horizontal line

3D Mobile Game Project

**Employing the**

**Unity 3D Game Engine**

**Game Design Document**

John Walsh

Table of Contents

[**Overview** 3](#_Toc445030058)

[**Game Concept** 3](#_Toc445030059)

[**Genre** 3](#_Toc445030060)

[**Target Audience** 3](#_Toc445030061)

[**Game Flow Summary** 3](#_Toc445030062)

[**Look and Feel** 4](#_Toc445030063)

[**Gameplay and Mechanics** 4](#_Toc445030064)

[**Gameplay** 4](#_Toc445030065)

[**Game Progression** 4](#_Toc445030066)

[**Mission / Challenge Structure** 4](#_Toc445030067)

[**Puzzle Structure** 4](#_Toc445030068)

[**Objectives** 5](#_Toc445030069)

[**Play Flow** 5](#_Toc445030070)

[**Mechanics** 5](#_Toc445030071)

[**Physics** 5](#_Toc445030072)

[**Movement in the game** 6](#_Toc445030073)

[**Objects** 6](#_Toc445030074)

[**Actions** 6](#_Toc445030075)

[**Combat** 6](#_Toc445030076)

[**Economy** 6](#_Toc445030077)

[**Screen Flow** 6](#_Toc445030078)

[**Game Options** 6](#_Toc445030079)

[**Replaying and Saving** 6](#_Toc445030080)

[**Cheats and Easter Eggs** 7](#_Toc445030081)

[**Story, Setting and Character** 7](#_Toc445030082)

[**Story and Narrative** 7](#_Toc445030083)

[**Game World** 7](#_Toc445030084)

[**Areas** 7](#_Toc445030085)

[**Characters** 7](#_Toc445030086)

[**Levels** 7](#_Toc445030087)

[**Training Level** 8](#_Toc445030088)

[**Interface** 8](#_Toc445030089)

[**Control System** 8](#_Toc445030090)

[**Audio, Music, Sound Effects** 8](#_Toc445030091)

[**Help System** 9](#_Toc445030092)

[**Artificial Intelligence** 9](#_Toc445030093)

[**Opponent and Enemy AI** 9](#_Toc445030094)

[**Non-combat and Friendly Characters** 9](#_Toc445030095)

[**Support AI** 9](#_Toc445030096)

[**Technical** 9](#_Toc445030097)

[**Target Hardware** 9](#_Toc445030098)

[**Development Hardware and Software** 10](#_Toc445030099)

[**Network Requirements** 10](#_Toc445030100)

[**Game Art** 10](#_Toc445030101)

[**Key assets** 10](#_Toc445030102)

[**Goals** 10](#_Toc445030103)

[**Specifications** 10](#_Toc445030104)

[**Milestones** 11](#_Toc445030105)

## **Overview**

### **Game Concept**

The original game concept was designed to be played on a touchscreen mobile devices, which means screen size will vary immensely. It’s unclear at this stage if other platforms like PCs & consoles will require support in the future. The gameplay primarily evolves around herding cattle on your farm, increasing your cash reserve and generally improve farm over time. To achieve this the player must buy and sell cattle at the local mart. Supplies will also be required to improve the wellbeing of the cattle on the farm.

**Future Development**

I believe the game has potential to grow and develop into more than just another farm simulator game. From an education point of view the game could teach kids more about farm animals and how to properly take care them. Helpful tips and information could be displayed on screen on occasion to progressively teach the player about each animal as they play.

List of other ideas:

* Gameplay features like private sales between the player and NPC's in-game could be an aspect to implement later in development, normal these private sales between farmers take place occasionally.
* Multiplayer support may be included later in the project, which means likely that player chat, buying and selling between players could be features implemented afterwards.

Overall the game should be quite basic at first with more features being added to the gameplay over time. To cap off the concept, the game must have a cartoony feel to the game to make it feel welcoming to all ages.

### **Genre**

Cattle Farm Simulator, Farming, Animal Farming

### **Target Audience**

The target audience could roughly range from 6 + or older, I'm not entirely certain what age group this game fits into.

### **Game Flow Summary**

The world will be constructed in 3D open environment, which means the player must be able to freely navigate their way around the world. A virtual joystick will implemented to allow players with touchscreen devices control the character.

Transitions between levels takes place when the player’s character comes into contact with a special object / area in-game that triggers an event to display a menu. This menu allows the player to transition from either the farm scene to the mart scene or back to the main menu.

### **Look and Feel**

The visual styling of the game will be important because not all device will be able to run high resolution textures and high ploy count objects within the game world. I believe by keeping the resolution fairly low will increase the amount of devices that would be capable of running the game. A cartoony feel to the game will look perfect on most devices.

## **Gameplay and Mechanics**

### **Gameplay**

Currently in (Version 0.6) the gameplay is still quite basic. The player starts with two animals which he / she must look after & develop the animal's stats, health and happiness. Cattle feed & supplies can be bought from the NPC at the stall in the mart.

Over time I've plans for developing the gameplay into more comprehensive experience, with more interactive activities with the environment, cattle and other NPC's in-game.

### **Game Progression**

The original design draft of the game contained no game progression, since the implementation stage allows me to experiment with different ideas to improve the gameplay we've added features like cattle feeding and interactive monitoring with the animals. Players can tap on the cattle to view individual stats on each animal, the health of the animal & happiness too. Eventually the addition of veterinarian & vaccinations could be an interesting feature to add later.

### **Mission / Challenge Structure**

The mission / challenge structure is unclear at this current time. There isn't mission structure per se, the primary goal is to simulate farming cattle, to look after and grow the herd as large as the player can achieve.

Future development of the gameplay would expand this area considerably as challenges like herding your cattle and developing your farm into a more profitable enterprise would be the main area of focus for the game.

### **Puzzle Structure**

Currently the game contains no puzzles to complete to progress through the game

### **Objectives**

What are the objectives of the game?

### **Play Flow**

How does the game flow for the game player

### **Mechanics**

'What are the rules to the game, both implicit and explicit? This is the model of the universe that the game works under. Think of it as a simulation of a world, how do all the pieces interact? This actually can be a very large section.'

The game mechanics is largely tied to the unity 3D engine. Interactions between objects and the game world & player are all handled by the in-built physics engine. Each item / asset contains unity components that attach to each game object. If for example the player needs high detailed mesh collider, unity has pre-made a component for this task that takes the rendered model of the player object and creates a high detailed mesh collider that interact with the 3D world.

Many game objects we employ use pre-made components from unity, it allows me to quickly prototype ideas we need to implement, test & deploy to the final game scene.

### **Physics**

The game physics in this game world employs the built-in unity physics engine. Collisions between objects and the player are all handled by components attached to the game objects in the game scene. The same is true for the game animals and the game world. The terrain in the game scene must response to collisions between game objects & players to create the feeling that it can be walked upon and explored by the players and used by world game objects.

Unity's terrain engine is quite powerful tool to the developer. The terrain in every game scene contains a layer that allows game objects like trees, farm structures and player's character to detect the height and depth of the terrain. This layer allows the physics engine reduce the amount of CPU cycles required, but also allows the developer to create complex environments.

A number of physics problems still exist in the game as of (Version 0.6), which needs addressing fairly soon. The animals while in motion can glitch slightly if the player's character's physics collider bumps into the mesh renderer of an animal, it could send the animal flying. Which sounds ridiculous really and needs to be fixed ASAP.

### **Movement in the game**

Character movement in the game will be handled by a virtual joystick situated on the left side of the player's screen. It's common to find this type of control input in games that require the full movement of the player's character.

### **Objects**

At the current stage of development (Version 0.6), in-game objects can not currently be moved by players. However the idea of allowing the player move objects around the farm is an interesting challenge. It would require each object's position in the scene to be tracked and recorded. Implementing the feature would require some sort of data storage, possibly serializing data to file.

### **Actions**

Including whatever switches and buttons are used, interacting with objects, and what means of communication are used

### **Combat**

This game genre does require any in-game combat, however the idea of maybe including a feature that requires male animals to fight over mating rights could be an interesting idea to implement!.

### **Economy**

The overall economy in-game requires players to slowly build up their animal herd to sell at the mart to hopefully make a profit.

### **Screen Flow**

A graphical description of how each screen is related to every other and a description of the purpose of each screen.

### **Game Options**

What are the options and how do they affect game play and mechanics?

### **Replaying and Saving**

Gameplay saving is handled by a script that serializes player object data into specific files. Every time the player transitions from one scene to another the object data from the player and farm animals are serialized to file. Once the new game scene is pre-preparing to load, the game data is de-serialized back into objects to be used by the loading scene.

### **Cheats and Easter Eggs**

No cheat code or Easter eggs will be implemented into the game until a couple of versions are released.

## **Story, Setting and Character**

### **Story and Narrative**

Includes back story, plot elements, game progression, and cut scenes. Cut scenes descriptions include the actors, the setting, and the storyboard or script.

### **Game World**

General look and feel of world

### **Areas**

Including the general description and physical characteristics as well as how it relates to the rest of the world (what levels use it, how it connects to other areas)

### **Characters**

Each character should include the back story, personality, appearance, animations, abilities, relevance to the story and relationship to other characters

The main character is quite basic at the current time as of writing (Version 0.6), and will require further development. I hope to add player customization into the character creation stage. It would greatly improve the gamification aspect of the entire game I believe.

The appearance of the player character is quite cartoony, and should require little resources to render on screen in real-time. The polygon count should be restricted to certain level to allow as many devices to run the character animations without any lag problems. Overall the theme of the game is quite cartoony so this fit perfectly for the requirements.

## **Levels**

Levels. Each level should include a synopsis, the required introductory material (and how it is provided), the objectives, and the details of what happens in the level. Depending on the game, this may include the physical description of the map, the critical path that the player needs to take, and what encounters are important or incidental.

### **Training Level**

There is currently no requirement for training levels. The game content does not really require players to first complete a training level. During development however I used test levels to debug certain aspects of the game’s source code.

### **Interface**

Visual System. If you have a HUD, what is on it? What menus are you displaying? What is the camera model?

The visual interface or GUI has been partially built using [NGUI](http://www.tasharen.com/?page_id=140) which contains some powerful features that allow developers build scalable interfaces for different screen sizes and device whether it be a tablet or smartphone. This was a major problem to overcome during the designing phase.

**Main Menu**

The main menu is a vital part of any game, both in a visual and function sense. It must contain the vital paths the player needs to navigate through the game. From this menu the player must be able to access the settings / options menu, the new game menu and the load game option.

**Options / Settings Menu**

The options menu contains the sound controls and difficulty rating for the game. The difficulty rating can be changed and applied to a current game save if the player is finding it difficult to get their farm to turn a profit.

**New Game / Player Name Menu**

As the title states this menu simply gets the player name and allows you start a new game. This menu should also allow the player to get back to main menu by pressing the back button.

### **Control System**

The game UI allows the player to issue commands to various parts of the game. The current version of the game (0.6) requires much more development to improve the gameplay.

### **Audio, Music, Sound Effects**

The audio management is handled by Unity’s sound engine. Unity has components like the [Audio Listener](http://docs.unity3d.com/Manual/class-AudioListener.html) and [Audio Source](http://docs.unity3d.com/Manual/class-AudioSource.html) that can be attached to game objects. These special components allow scripts attached to game object to call these components when need to keep the source code de-coupled and maintainable. I’ve used both components along with the [Audio Clip](http://docs.unity3d.com/Manual/class-AudioClip.html) component to construct sound loops to be played in game menus. I’ve also used these components to attach sound effects to in-game scenes, whether it be animals, the player character or environment sound effects.

I’ve sourced a lot of sound effects from various places, all royalty free of course. To name a few sources, [SoundCloud](https://soundcloud.com/), [FreeSound](https://www.freesound.org/) and other various Unity assets on the Unity [Assets Store](https://www.assetstore.unity3d.com/en/).

### **Help System**

The game did include a basic menu explaining the rules & guidelines. However the menu needs more work so I removed it from the current version (0.6) for the moment.

### **Artificial Intelligence**

The game contains a limited amount of AI to allow for instance, an animal to travel to a particular area in the game scene. This script fires the animation engine to kick to play the correct animal animations require to make it look like it’s in motion. If the animal fails to find a route to the specific location it re-traces its step to find another route.

### **Opponent and Enemy AI**

The active opponent that plays against the game player and therefore requires strategic decision making

### **Non-combat and Friendly Characters**

There is currently one NPC in-game in the mart scene which allows you to buy more supplies required to keep your farm running.

### **Support AI**

Player and Collision Detection, Pathfinding

### **Technical**

Text Here

### **Target Hardware**

The specific hardware this game is targeting is quite diverse. While the Unity game engine allows you to port too many different platforms like PlayStation & Xbox, I’ve been focusing my attention on the Android platform. The Android platform can present game development with a couple of problems, one would be the API level. Over the years the different versions of Android all the way from 1.5 to 6.0 presents one major problem to the developer. What API level do I develop for? Unity thankfully handles this problem by allowing the developer to choose the lowest API level required to run the game. The list of APIs of Android [here](https://en.wikipedia.org/wiki/Android_version_history).

### **Development Hardware and Software**

**Hardware**

The development hardware consisted of my own personal collection of Android devices, in all different screen sizes and Android versions. It’s quite simple to enable debugging on smartphones & tablets to run the compiled APK file to install the game. Before I can build project & compile the source into APK file, I first must set or create the “keystore” file. Read more about the file [here](https://en.wikipedia.org/wiki/Keystore). It’s basically signing certificate that must be included with project when submitting applications to the Google play store. Also any incremental updates that proceed must also include this special file when building the project before submitting to the store.

**Software**

Unity employs C# as its main programming language, it’s quite similar to Java in respect, which is my main programming background. The C# programming language is built upon the .NET framework, with powerful libraries at its core, I can see why Unity’s developers choose to use Microsoft’s language to develop games.

Unity has some advantages & disadvantages over other game engines, below I’ll list some of the reason why I choose to use this SDK to develop this game.

**Advantages**

* Platform support, lots of different platforms available. Android, iOS, Xbox, Playstation
* Strong libraries, uses the C# programming language, great support and documentation

**Disadvantages**

* No game templates or gameplay frameworks to build upon and develop
* Everything from scripts and assets need to be built from scratch, unless you already have assets built from games before or sourced assets from the store
* Visuals are not the best compared to [UDK](https://www.unrealengine.com/what-is-unreal-engine-4) or [CryEngine](http://cryengine.com/). This isn’t much of a problem as the game artwork has been determined to have a cartoony look and feel.

**Debugging**

Unity’s IDE Mono-Develop supports the break line feature even when the game running, which is quite powerful when dealing with elusive problems with scripts.

### **Network Requirements**

The network requirements as of (Version 0.6) do not require the game to consistent internet connection. However I do plan on releasing a version soon that will require the player to login to the Google play service backend using a Gmail account. This service that Google provides the tools required for developers to build real-time multiplayer games. Other features like leader boards, player stats and achievements can all be implemented by using this service.

### **Game Art**

The artwork will be based around cartoony type textures, to keep the amount of memory and processing power required to run the game to a minimum. This will allow the game to be played on much older devices then the latest and greatest smartphone and tablets available today.

Some resources / assets like the animal models and texture have been taken from the Unity assets store. Other assets like housing structures and barns have been constructed using the application called [Blender](https://www.blender.org/). This application allows game models to be constructed and mapped to textures in no time at all.

### **Key assets**

How they are being developed. Intended style.

## **Goals**

The goals of this project were to develop a 3D mobile application that could target multiple platforms. The general theme of the game was a “Cattle Mart” game that would allow players to buy & selling cattle.

## **Specifications**

The specifications for this project were originally quite large for two people to develop. Several meetings took place to gather the requirements, focusing on the bare functionality of the game.

## **Milestones**

TODO

* GitHub commits write up
* Review the progress made over time
* Code samples included in this section
* Documentation of the code
* Etc.