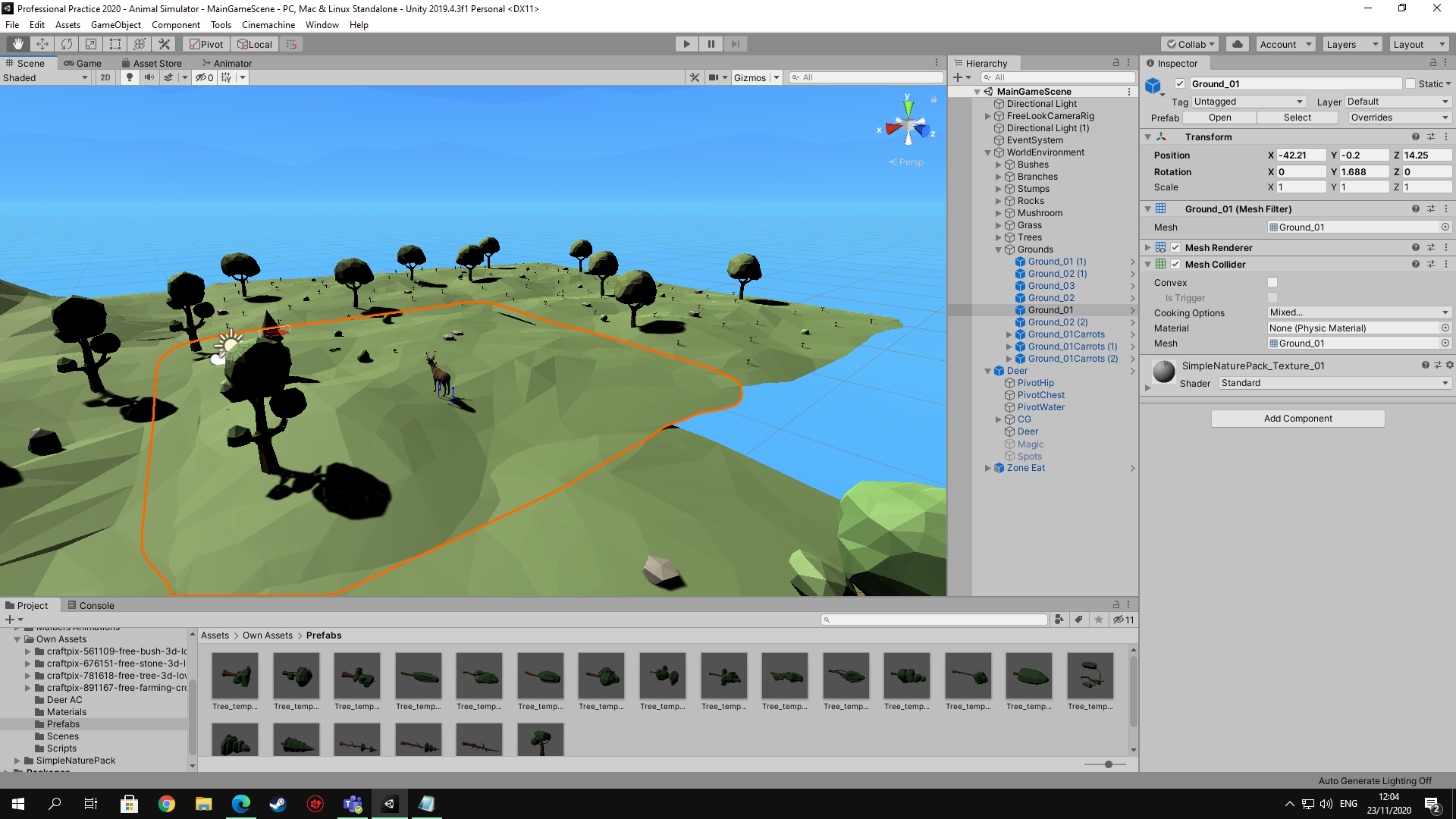


Figure 1 Unity Graphical User Interface

Unity supports a wide range of interfaces, operating systems, and VR hardware. Developers can use Unity in operating systems such as, Windows, OSX and Linux, including mobile operating systems such as iOS, Android, and BlackBerry 10. Unity supports 3 types of application programming interfaces such as DirectX 10, 11 and 12 which allows developers to implement better graphical shaders into their games. It also has VR support for Oculus, HTC Vive, SteamVR, Cardboard, Gear VR and Google Daydream.

Unity has a built-in programming feature called Visual Scripting. The user can click and drag gameplay functions to create games without writing code. However, we won’t be using this for Animal Simulator, the game will be written.

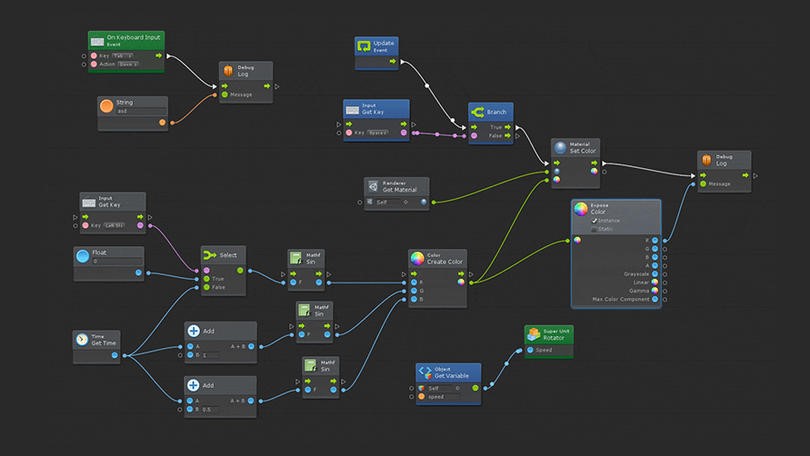


Figure 2 Unity Visual Scripting

Language C# (C-Sharp)

The scripts of Animal Simulator are written in C# because that’s the language Unity understands. C# is a high-level language as programmers can use simpler code or hide relevant properties and functions of an object which reduces complexity and optimises hardware functionality. C# was derived from C and is like C++, it uses the same operators, it’s object-oriented, case sensitive and has identical syntax.

Phaser

Phaser is a free web-framework available from this link: <https://phaser.io/>.

In 2018 we developed a 2D side-scrolling game called Jungle Ice Golem. It was developed using a game framework for HTML5 called Phaser. The game has a menu UI, 2 levels and 2 different maps, the first level is in the jungle and the second level is in a snowy environment.

The main story starts with a heroic Golem that makes his journey into the unknown fighting Trolls and collecting coins. He needs to collect coins and give them to the Troll King as payment to end the war and save the Golem race from extinction.

Phaser focuses on less hardware intensive games and is mainly used for 2D game development for the web. It doesn’t have a GUI workspace or Cinematic view and Visual Scripting, so creating the game world and adjusting assets are more sophisticated.

It uses JavaScript which is a web-based language. JavaScript’s programming paradigm is based on the procedure, whereas C# is more class-based object-oriented programming. The major downside of Phaser is, it is less supported than Unity. When a new version is released certain codes are changed and there is less documentation.



Figure 3 Jungle Ice Golem Main Menu



Figure 4 Jungle Ice Golem's Gameplay Preview

Structure of Unity

Controls

 Hand Tool – this is used to move around the game world and hot key for this is Q

 Move Tool – this is used to move game objects and hot key for this is W

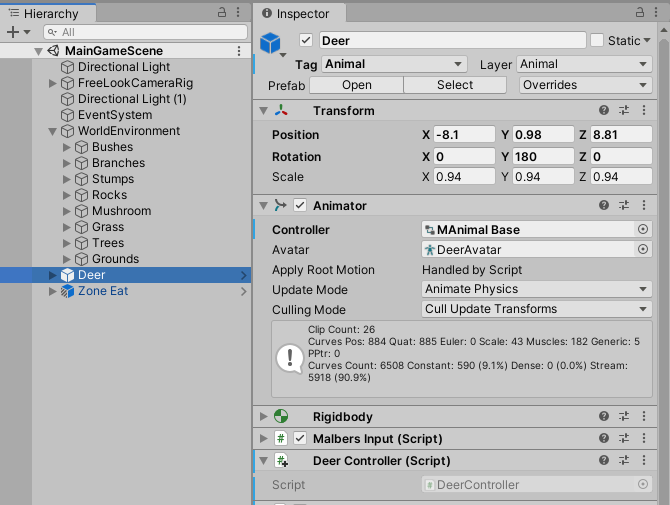
 Rotate Tool – this is used to rotate game objects and hot key for this is E

 Scale Tool – this is used to change the scale of game objects and hot key for this is R

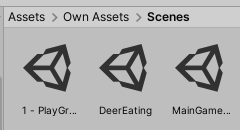
 Rect Tool – this is used to change the length of game objects and hot key for this is T

Hierarchy and Inspector

The Hierarchy contains assets of the game that can be deleted or grouped into folders and configured in the Inspector. The Inspector shows the properties of the game object such as name, tags, size, and colour which can be configured.



Scenes Folder

The Scenes Folder contains scenes of the game. A scene is one part of the game e.g. the Main Menu scene, and this scene will contain the assets, fonts, and scripts. Instead of working in the whole entire game, we can create scenes to divide the game into different parts.

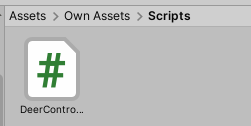
Prefabs Folder

When you add a new game object inside the Hierarchy in Unity and configure it, you can drag and save it inside he Prefabs Folder. These game objects are already configured, and they can be reused or configured again.

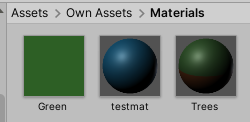
Assets Folder

The Assets Folder is the main folder that contains the scenes, prefabs, scripts, and materials. When an asset is imported from the Unity Store a new Assets Folder is created.

Scripts Folder

The Scripts Folder will contain the scripts files of the game code. Scripts can be added as a component.

Materials Folder

The Materials Folder will contain the materials, textures and colours.

Design Patterns

The world environment of Animal Simulator consists of squares with hills and grass. The fundamental assets that make up a forest such as different plants, trees, rocks, and bushes were placed on the grass land. A folder called Grounds was created in the Hierarchy and all the grasslands were added into that group. In order to override any changes made in the properties, a completed grassland was placed inside the Prefabs folder. This makes it more efficient to expand the world environment.

Should the need to be made larger (as it undoubtedly will, in an open-world game), Unity’s built-in Terrain feature might also be utilised. This feature enables the game developer to create vast expanses of land, onto which they can “paint” features like different textures, trees, hills and smaller details like grass. Neighbouring terrains can also be created, which “link” the swathes of land together to form an even bigger environment.

However, for the style of this game, it was felt that Unity’s inbuilt Terrain feature would create a land that lacks character and individuality. Assets placed on such Terrains tend to look very cut-and-pasted, compared to an environment where everything is strategically placed by hand and eye. (Of course, there is still an element of randomisation to keep the environment interesting to the player.)

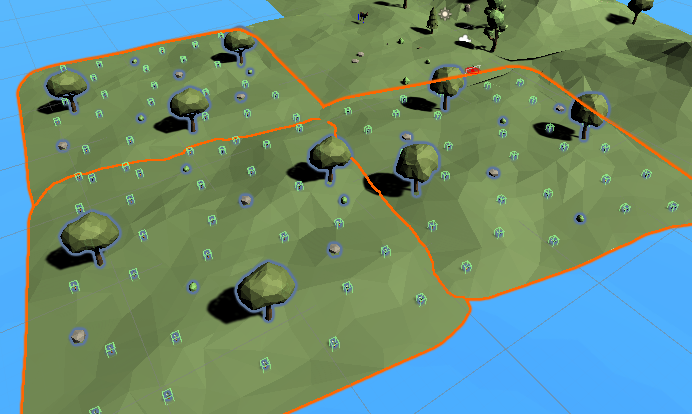


Figure 5 Animal Simulator's World Environment

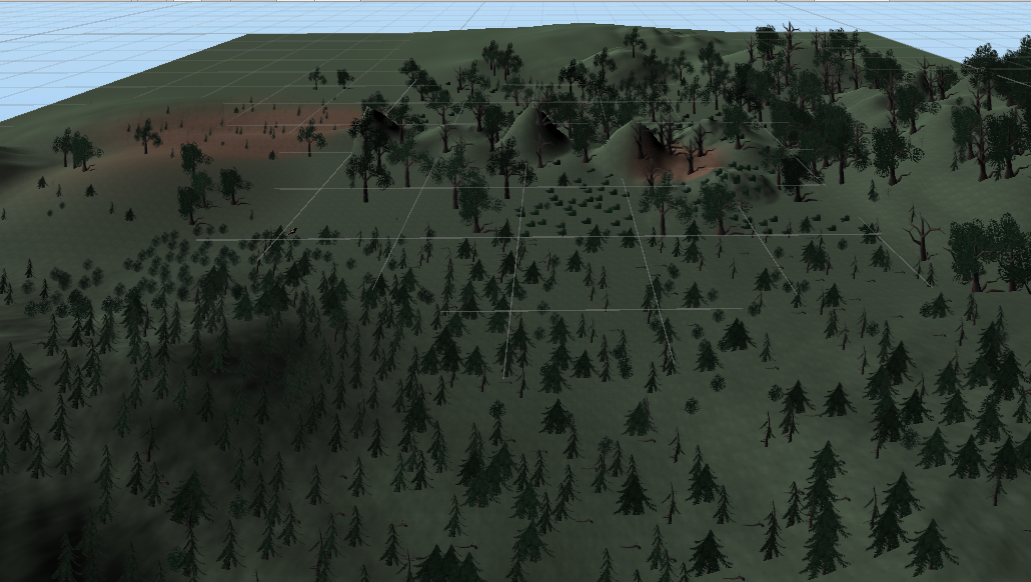


Figure 6 Example of a Terrain (in a different game)

Application architecture (1 page)

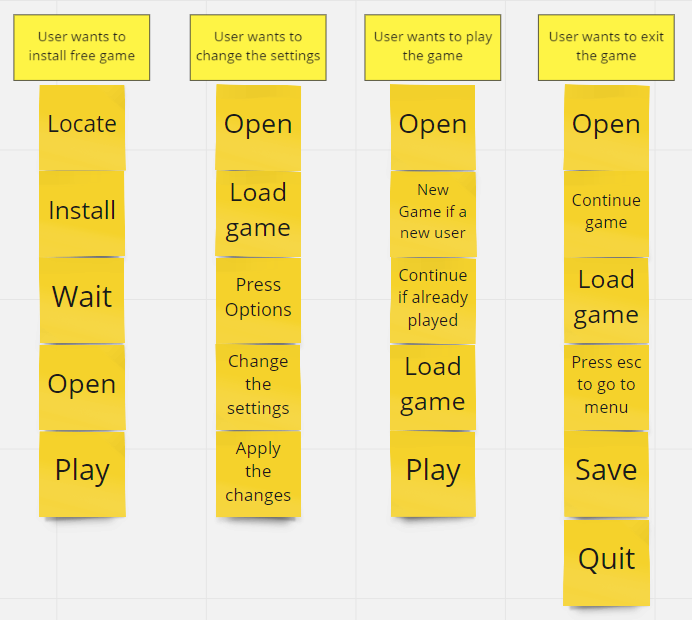


Figure 7 Block Diagram of the Application Architecture

### Database design

Our game will not require a database because it will not be hosted online. It’s an offline game and the save files are stored locally on the player’s computer.

Process design

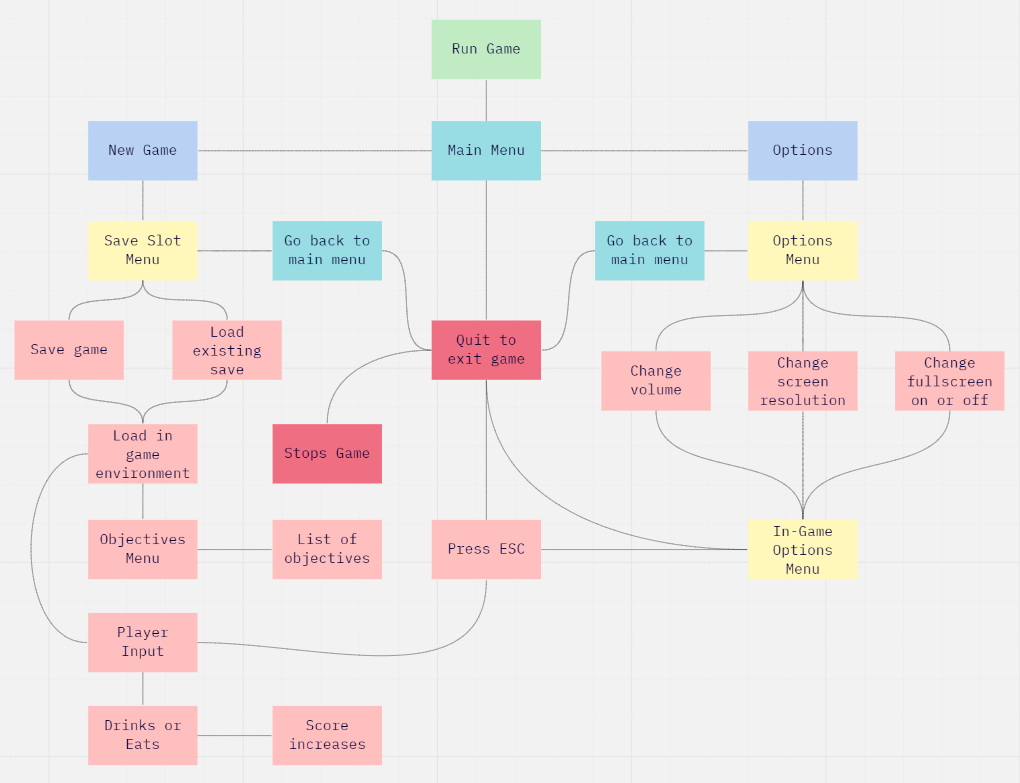


Figure 8 Flow Diagram of Animal Simulator's Process Design

User interface design

This is the in-game user interface of Animal Simulator. The placement and style of the scoring system and the objectives menu were inspired by Goat Simulator’s user interface.

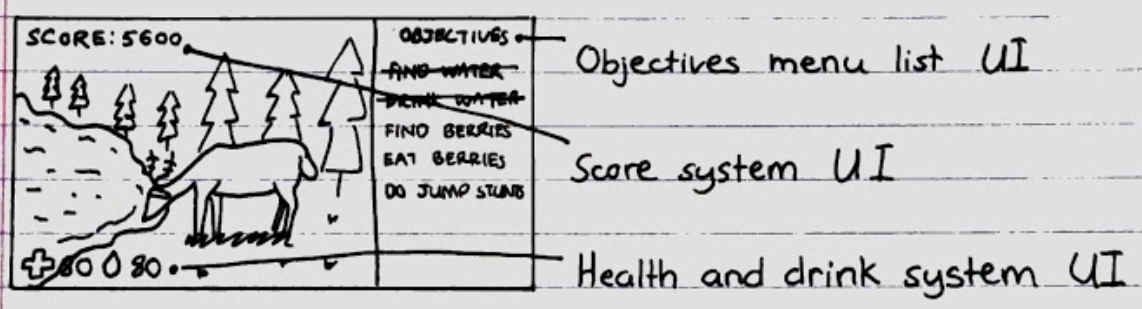


Figure 9 Animal Simulator's User Interface



Figure 10 Goat Simulator's Animal Simulator

This is the Animal Selection Menu of Animal Simulator. It was inspired by *The Isle* and *WildCraft*. Both games allow the user to customise and configure their animal. In those games, you can select different breeds of wild animal, such as wolves or foxes (in the case of *WildCraft*), and dinosaurs (*The Isle*) and configure their properties, e.g. change their coat patterns, add cosmetics, or give them a name.

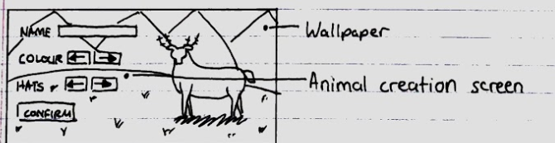


Figure 11 Animal Simulator's Animal Creation UI



Figure 12 The Isle's Dinosaur Creation UI



Figure 13 WildCraft's Animal Creation UI

Wireframe

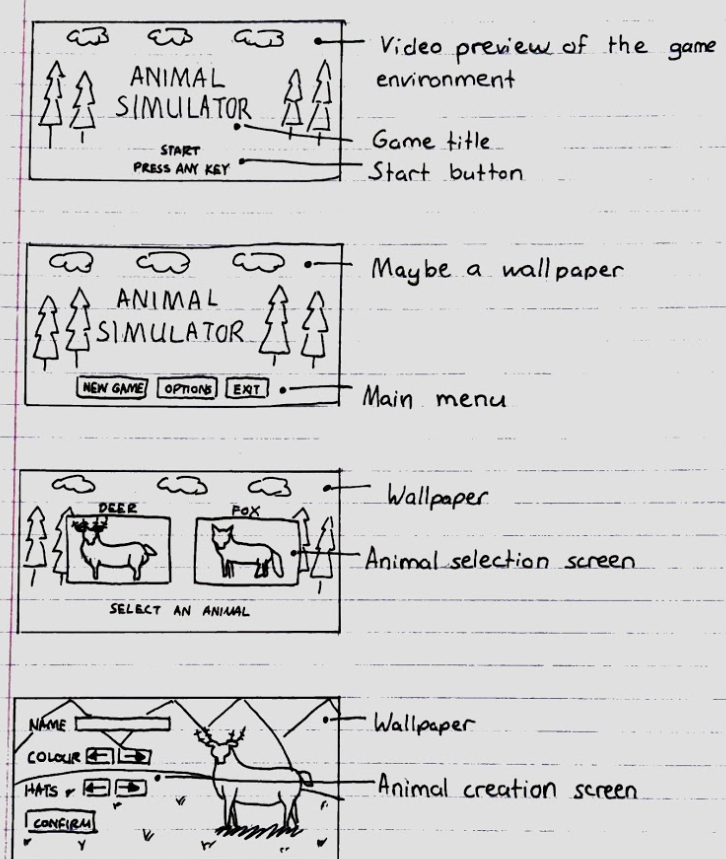


Figure 14 Animal Simulator - Wireframe 1

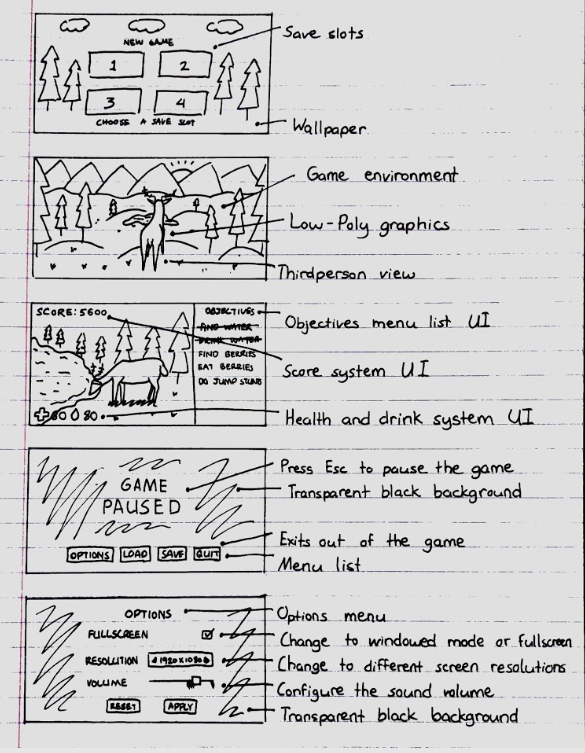


Figure 15 Animal Simulator - Wireframe 2

User Flow Diagram

This flow diagram represents how the user will navigate in the game.

When the user runs the game, they will see the Main Menu which has 3 options: to start a New Game or go to Options or Exit out of the game.

If the user selects Options, the Options Menu will appear and the user will be able to configure the volume, change the display to full screen or windowed mode, and change the resolution. When the user is in the game and presses the Esc key, the game will pause, and the Options Menu will appear in-game.

If the user selects New Game, the Save Menu will appear, and the user can select an Animal. Currently, there is only 1one animal available: a deer. There will also be a button to go back to the main menu. When an animal is selected, the game environment will then load, and the user will be able to play the game. For future development, we may add a Map Menu and add another animal, so the user can choose to play maybe as a carnivore in a snowy environment.

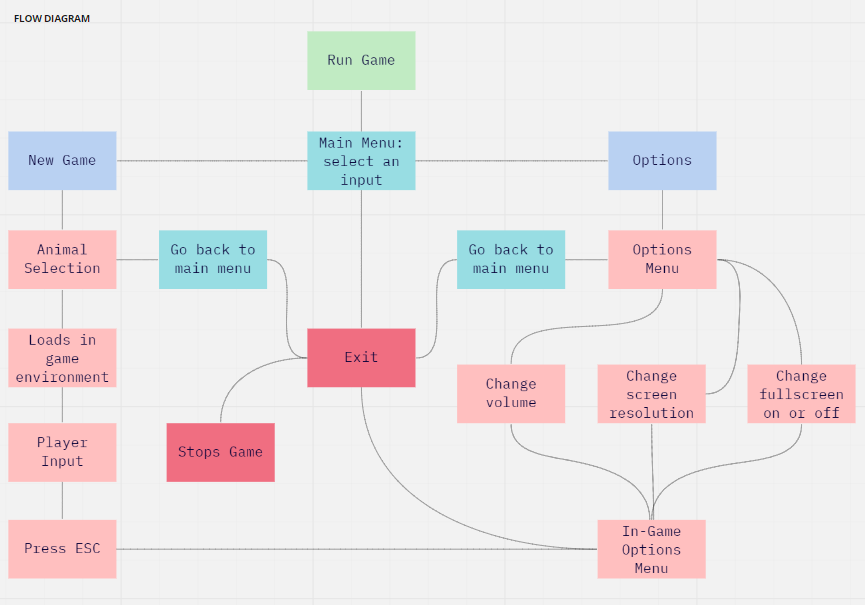


Figure 16 Animal Simulator's User Navigation Flow Diagram

Style guide

While we have not yet implemented a main menu or UI, the plan is first to find a user-friendly font that is not only easy to read, but is eye-catching and complements the theme of the game.

Several potential fonts were:

*Pachyderm* by CybaPee



This font has a round, friendly feeling to it, which might work with the relaxed, casual atmosphere of the game.

*Riffic* by InkyType



Again, this font has a friendly feeling to it, though its slightly less organic structure might not work as well in a game that focuses on being an animal in a natural environment. It is easily readable though which is good for a casual game which may attract players with impairments, eg. Dyslexia.

As for colour scheme, all fonts and UI used in the game will probably be some variant of the earthy, natural green, brown, and grey tones used throughout the game. Perhaps even some “forest” themed UI pack might be utilised (similar to that used in *Jungle Ice Golem,* above).

Storyboard

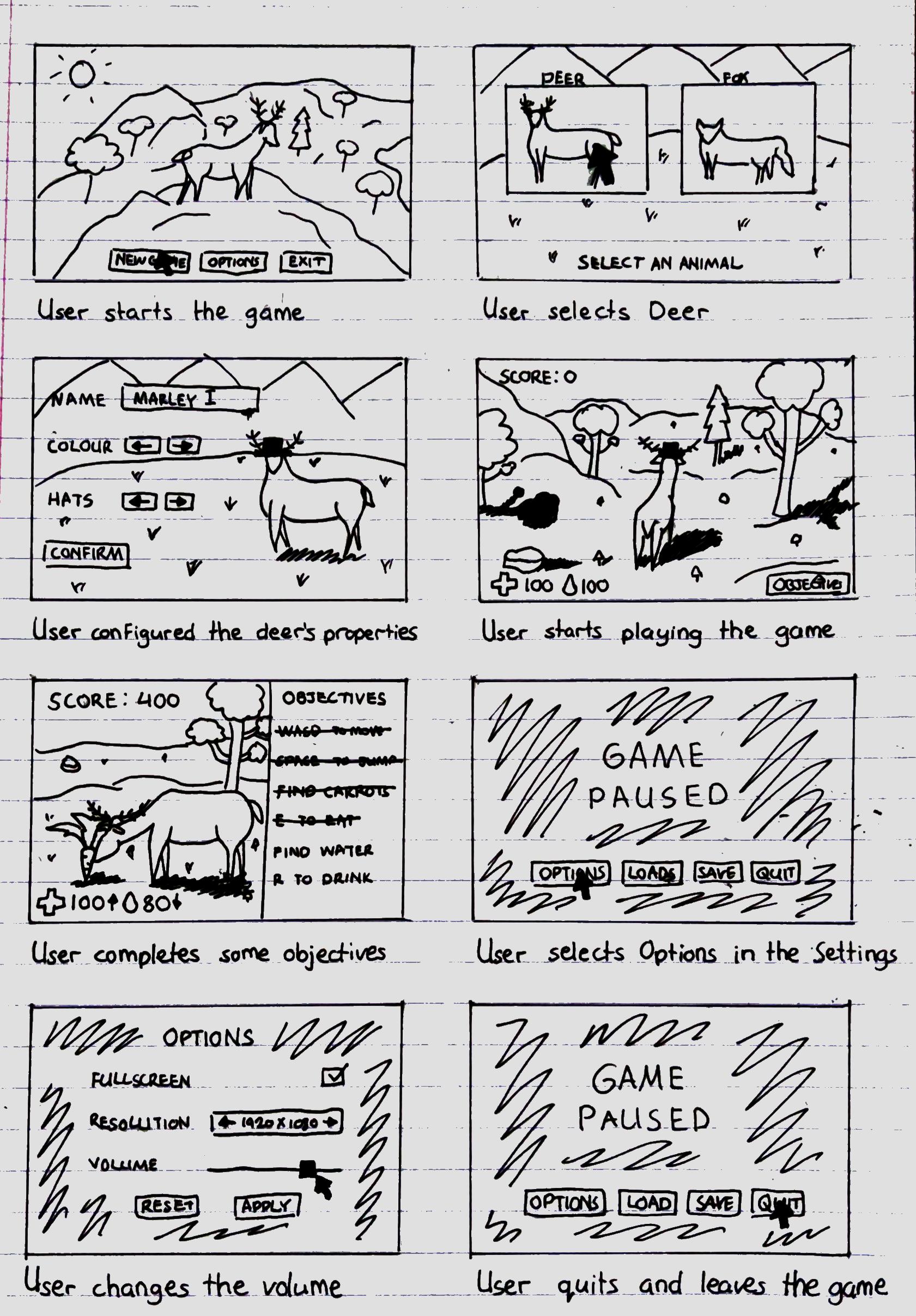


Figure 17 User Storyboard of Animal Simulator

Level Design

The aesthetics of the game hinge on its low-poly artstyle. Each and every detail in the game world has a stylised, almost cartoon-y look to it, from the player’s animal character to the trees and grass of the environment.

Animal Simulator will also not have a linear level design where the player must reach a certain level to equip a weapon or access a certain place. Not many sandbox games have this type of level design.

For example, in *Lego Worlds*, when a player levels up they can unlock more maps and skins, and we may implement this feature into the game. This feature will work well with a score system. If the player reaches a particular score and has completed certain objectives, they can unlock a new map or a new animal. The player of course can continue in the same world if they choose.

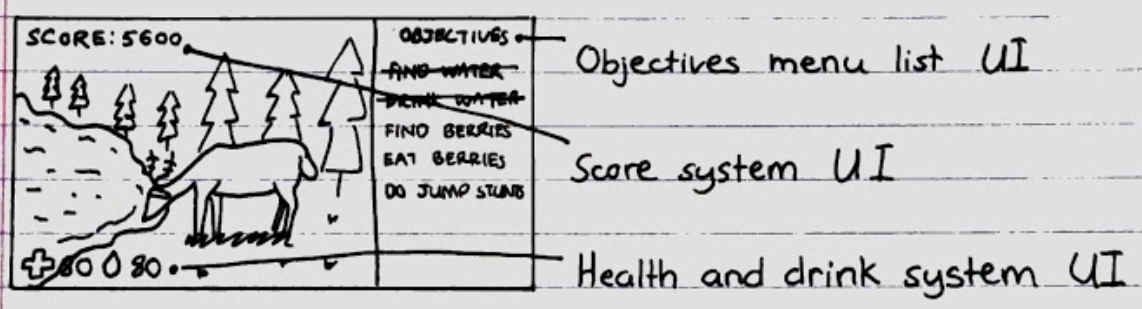


Figure 18 Animal Simulator's in-game user interface

The player can select two different animals, but by default, only one animal is available when they start a new game. When a player unlocks a new animal - in this case, a fox - if they go back to the main menu and start a new game, they can select the fox and play in a different world environment.

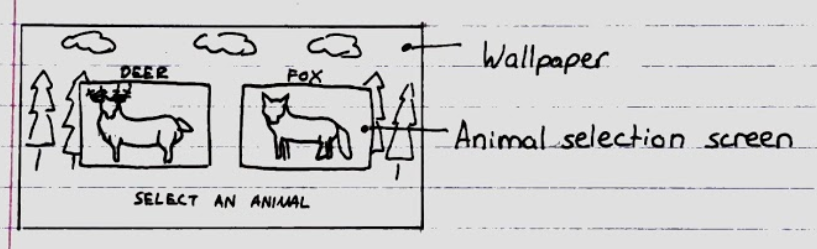


Figure 19 Animal Simulator's Animal Selection Menu

Environment

The game environment is an open-world forest. The forest floor is covered with flat grass and the edge of the world environment will be surrounded with mountains that have collision detection so the player cannot go out of bounds.

Inside the forest environment, there are evergreen trees, small lakes, grass, mushrooms, and rocks. All these assets will also have collision detection.

The player can eat the mushrooms or grass which will increase the score and decrease hunger. The player can also drink water from the lakes which will decrease thirst.

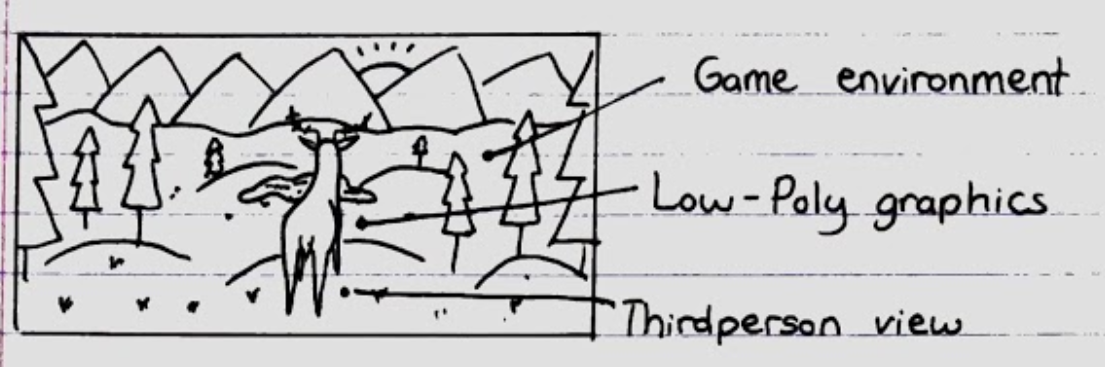


Figure 20 Animal Simulator's game world environment

Conclusion

While it is still a long way from completion, it is hoped that the goals in mind for Animal Simulator will be achieved in an efficient manner during development. Some ideas, such as player customisation, might prove too complex for the limited development timeframe, so they are not final.

From the above diagrams, it should not be difficult for casual players to learn how to play, but it should also not be too easy for more seasoned gamers. We hope that the design choices made during development will encourage players of both types to pick up and play this game.

References

<https://www.visartech.com/blog/unity3d-2020-what-is-new-and-how-business-benefits/>

<https://store.unity.com/#plans-individual>

<https://unity.com/how-to/make-games-without-programming>

https://techterms.com/definition/c\_sharp