# Design and development of a farming/life simulator game

## Mythrest city, a lifelong journey

UNIVERSITAT JAUME·I

Author: Helena Or Rubio
Tutor: Raúl Montoliu Colás

# Abstract

The following document embodies the process of the development of the game Mythrest city. The videogame is a bachelor's thesis of the degree in Videogame Design and Development. The game is a farming/life simulator, the main goal of the player is to grow crops and sell them to earn money. In addition, the player can be friends with the villagers by giving them gifts and talking to them every day. The project is inspired by the Harvest Moon and Story of Seasons sagas [1]. This game will be played on a computer and has 3D graphics.

# Key words

Videogame, life simulator, farming simulator.

# INDEX

# LIST OF FIGURES

# LIST OF TABLES

# 1. Technical proposal

This chapter presents a first glance of the project. Here an introduction to the story and motivations of the game are provided as well as the projects scope and main tools to develop it.

---

## 1.1. Introduction

Not all videogame genres had been developed as much as they could, for example farming simulators. Due to my personal experience in those kind of games I know that the market is very reduced, there are few videogames in this genre, and if a look is taken at the computer games there are even less games (Harvest Moon:Light of Hope [2] and Stardew Valley [3]). So one of the main motivations of this bachelor's thesis is to do a farming simulator for computer due to its reduced market.

In this bachelor's thesis the design and development of "Mythrest city, a lifelong journey" will be made. The game takes place in a small village named Mythrest. The main objectives of the player are to farm crops, take care of animals and build a good friendship with the villagers. In this game the player will have two or three different kind of crops each season, only one type of animals (a chicken), four villagers and some wild items (colored grass which will help the player to recover it stamina for example).

The player will be able to plant crops, water them and see how they are growing while the days pass, and when they are ripe the player will be able to recollect and sell them. With the money earned, the player can buy more seeds or fertilizer so our crops grow with a better quality. Moreover, the player will have a chicken, which will lay an egg once a day. If a good friendship is made with our chicken, the laid eggs will have a better quality.

The game will need a time manager so days, months and years pass in the game space. Each day will last 18 hours, an in game hour will be one minute in real life, the player will wake up at 6:00, and he will be automatically sent to sleep at 23:59 if he did not go to sleep on his own. A database will be created to store the different kinds of crops, for the inventory of the player.

A demo of the game will be made due to time restrictions. The elements which will be included are: 4 NPCs, 2 kind of crops each seasons, one tree each season, one chicken to represent the animals.

In relation to the art, it will be Animal Crossing [6] like. The 3D models will be low poly with vivid colors. The human models will be bigheaded[7] (animal crossing model example) and what will give hints to the player about the physical appearance would be the

2D art of the villager (example). The advantages of this kind of art and modelling is that the amount of time used for animations is greatly reduced, although the time used for 2D art is slightly increased. The final result is really interesting when using this kind of representations.

## 1.2. Related subjects

The main subjects regarding this project can be:

VJ1216 → 3D Design:

The game has several 3D models, approximately 85. All the objects were designed and modeled by hand.

VJ1223 → Videogame art:

The project has several art items, like the NPCs portraits, the interface...

VJ1227 → Videogame engines:

The project has been built in Unity3D and that was the tool we used on this subject.

VJ1231 → Artificial intelligence:

In the project there are NPCs who have a preestablished route, and to create it we need a little AI.

## 1.3. Objectives

The main objectives to accomplish on the development of the project are:

- ➔ Create 2D and 3D art appropriate for the game genre(farming/life simulator).
- ➔ Make a playable demo which faithfully represents the final game.
- ➔ Develop a game which provides a fun and dynamic game experience.

## 1.4. Planning

The planning of the game will be further explained on 4th section.

1. **2D Art/3D Modeling and animation (65h)**
   - 1.1. 2D Art(20h)
     - 1.1.1. NPC's portraits(10h)
     - 1.1.2. Object portraits(6h)
     - 1.1.3. Interfaces(4h)
   - 1.2. 3D Modeling and animation(45h)
     - 1.2.1. Sceneries(5h)
     - 1.2.2. PJs models(10h)
     - 1.2.3. PJs animations(20h)
     - 1.2.4. Objects models(10h)

**2. Programming(185h)**

- 2.1. World related(37h)
  - 2.1.1. Adding sceneries and hitboxes(2h)
  - 2.1.2. Map floor positions into a matrix(20h)
  - 2.1.3. Time manager(15h)
- 2.2. Player related(48h)
  - 2.2.1. Unity Animators(3h)
  - 2.2.2. Player movement and actions(40h)
    - 2.2.2.1. Walk(1h)
    - 2.2.2.2. Water(9h)
    - 2.2.2.3. Crop(9h)
    - 2.2.2.4. Plow(9h)
    - 2.2.2.5. Idle(2h)
    - 2.2.2.6. Other in-game actions/interactions(10h)
  - 2.2.3. Object quality system(5h)
- 2.3. NPC related(45h)
  - 2.3.1. Create dialogs and talking system(5h)
  - 2.3.2. Route creation(20h)
  - 2.3.3. Friendship system(10h)
  - 2.3.4. Shop creation(10h)
- 2.4. Database related(25h)
  - 2.4.1. Database creation(10h)
  - 2.4.2. Game saver(5h)
  - 2.4.3. Inventory system(10h)
- 2.5. Testing and error solving(30h)

**3. Documentation(50h)**

- 3.1. Technical proposal(6h)
- 3.2. GDD(10h)
- 3.3. Final memory(25h)
- 3.4. Presentation
  - 3.4.1. Planning(8h)
  - 3.4.2. Presenting(1h)

## 1.5. Expected results

The expected results of this bachelor's thesis are:

- ➔ A videogame where the player has a good time.
- ➔ The chance to grow the game bigger with the time.
- ➔ The possibility to add another farming/life simulator to the market with the time.

## 1.6. Tools

1. Unity 3D
2. Blender[4]
3. Adobe Photoshop
4. Inkscape[5]

# 2. Game design document

The following section describes the game design document of the game.

**TITLE:**

Mythrest city, a lifelong journey

**SUMMARY:**

Videogame of the genre farming/life simulator in which the player embodies a farmer making a fresh start. The player needs to enlarge its farm as much as possible and make the farm thrive at the same the player needs to improve its relationship with the villagers.

---

## 2.1. Section 1 - Videogame Summary

### 2.1.1. Game Concept

In Mythrest city the player incarnates a farmer starting up in the farm life. The main character, tired of the inconveniences and problems of the modern life, decides to buy a small farm and to take care of it. The player can grow its own crops to sell them and earn money to grow its farm bigger or to buy animals. The relationship with the villagers can be improved (or not).

### 2.1.2. Characteristics

Mythrest city will be developed to be played in a computer and its genre is farming/life simulator. As main characteristics of the game it can be distinguished:

I. Although the videogame has a story, that is not 100% relevant. The player can create its own schedule to do its farm chores, decide if he wants to be friends with the villagers, decide if he want to keep animals...etc

II. The art will be simple. Art is important in videogames of this genre, but most of the time they are faithful to a simple style, reinforced by 2D graphics shown in the dialogs or the object descriptions.

The game can be compared to other games like the Harvest Moon [1] or Story of Seasons [2] sagas. In addition, it can be compared to Stardew Valley [3] , an indie game which acquired big success.

#### 2.1.2.1. Genre

The game Mythrest city mixes up three genres:

I. Life simulator: Genre in which the player controls one or more individual AI units.
II. Farming simulator: Subgenre of the building and management genre. The player needs to control a virtual farm in which he can take care of crops and animals.
III. Role-playing: The player controls a character in a world with well-defined rules.

#### 2.1.2.2. Target Audience

Based on the aforementioned references (Harvest Moon, Story of Seasons and Stardew Valley) the target audience decided are children over seven years old. This is because the game there may be references to alcoholic drinks and comic mischiefs. The game does not only target children, it does not have age limit.

### 2.1.3. Visual Style

*Illustration 1: Videogame logo*

The game's visual style will be simple and colorful, a style liked by young and old people. The art is distinguished in two sections:

I. 3D art: In relation to the models, they will have a low poly and colorful style. They won't be neither realistic or detailed.
II. 2D art: It will have a japanese comic style mixed up with a ocidental drawing style.

### 2.1.4. Scope of the Project

Due to the 300 hours time limitation that this project has, in the distinct functionalities of the videogame a small amount of elements will be developed:

- 4 NPCs
- 2 seasonal crops
- 1 seasonal fruit tree
- 1 farmer gender

## 2.2. Section 2 - Gameplay and Game Mechanics

### 2.2.1. Gameplay

The gameplay of Mythrest city is very open, the player does not need to do anything mandatorily, but its actions imply some consequences. As an example, if the player decides that he does not want to grow any crops he will not have money to buy animals or objects liked by the villagers so he won't be able to be friends with them.

#### 2.2.1.1. Objectives

The main objective is to make your farm successful, in order to achieve that, the player must grow crops and sell them when they are fully grown. With that money the player can buy more seeds and other in game objects or build/upgrade his farm (ej: poultry house).

### 2.2.2. Mechanics

#### 2.2.2.1. Player Mechanics

The player puts himself in the shoes of the main character, a farmer who wants his farm to flourish. The gameplay of the game resides in some main tools that the player has at his fingertips since the start:

- Watering can: Allows the player to water his crops.
- Sickle: Allows the player to cut weeds or dry crops.
- Hoe: Allows the player to plow the soil so he can plant crops.
- Seeds: If we plant them in the plowed soil a crop will grow in some time.

The use of this tools will make the player lose stamina, if the stamina reaches zero the player will pass out and wake up the next morning having recovered only a half of its stamina.

**Objects**

The player has at disposal different objects with which he can interact:

- Wild plants:
    - Weed: Without value, it will uglify the zones where weed and other plants grow.
    - Edible wild fruits:
        - Plum: Grows in spring.
        - Apricot: Grows in summer.
        - Blackberry: Grows in fall.
        - Goji berry: Grows in winter.
- Crops:

*Table 1: Seasonal crops and trees.*

| Season | Crop | | Tree |
|---|---|---|---|
| Spring | Turnip | Cabbage | Cherry tree |
| Summer | Onion | Radish | Peach tree |
| Fall | Carrot | Spinach | Apple tree |
| Winter | Daikon radish | Broccoli | Oranje tree |

- Buyable objects:
    - Vegetable smoothie
    - Fruit smoothie
    - Wine
    - Bracelet
    - Book
    - Photo
    - Fertilizer
- Tools:
    - Watering can
    - Sickle
    - Hoe

**Actions**

The playable character is a farmer who wants his farm to flourish. This farmer can do the following actions:

- Main actions:
    - The player can walk through the walkable zones.
    - Plow: If the player has equipped the hoe, is located on a fertile zone and has enough stamina, he can plow the soil.
    - Water: If the player has equipped the watering can, is located in front of the plowed soil and has enough stamina, he can water the piece of ground (even if there are no seeds planted).
    - Plant: If the player has equipped the seeds, is located in front of plowed soil and has enough stamina, he will be able to plant seeds.
    - Cut weeds/crops: If the player has equipped the sickle, is located in front of a crop or a wild plant and has enough stamina, he can remove it with the sickle.
    - Harvest: If the player is located in front of a collectible object without tools equipped, he will be able to recollect the object pressing the adequated key.

- Other actions:
    - Talk to the villagers: The player can talk with the villagers positioning himself in front of them and pressing the adequate key.
    - Give gifts to the villagers: The player can give gifts to the villagers positioning himself in front of them and with what he is going to give them equipped press the adequate key.
    - Grab animals: The player can grab his small animals (chickens) to improve his relationship with them. Improving the relationship with animals will make them happy and hence its animal products will have a better quality.
    - Buy: The player can obtain different objects.
    - Eat: The player can buy some edible objects or recollect wild fruits and eat them to recover stamina.

**Missions/Challenges**

The game does not have a concrete mission or challenge but the player has some things to do:

- Harvest all kinds of crops.
- Sell crops and earn money.
- Be friends with the villagers.
- Own animals.
- Be friends with the animals.

#### 2.2.2.2. Machine Mechanics

The NPCs have the following machine mechanics:

- Each NPC has a default route determined by the day of the week and the weather.
- The player can give gifts to the NPCs, if they like it or not you will earn or lose friendship points with them.

### 2.2.3. Economy

The economy system is simple, to sell the objects that the player has been recollecting he must put them in the shipping box and they will be sold automatically. There is also a shopping system, the player can buy distinct kinds of things. Below is shown a table with the different kind of objects and its trading prices:

*Table 2: In-game items and its trading prices*

| The name of the in game money is Glod, and the abbreviation is g. | | |
|---|---|---|
| Object | Buy | Sell |
| Turnip seeds | 150g | 150g |
| Cabbage seeds | 320g | 320g |
| Onion seeds | 160g | 160g |
| Radish seeds | 220g | 220g |
| Carrot seeds | 140g | 140g |
| Spinach seeds | 280g | 280g |
| Daikon radish seeds | 220g | 220g |
| Broccoli seeds | 300g | 300g |
| Cherry tree seeds | 1700g | 1700g |
| Peach tree seeds | 1950g | 1950g |
| Apple tree seeds | 2000g | 2000g |
| Orange tree seeds | 2100g | 2100g |
| Turnip | - | 50g |
| Cabbage | - | 106g |

| | | |
|---|---|---|
| Onion | - | 53g |
| Radish | - | 73g |
| Carrot | - | 46g |
| Spinach | - | 93g |
| Daikon radish | - | 73g |
| Broccoli | - | 100g |
| Cherry | - | 425g |
| Peach | - | 487g |
| Apple | - | 500g |
| Oranje | - | 525g |
| Plum | - | 21g |
| Apricot | - | 18g |
| Blackberry | - | 15g |
| Goji berry | - | 24g |
| Fertilizer | 100g | 100g |
| Vegetable smoothie | 3000g | 3000g |
| Fruit smoothie | 2700g | 2700g |
| Wine | 250g | 250g |
| Bracelet | 1200g | 1200g |
| Book | 570g | 570g |
| Photo | 210g | 210g |
| Weed | - | 1g |
| Branch | 50g | 1g |

## 2.2.4. Screen Flow

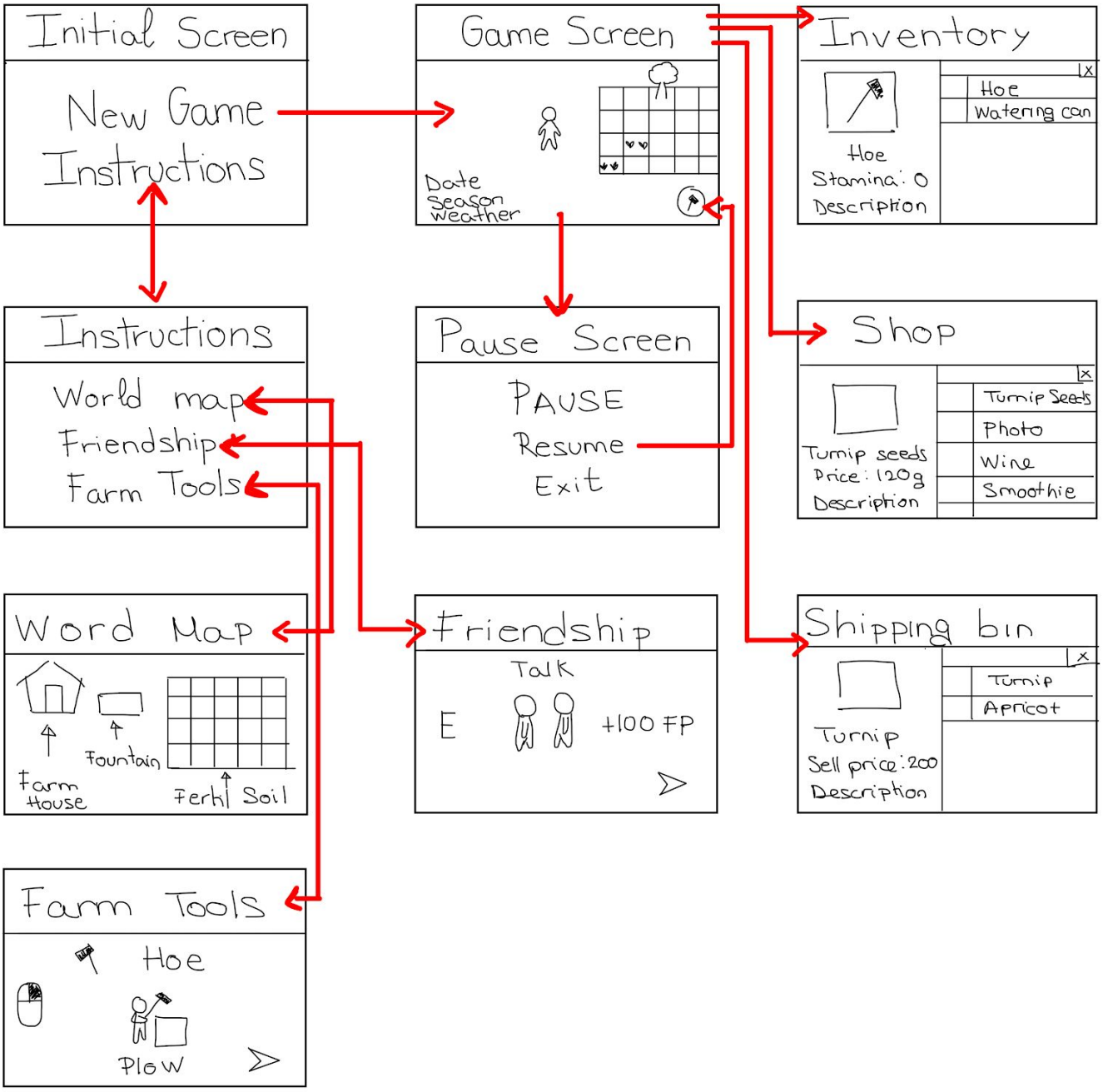


*Illustration 2: Screen flow*

Illustration 2 shows the screen flow of the game:

- From the **Initial Screen** the flow can move to the Game Screen if the player presses the New Game button or to the Instructions Screen if the player presses the Instructions button.

- From the **Instructions** Screen the flow moves to World Map, Friendship or Farm Tools screens. On those screens different instructions are shown.
- From the **Game Screen** the flow can shift to the Pause Screen, the Inventory Screen, the Shop Screen and the Shipping bin Screen.
    - Pause Screen: When the player presses “Esc” the screen shown is the pause screen, the player can resume or exit the game.
    - Inventory Screen: If the player presses “I” the inventory screen is displayed. To return to the game screen the player needs to press the “X” button.
    - Shop Screen: If the player interacts with the shop counter the shop screen is displayed. The player can buy items from there. To exit the player needs to press the “X” button.
    - Shipping bin Screen: When the player interacts with the shipping bin its screen is loaded. The player can sell items from there. To exit the player has to press the “X” button.

## 2.3. Section 3 - Story, Environment and Characters

### 2.3.1. Characters

The game has diverse characters, each one has a little background which they will tell us with the time.

- Player: Tired of the stressful city life and the people around him, decides to buy a farm in a small village to start over and “detoxify” himself of the modern life.
- NPCs:
    - Emily Lawrence (F): When she was a little girl he become sick and her parents decided that it was better for her health to be in the village with her grandparents. The city pollution was not great for her health so she decided to stay in the village with her grandparents.
    - Riley Bates (M): He was born and raised in the village, his dream is to go out the village and travel around the world.
    - Tyler Davis (M): He likes to play in the town square. His dad had to leave the village in order to work.
    - Lily Davis (F): Tyler's mother. She lives alone with her son in the village. Her husband had to travel to the city in order to work and to take care of his parents. He will come back to the village when his parents feel better and he has made enough money.Madre de Tyler.
- Deities: There are three deities in the game to whom the player can pray in the town church.
    - Matris (F): Friendship goddess.
    - Vyphy (F):  Nature goddess.

- Lonera (M): Animals god.

### 2.3.2. Story And Narrative

The narrative design is really simple because this game is not centered in narrative and it pretends to offer an entertaining gameplay. First of all, when the player starts the game a little bit of the main character story will be shown. Apart from this, as the player becomes friend with the villagers they will tell him a bit of their background.

# 3. Project development

On the following chapter the technical and artistic details will be provided as well as examples of the development.

---

## 3.1. 2D Art/3D Modeling and animation

### 3.1.1. 2D Art

#### 3.1.1.1. NPC's portraits

*Illustration 3: All NPCs  portrait*

On illustration 16 all the portraits of the NPCs are shown, they were drawn and painted on photoshop and the outline was made with illustrator.

#### 3.1.1.2. Object portraits

For the creation of the 2D portraits of the objects the final solution was to render the 3D model of the object, make the image shape quadrate, remove the background and add a shadow around it so that the image was more visually pleasant.

#### 3.1.1.3. Interfaces

The interfaces of the game are mainly designed in two colors, orange and blue:

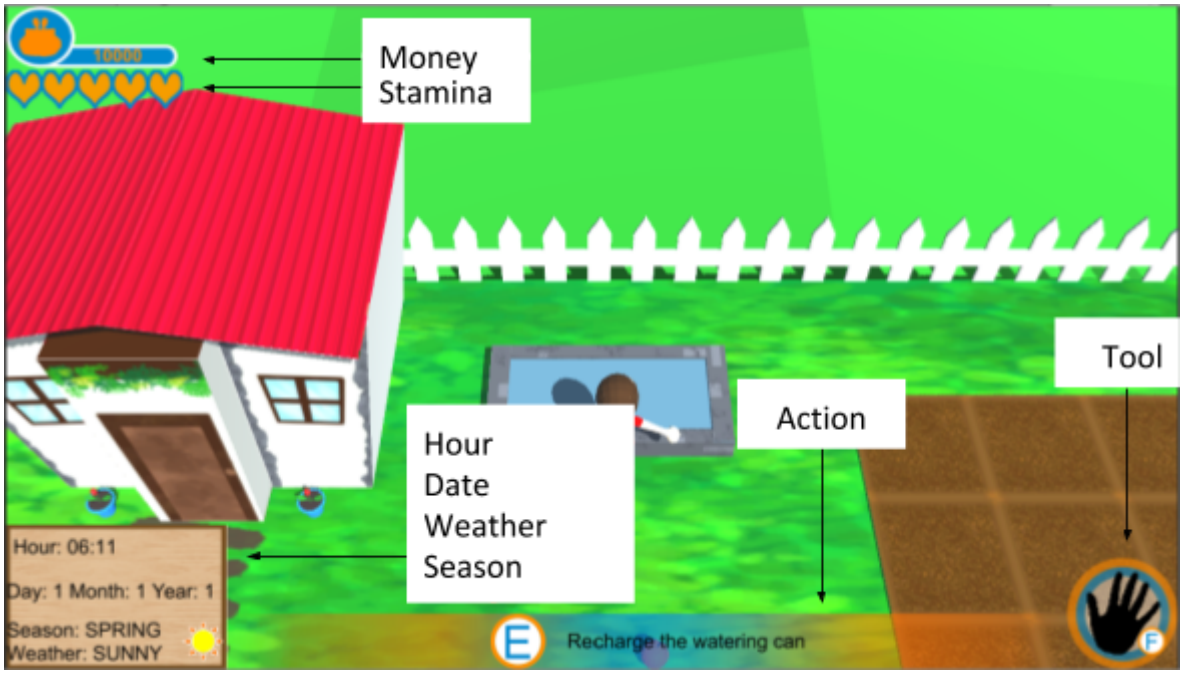


*Illustration 4: Main interface*

The information displayed in the interface is: the hour, the date, the month and the season. Aside from that, the weather is shown too. In the upper left corner, the players stamina and money are displayed.

### 3.1.2. 3D Modeling and animation

#### 3.1.2.1. Sceneries

The game is compound of nine sceneries, and and it was said before, all the 3D models and UV textures were made by hand using blender for the 3D models and Adobe

Photoshop to paint the unwraps so that the final result was more pleasant. The eight sceneries the game is compound of are:

Farm

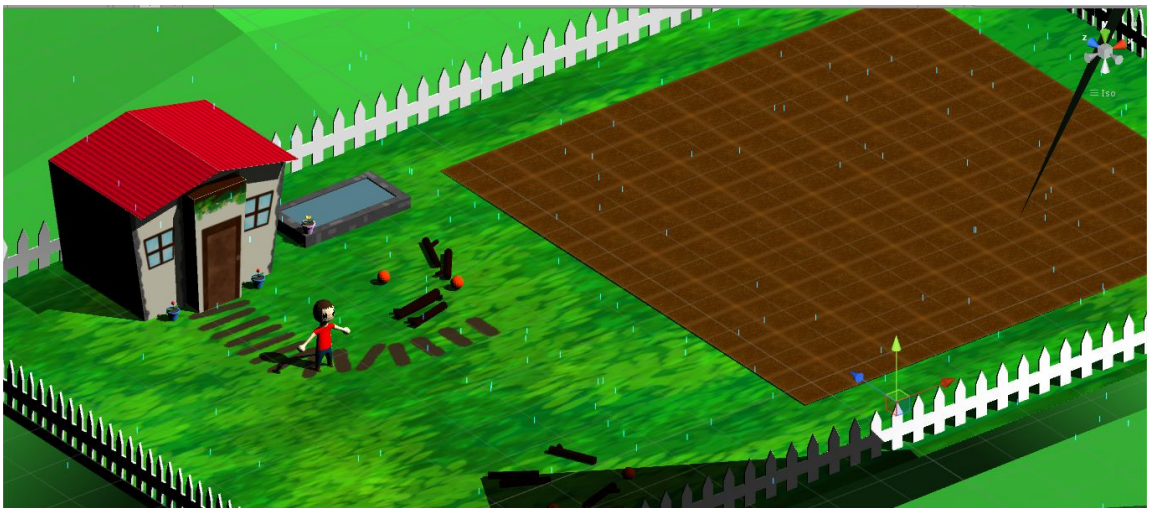

*Illustration 5: Farm screenshot (in-game)*

Farmhouse (players house)

*Illustration 6: Farmhouse screenshot (in-game)*

Road

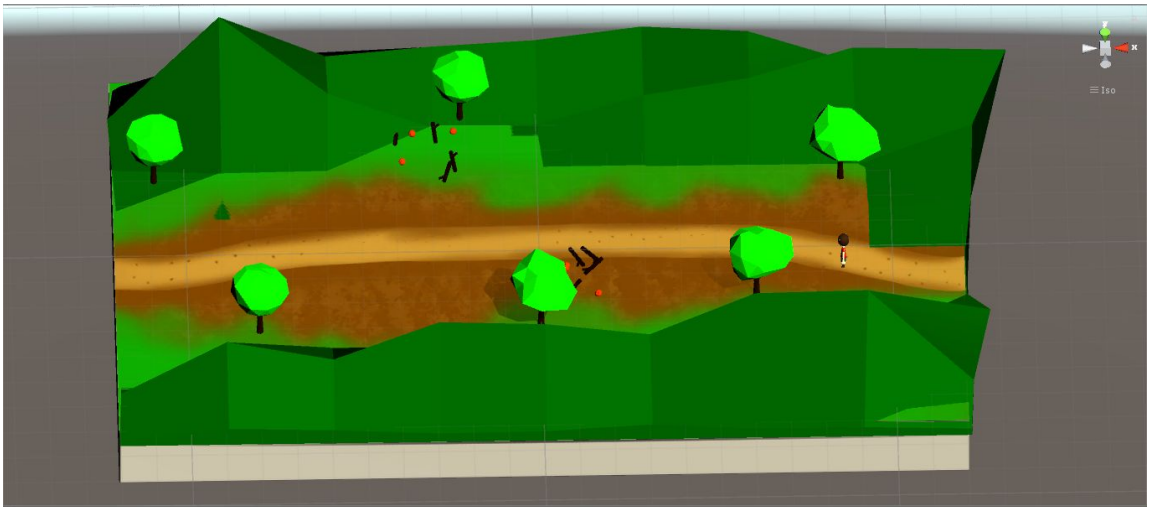

*Illustration 7: Road screenshot(in-game)*

Village

*Illustration 8: Village screenshot (in-game)*

Emily's house

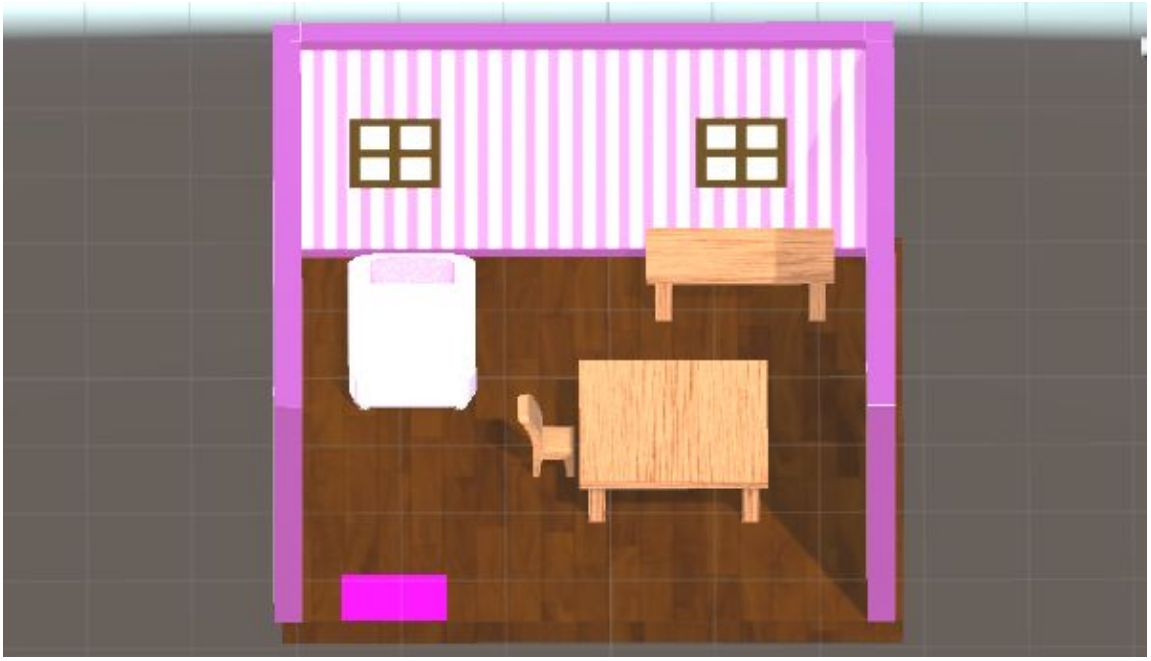

*Illustration 9: Emily house screenshot (in-game)*

Riley's house

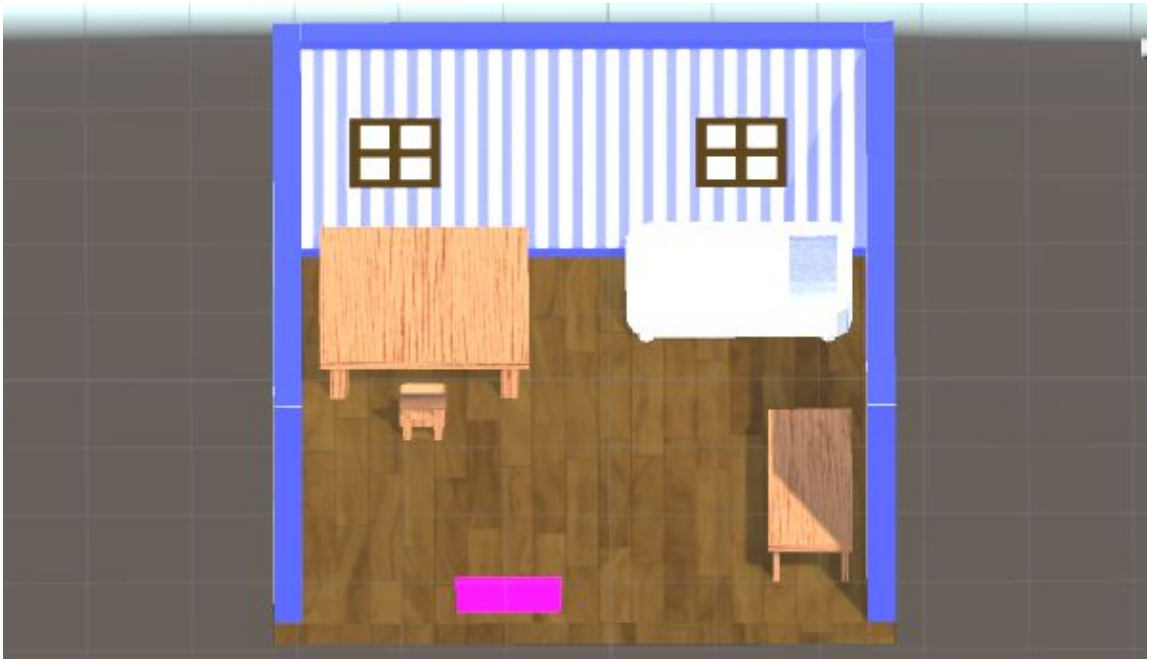

*Illustration 10: Riley house screenshot (in-game)*

David's and Lily's house

*Illustration 11: Tyler and Lily house screenshot (in-game)*

The shop


*Illustration 12: Shop screenshot (in-game)*

### 3.1.2.2. Characters models

For the 3D models of the characters a base model was created. The texture was changed (painting a new unwrap) and new walking animations different from the player ones were made.

### 3.1.2.3. Characters animations

The character the player controls has the following animations: IDLE, IDLEHOLDING, WALKING, WALKINGHOLDING, GRAB, PLOW, PLANT, WATER, CHARGEWATERINGCAN, SICKLE.

All the animations were made by hand using blender. As the model is humanoid but has different bones compared to a complete human, the bones were added manually.

As illustration 13 shows, the player animator has one animation for each action.

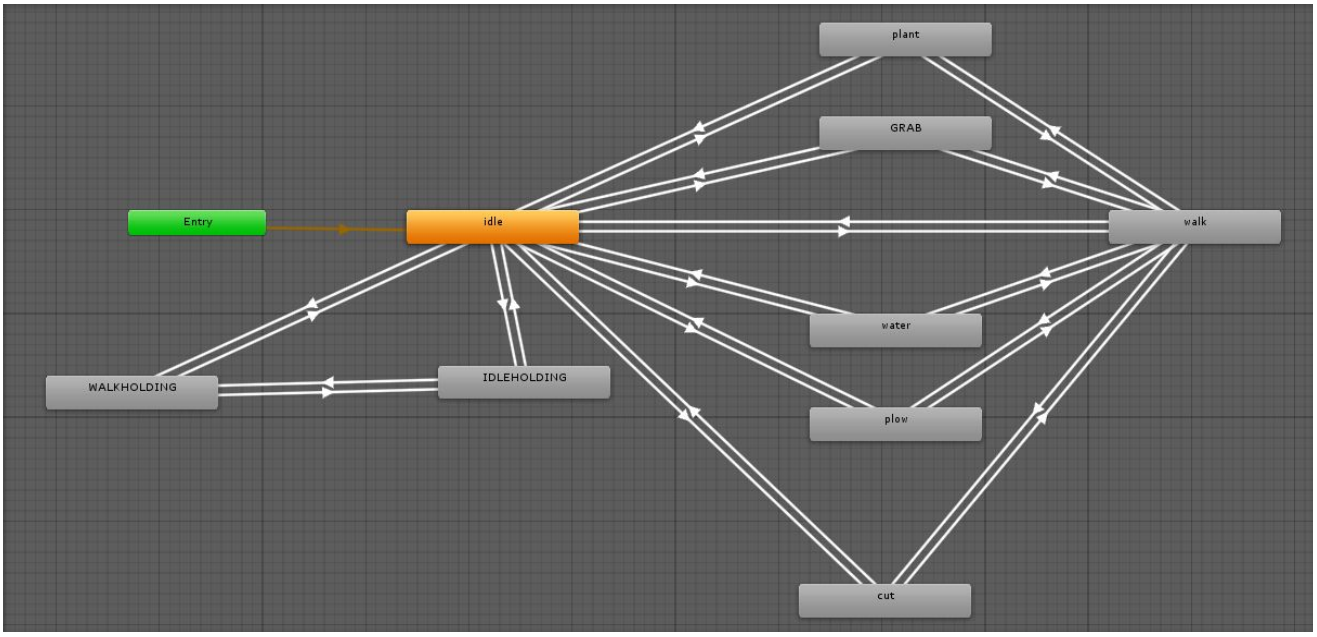


*Illustration 13: Player animator controller*

### 3.1.2.4. Objects models

The game art has a low poly style (like all the games of this genre). The objects do not have a smooth shading. The shading model is the flat shading used by most of the low poly games use.

Each one of the 3D models of the objects have been made by hand, there is no object downloaded from the Internet or picked from the Asset Store of Unity.

# 3.2 Programming

## 3.2.1. World programming

### 3.2.1.1. Map floor positions into a matrix

Initially it was thought that it would have been suitable to map floor positions into a matrix, in the end the behaviour of the floor was implemented on a different way. To program its performance a class was implemented. This class stores the GameObject of the soil and its type, each floor square is an instance of this class. When the game is started the CreateSoil script creates the soil by code, planes are instantiated and objects of type soil are created and added to a soil type list. Instead of adding it by code, I would have been possible to create a group layout and add all the maps to a parent GameObject, it would have spared some programming time but the result was not good enough for the game.

The soils works as a state machine and its states can be: NOTPLOWED, PLOWED, PLANTED, WATERED, WATEREDANDPLANTED. The NOTPLOWED state is the floor initial state, the PLOWED state is the state after plowing the square soil, the WATERED comes by when you water a PLOWED soil before it is PLANTED. The PLANTED state is the soil state when you plant a crop on a PLOWED soil. And the WATEREDANDPLANTED state is the state of the soil when the player plants a crop on the WATERED soil or when the player waters a crop. Illustration 7 provides a more visual explanation.

A further explanation of illustration 14 will be provided:

1. When we plow the soil it moves from NOWPLOWED to PLOWED.
2. If the player waters a PLOWED soil it goes to the WATERED STATE.
    a. If the player plants a crop on a WATERED soil it moves to the WATEREDANDPLANTED state.
3. If the player plants a crop on a PLOWED soil it pases to the PLANTED state.
    a. If the player waters a crop the soil will go to the WATEREDANDPLANTED state.
    b. If the player cuts or recollects a crop, the soil will come back to the PLOWED state.
4. If the player waters a crop in the HARVEST state and then recollects it, the soil will go back to the WATERED state.

The weather affects the soil states, if it is raining, when the player plows the soil it will go directly to the WATERED state, and the PLANTED soil will be in the WATEREDANDPLANTED state.

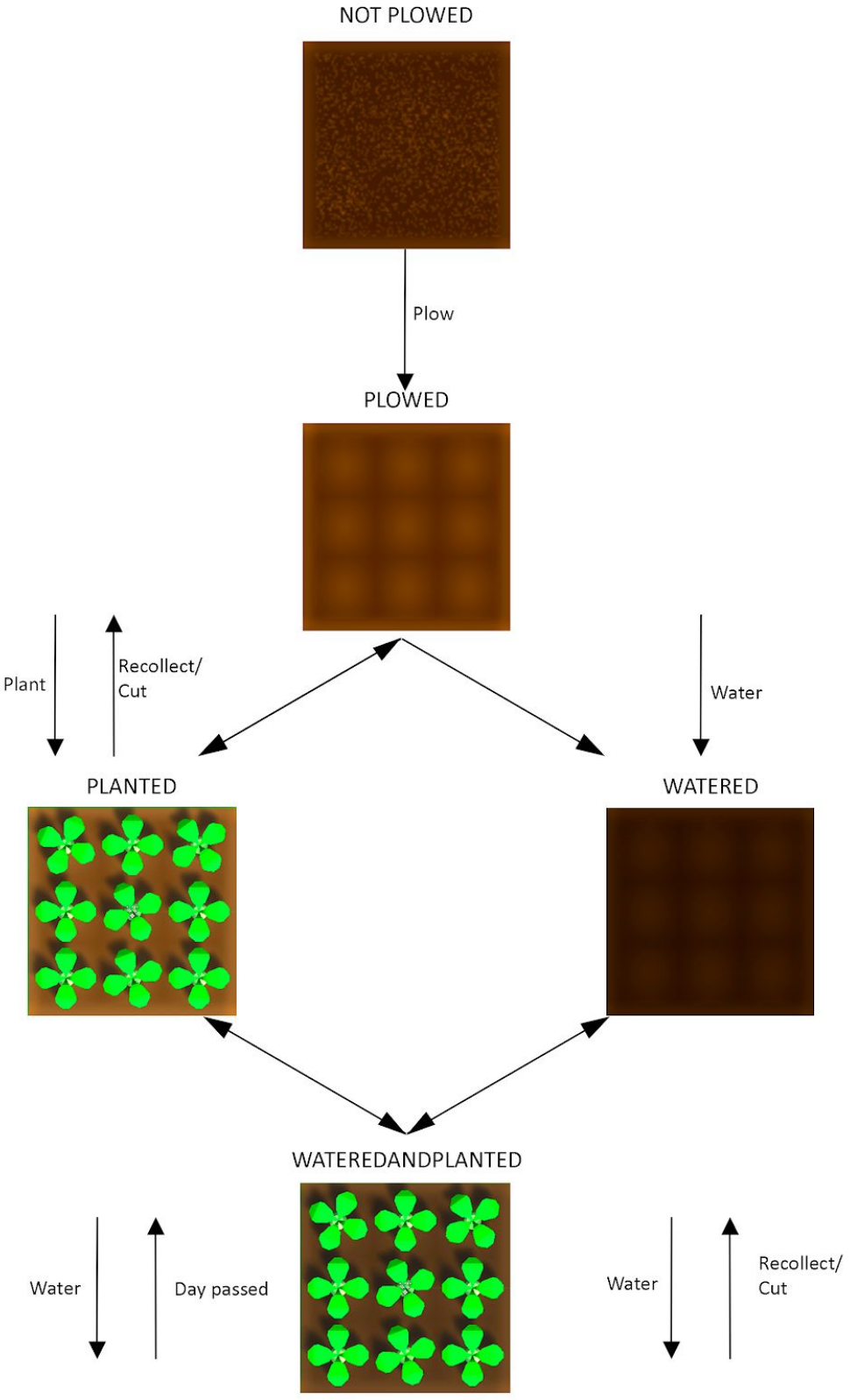


*Illustration 14: Soil statemachine*

#### 3.2.1.2. Time manager

The TimeManager script is the one that manages the pass of the time, its programming is based on the Unity Time functions.

This class controls the seasons and weather of the game. There are four seasons: SPRING, SUMMER, FALL and WINTER and each season comprises 30 days. The weather system is based on a probability system, as example, in spring and fall the rain probability is higher than in summer. Moreover, in winter it can snow. If it is raining, the player will not need to water its crops. In addition, if it is snowing the player will need to water the crops because the snow does not water them.

The time manager is in charge of resetting the game each day. When the player goes to sleep, when he faints due to the stamina going down to zero or when the day comes to its end, the resetDay() function is executed. The function is in charge of doing the following:

1. **Reset the soil.** A for loop runs through the soil list. If the soil square is on PLOWED or WATERED state it goes back to the NOTPLOWED state, if it is on WATEREDANDPLANTED or PLANTED it goes back to the PLANTED state.
2. **Update crop states**. A for loop runs through the crops list. If the crop is not on HARVEST state and it has not been watered the dry boolean of the crop updated to true and the days until it completely grows are increased by one. If it has been watered, the grow function of the crop is called.
3. **Player stamina reseted.** If the player goes to sleep its stamina is completely restored, but if the player faints, the stamina is only half recovered.
4. **Update trees.** As the trees do not need to be watered, on the for loop it is only updated its growing time and called its growth function.
5. **Update NPCs.** For each NPC on the game the talk and gifted booleans are reseted and go back to false.
6. **Update spawned items.** On the game there are four different areas where random items spawn. Each day those areas are updated in order to spawn different items each day.
7. **Update time.** The hour goes back to 6:00 am and the weather for the next day is setted.

### 3.2.2 Player programming

The player has five tools that he can use freely: the hoe, the sickle, the watering can and the seeds. Each tool has its own behaviour in order to maintain the code ordered. As the player can only interact effectively with the fertile soil, a raycast is thrown and if we can interact with the object we are clicking the player will do the action.

To control which action is the player doing the PlayerActions script was created. From there it is checked which input is the player pressing and if it is holding something or not. If the player is holding something the only actions which can be done is talk to a NPC to gift the item or to save the item which it is holding.

The tools behaviours are controlled by the ToolBehaviour script, on this script it is checked which tool the player clicked to equip it and tell the PlayerActions script which action will be done when pressing the right mouse button.

### 3.2.2.1. Player movement and actions

**Walk**

If the player is not doing an action like plow, water or plant, it can walk around the map. The player has an eighth directions movement as the illustration 15 explains.

*Illustration 15: Player movement directions*

**Water**

The soil is watered with the watering can, if the soil is PLOWED or PLANTED and the player has the watering can equipped. If the player GameObject is colliding with a soil square and the other requirements are accomplished the soil will be watered.

The detailed performing of this action is:

1. If the watering can is not empty.
2. For each soil, if the player is colliding with it and it is PLOWED or PLANTED:
3. A raycast is thrown to the object we are clicking, and if the hit of the raycast hits the soil square or the crop on the soil, the player can water the soil

When the watering can is empty, the player can recharge it in the fountain alongside its farmhouse. On illustration 16 and illustration 17 it is shown how the player waters its crops and how it recharges the watering can.

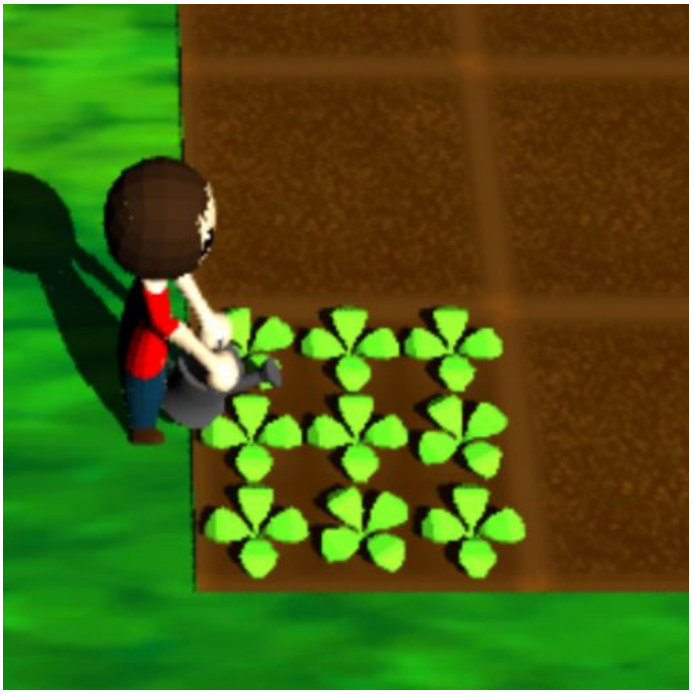

*Illustration 16: Watering*


*Illustration 17: Recharge the watering can*

**Crop**

The crops are programed in two parts: The C# class Crops and the Monobehaviour class CropsBehaviour. The Crops class is the base class for every crop, it stores its name, description, days to stage one, days to stage two, days to harvest, price, code, season and stamina recovery. The states which a crop can be in are: SEED, ST1, ST2, HARVEST.

The Monobehaviour class, CropsBehaviour, is the one in charge of updating the crop and make them grow or dry. At the start it is checked which crop the GameObject attached to this script is, the the materials and meshes of each stage are loaded. It was decided to do this that way because deleting the GameObject and instantiating a new one was a lot messier. The grow function works as follows:

- If the crop is planted on its season it is checked if it needs to go to the next stage, if it needs to grow its mesh and materials are changed.
- If the crop is not on its season or its dry boolean is true its material is changed to the dry material.

The actions related with crops are: Plant, Recollect and Cut.

Plant

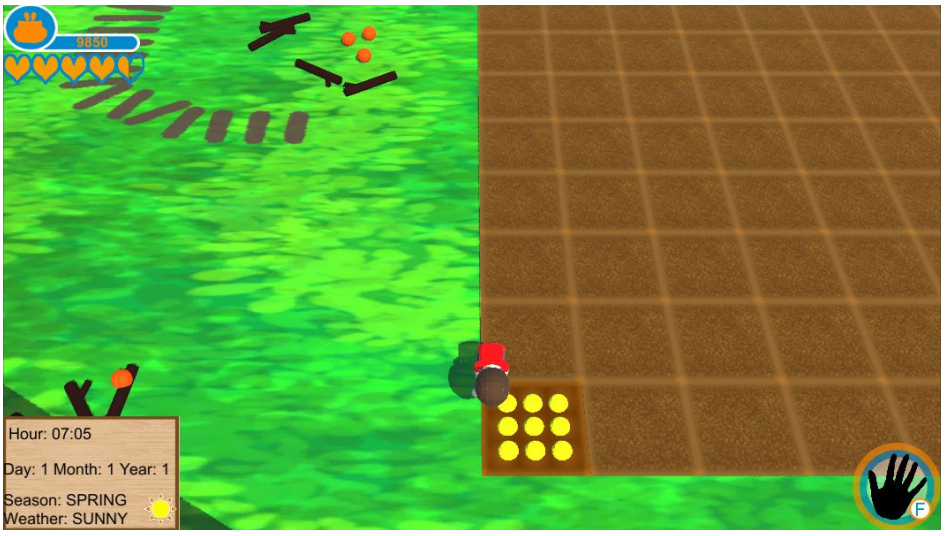


*Illustration 18: Plant action*

First of all, the plant action is done when the player presses the right mouse button on a PLOWED square of soil with SEEDS equipped. After that, the seeds GameObject is loaded using Resources.Load<Transform>("Crops/cropname +"/"+cropname+"seeds"");. The transform loaded is a prefab which already contains the CropBehaviour script. If the player uses its last seed of a crop, the button on the Tool Panel is destroyed. On illustration 18 it is shown how the player plants some seeds.

Recollect


*Illustration 19: Player recollecting*

On the second place, to recollect, the player needs to be collisioning with the crop and the crop needs to be in the HARVEST state. When the player clicks on the crop it is

recollected, added to the inventory and the crop GameObject is destroyed. Illustration 10 shows the player recollecting a fully grown turnip. Illustration 19 shows the player recollecting turnips.

Cut


*Illustration 20: Player cut*

On the third place, the Cut action is the same as the recollecting one but the crop is not added to the inventory, is only destroyed. Illustration 20 shows the player cutting a crop.

A class called MockCropDB creates all the crops that can be harvested on the game [Table1: Seasonal crops and trees].

**Plow**

If the player needs to plow a square of soil the process is similar to the watering process. The player needs to collide with a square of soil in NOTPLOWED state, if he is colliding, has the hoe equipped and right clicks the square of soil it will be plowed. The raycast hit transform needs to be the same square the player is colliding with to be plowed.

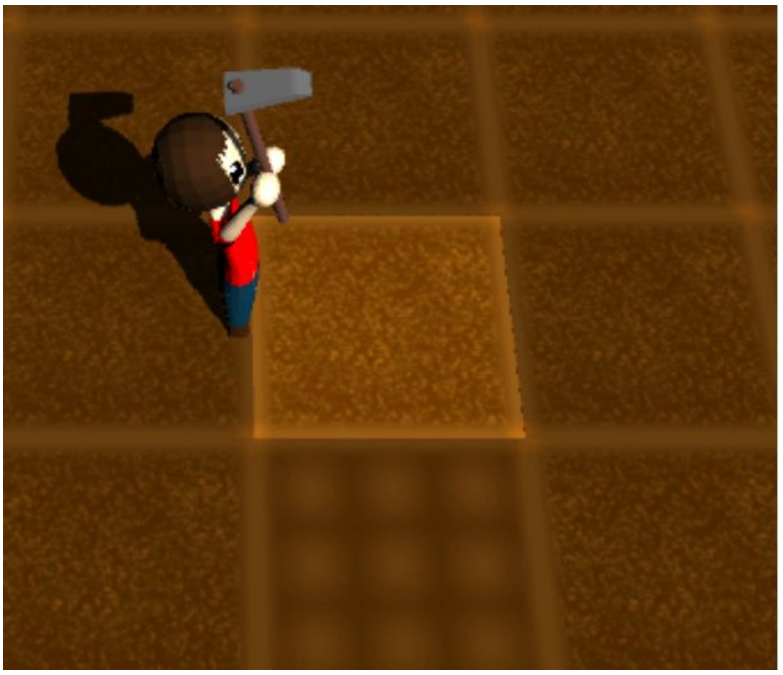

*Illustration 21: Player plowing*

**Other in-game actions/interactions**

Besides the actions already mentioned, the player can do a couple of actions no less important.


*Illustration 22: Player talking*

First of all, the player can **talk** to the villagers, he needs to position himself touching the NPC and then press “E”. Each NPC has a couple of dialogs which you can read on the talkbox. Illustration 21 shows the player talking to a NPC.


*Illustration 23: Player holding item*


*Illustration 24: Player gifting*

Secondly, the player can **hold** items as it is shown on illustration 23, why would we want to hold and item? In orden to give it as a gift to the NPCs on the game as it is shown on illustration 24. When we give a gift the player's friendship with that villager will go up and we will discover new dialogs.

Moreover the player can **open the inventory** to see the items he has. The player needs to press "I" while holding nothing. In the inventory the player can eat, hold or throw an item but that would be discussed later. Besides the player can buy objects, the player needs to go to the shop and press "E" near to the counter to open the shop and then buy objects. Lastly the player can sell items in order to gain money to buy seeds and objects. The player needs to go to the shipping bin and ship the objects he wants to sell.

#### 3.2.2.2. Object quality system

Due to time restrictions the object quality system will not be implemented.

### 3.2.3. Non playable characters programing

First of all, the player can talk with the NPCs on the game and gift them in-game items, you can only give one gift per day, and talking to them once per day will increase their friendship by 100;

#### 3.2.3.1. Create dialogs and talking system

For the dialog system a dictionary could have been created using the name of the NPC as key, but later it was decided to use different lists for each dialog. The NPCs have different dialogs depending on the players friendship with them. The lists created store three different line dialogs: the low friendship, the medium and the high friendship ones. There is a list for the thanks dialog when giving them a gift.

In the NPCBehaviour class which dialog to say is chosen depending on our friendship. As we have different lines on each friendship leven one is chosen randomly. The dialogs are displayed as shown on illustration 25.

To check if the player is talking to them or giving a gift, the code of the item is checked. If it is greater or equal to zero it is giving them a gift, if it is less than zero the player is talking to them.


*Illustration 25 : NPC dialog*

### 3.2.3.2. Route creation

For the route creation, instead of programming the pathfinding right from the start, it was decided to use the NavMesh functionality of Unity. The baked scenes are: the vilage, Emilys house, Rileys house, Lily and Tylers house and the shop.  Each NPC has a different schedule.

*Table 3: Emily's schedule*

| **Inside her house** | **Village** |
|---|---|
| 6:00am - 9:00am<br>19:00pm - end of day | 9:00 am - 19:00 pm |

*Table 4: Riley's schedule*

| **Inside his house** | **Village** |
|---|---|
| 6:00am - 10:00am<br>20:00 pm - End of day | 10:00am - 20:00pm |

| *Table 5: Lily's schedule* | | |
|---|---|---|
| **Inside her house** | **Village** | **Shop** |
| 6:00am - 7:00am<br>20:00 pm - End of day | 7:00am - 8:00am<br>17:00pm - 20:00 pm | 8:00am - 17:00pm |

| *Table 6: Tyler's schedule* | |
|---|---|
| **Inside his house** | **Village** |
| 6:00am - 7:00 am<br>19:00 pm - End of day | 7:00am - 19:00pm |

The NPCs can only move on the baked areas and when they arrive to its destination they stop for a few seconds and then they start walking again. This was done this way so that the player could easily talk to the NPCs.

### 3.2.3.3. Friendship system

The gifts that the player can give to the NPCs make their friendship go up, but it can lower it too, each NPC has some liked and disliked items which are listed on different lists. They have one favourite item, various liked and disliked and a horror gift. The items which are not listed will increase the friendship in 50 points, they are the neutral gifts.

| *Table 7: NPCs gift preferences* | | | | |
|---|---|---|---|---|
| | **Emily** | **Riley** | **Lily** | **Tyler** |
| **Favourite (+800)** | Cherry | Wine | Bracelet | Apple |
| **Liked (+300)** | • Turnip<br>• Carrot<br>• Fruit Smoothie<br>• Bracelet<br>• Book<br>• Photo | • All crops<br>• Vegetable Smoothie<br>• All fruits | • Vegetable Smoothie<br>• Fruit Smoothie<br>• Bracelet<br>• Photo<br>• Book<br>• Onion<br>• Radish<br>• Peach | • Apricot<br>• Plum<br>• Blackberry<br>• Goji berry<br>• Branch<br>• Weed<br>• Cherry<br>• Peach<br>• Orange |

| | | | | |
|---|---|---|---|---|
| **Disliked (-300)** | • All seeds<br>• Branch<br>• Weed<br>• Vegetable Smoothie | • Apricot<br>• Plum<br>• Blackberry<br>• Goji berry<br>• Branch<br>• Weed<br>• Bracelet<br>• Photo | • All seeds<br>• Branch<br>• Weed | • All vegetables |
| **Horror (-800)** | Spinach | Peach | Broccoli | Wine |

When the player talks to a NPC, an int with the code of the item which it is holding is passed as an argument to the function. If that int is greater or equals to zero the player is giving that item as a gift. To see how many friendship point the player will earn or lose the following is done:

- The howLiked function checks in which list the item is.
- Depending on which list the item a string is returned: favourite, liked, disliked, horror or neutral.
- Depending on the string returned the friendship points are increased or decreased and a thanks dialog is chosen.

The favourite item will increase their friendship by 800, the liked items by 300. And the horror gift will make its friendship go down by 800 points, and the disliked items by 300. To understand the functioning of those behaviours, on the following link the code can be seen:
https://github.com/HelenaOr/Mythrest-city/blob/master/Assets/Scripts/NPC/NPCBehaviour.cs

### 3.2.3.4. Shop creation


*Illustration 26: Shop details*

There is a shop where the player can buy the objects on the referenced table [Table 2: In-game items and its trading prices]. The shop is compound of the following classes:

**ShopItem**

The ShopItem class is the one which stores the details of each shop item (code, name, description, buy price, item type...). Is the base class for all shop items.

**MockShopDB**

This class creates all the shop items, it will be explained later, but the game does not have a database because there are not so many items to need one. After creating them they are stored into a list made of ShopItem object types.

**ShowShop**

In the ShowShop class, for each item on the ShopItem list the shop item prefab is loaded from the resources the shop item prefab which contains an image for the background, an image for the item image, a button on which the player can click to buy the

item and a text with the price. Moreover, the item code is stored on the ShopButtonItemID class, which only stores an int with the item code.

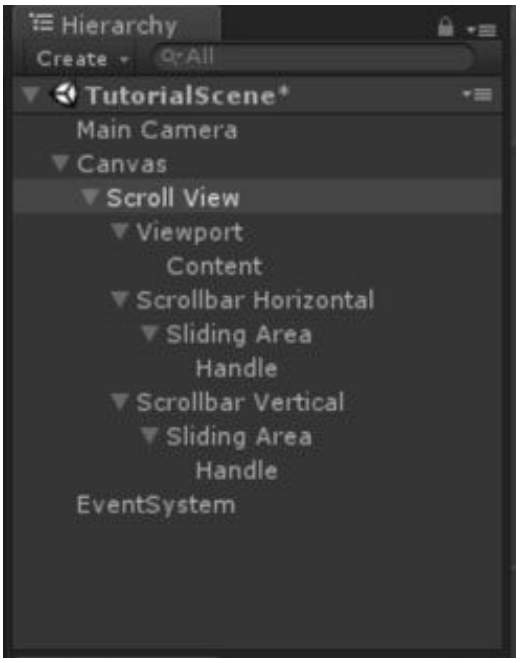


*Illustration 27: Unity ScrollView*

Besides for creating them, they are displayed on a ScrollView. First of all it is going to be explained how the ScrollView works. The ScrollView is an Unity component which allows to create dynamic scroll content. It is pretty useful when the user needs to do a scrollable list. As it is shown on Illustration 27, the ScrollView UI object is compound of a Viewport and two scrollbars, one vertical and one horizontal. The Viewport contains the content. The content is in charge of storing the items which are wanted to be displayed. As what it is wanted to be displayed is a list, it will be needed a vertical layout group to format the view. In each shop item prefab it will be needed a layout element to control its size. In this script, the prefab already mentioned is set as child of the content of the ScrollView.

When the shop is closed, the prefabs are not destroyed, so what it is done when the view is reopened, is to check if the number of childs of the content is correct, if the content has all his children we do not do nothing, but if the content does not have all the expected children, they are reinstantiated.

**ShopButtons**

ShopButtons is the script where we add the listeners for the button on the prefab. When the player clicks on an item that is not a seed is stored in the inventory, but if a seed is brought, it is added not only to the inventory but to the tool panel, where we switch our

tools. A button with the brought seed is added using the Resources.Load<Button>("...") function.

**ShopManager**

In this script we manage the visibility of the shop, when the player opens it, the scripts make its panel visible and sets the time scale to 0.0 so the player can not do actions.

**ShowShopItemDetails**

This script is the one in charge of showing the item details such as its image, name, price and description as shown on the illustration 26 above. The behaviour of this script is the following: It checks if the player has putted the mouse over the item without clicking on it, if the mouse has been over the item it shows the description.

**ShopButtonItemID**

This behaviour stores an integer which is the item code so it can be searched when the button is clicked.

## 3.2.4. Database related

### 3.2.4.1. Database creation

Finally it was decided to not implement a database, the game has forty items, so implementing a database was going to consume a time that could not be spent. Unity does not provide an easy system to implement databases and store them like, for example, AndroidStudio provides. There were created a couple of classes in which the item types needed are created:

MockShopDB → Create ShopItem types which will be used on the shop creation.
MockCropDB → Create Crops types which will be used on the CropBehaviour.
MockNPCDB → Create NPC types.
MockDialogsDB → Create dialogs for the NPCs.

### 3.2.4.2. Game saver

Due to time restrictions the Game Saver could not be implemented.

### 3.2.4.3. Inventory system


*Illustration 28: Inventory details*

For the game an inventory system has been created from which the player can throw elements, hold them by making an instance on the game world or eat them. The inventory is compound of two panels, one holds the item description and the other the ScrollRect which holds all the inventory items. Besides the ScrollView, the inventory has a planel which stores the hold, throw, eat and cancel buttons. This panel is Loaded from the resources folder when the player clicks on the button of the item.

The construction of the inventory is done by the following scripts:

**InventoryItem**

The InventoryItem script stores the name, description, code, sell price, stamina recovery and the type of all the items the player has in its inventory.

**ShowInventory**

As it was said before, the inventory is compound of a ScrollView and its behaviour is similar to the ShowShop one. The inventory creation on the scene is controlled by the ShowInventory script, each inventory item has a boolean which controls if the item has been added to the view, if an item has been already added it is not added again, besides that, the number of children of the content is checked too to avoid errors.

To create the inventory item prefabs, which will be stored on the content, we load the image of the object by code through Resources.Load<Sprite>("..."); as each image is stored on a folder depending on its type, it is checked the inventory item type too to load the image from the correct folder.

As we did in the ShowShop script, the content of the ScrollView is set as the parent of each inventory item prefab, so it is properly placed on the scene.

**InventoryItemDetails**

This class works exactly the same as ShopItemDetails, it was created because the information loaded is different and comes from other places different than the shop information and the code was wanted to be clean and neat.

What it is done is that when the player passes the mouse over an item without needing to touch it, the name, stamina recovery, description and image are shown.

**InventoryManager**

The inventory is stored on a list of InventoryItems, it was decided to use this kind of structure because the remove and adding functions produced a more optimal and clean result.

The InventoryManager controls all the operations which can be done with the inventory. The operations are:

1. AddItem(InventoryItem item)
2. GetItem(int code)
3. DeleteItem(int code)

On the AddItem(InventoryItem item) function, the new inventory items are added. If the player already has an item and one more is added, the quantity increments by one or more. GetItem(int code) returns the InventoryItem which was requested. Finally, DeleteItem(int code) deletes the item requested.

**InventoryButtons**


*Illustration 29: Inventory item options*

In this class the listeners for each button of the inventory item prefabs are added. As when the inventory is closed the parent of the prefabs is disabled, when the object is enabled again it is needed to remove the listeners and add them again to avoid some problems. Illustration 29 shows the options panel of each inventory item.

If the listeners were not removed, the planel which controls the actions the player can do with the items was duplicated, triplicated... depending on how many times the inventory had been opened. So the decision was to remove all the listeners at the beginning and add them again.

When the player clicks on a button, a menu with its options is displayed. This menu is not created by code, when the listener for each inventory item prefab is added, the item code is passed as an argument to the listener. In the listener function of the options panel, the item code is passed as an argument too with some other arguments necessary to close the panel or destroy the inventory item in case that the player wanted to throw it.

To eat an item, it needs to be edible. If it is edible, it is possible to eat it and recover some stamina. The player's stamina value is updated as well as the slider value. If the last item is eaten (the player has one apricot and then eats it) it is removed from the view and from the inventory.

To hold an item it is instantiated on the game world as an object child of the player's GameObject. When holding an item the player can not perform any other action than giving it as a gift to a NPC.

Lastly, when the user want to throw an item the only thing needed is to click on the button, all the quantity of the item selected will be deleted. It was decided to throw all the items because if the player wants to throw, for example, all the branches on the inventory it was better to do it at once.

**InventoryButtonItemID**

As in the shop, an integer with the item code is stored on each button to have persistent data.

# 5. Results

Oh the following sections it will be discussed if the bachelor's thesis expected results were accomplished and some deviations.

---

## 5.1. Expected results

- ➔ A videogame where the player has a good time.
- ➔ The chance to grow the game bigger over the time.
- ➔ The possibility to add another farming/life simulator to the market over the time.

First of all, regarding the first point of my expected result, in my opinion the game makes the user have a good time, it is really fulfilling see the crops grow and the recollect them and see how you money increases and how you make friends with the villagers on the town. It is true that this bachelor's thesis is only a demo and there are lots of thing I would have liker to include but I did not have all the time I would have needed and had to pick the most important things that a farming/life simulator needs to include to be fun and interactive.

On the second place, allusioning the point "The change to grow the game bigger with time", it can be done if I keep working on it, and I am already thinking in it.

Lastly, I would be really pleased if I could include the game in the market and see what people think of it and if they really enjoy the experience of playing the game.

## 5.2 Project results

In the end the project on its totality is what I expected it to be, the player can use each one of its tools perfectly. The player can plow, water, plant, recollect, cut, water, buy objects, ship them to earn money, give them as a gift and talk to the NPCs. So the game has:

*Table 8: Project results*

| | | |
|---|---|---|
| ● Player actions<br>○ Plow<br>○ Water<br>○ Plant<br>○ Recollect<br>○ Cut<br>○ Water<br>○ Buy items | ● Crops and trees<br>○ Different crops<br>○ Seasonal crops<br>○ Crop stages | ● Weather and time<br>○ Different seasons<br>○ Different weather<br>○ Weather effects |

| | | |
|---|---|---|
| ○ Ship items<br>○ Give items as a gift<br>○ Talk to the NPCs | | |
| ● NPCs<br>○ Routes and schedules<br>○ Talk system<br>○ Friendship and gift system<br>○ Dialog system | ● Art 2D/3D<br>○ Eight different sceneries<br>○ Four different NPC models and portraits<br>○ Item portraits and models<br>○ Background canvas art<br>○ Soil textures<br>○ Interfaces | ● Other<br>○ Economy system<br>○ Save the information between scenes |

## 5.3. Other results

During the development of the project each task I completed was like completing a completely new objective. I think the crop growing system is an objective perfectly completed. The crops grow on time, they change its models and materials and they are visually pleasant.

While programming the demo, I realized that every time someone set me aside on the programming tasks meant nothing, I can perfectly do the programming of a game and make it look good. Doing this project and choosing it to be a videogame instead of only art with a little programming was a really good decision to make.

# 6. Conclusions

## 6.1 Project deviations

At the start of the project I thought it would be much more difficult to program certain things, and much more easier to program others. But while I was developing the project I noticed that things that I had underestimated where much more difficult to implement and I spent tons of hours that were not planed doing them.

Tasks like the crop system, the inventory or the shop consumed a big amount of time and I had to set aside other tasks. The 3D modelling consumed a big amount of time which I had not expected, I supposed that, how the game was going to be low poly style the 3D models could be done easily, but I was wrong, each 3D model of a scenery costed a big amount of time. In the game there are eight sceneries, and each one costed 3 hours or more depending on the simplicity.

While some developing phases, I understood that my initial task planning was wrong, setting aside the time deviations, my task division should have been something like this if I wanted to complete them all:

*Table 9: Alternative task planning*

1. Player Related
    a. Player subsystems
        i. Money
        ii. Stamina
    b. Actions
        i. Plow
        ii. Water
        iii. Sickle
        iv. Pick up
    c. Model
    d. Animations
2. Crop related
    a. Behaviour
        i. Growing
        ii. Changing meshes and materials
        iii. Detect if watered

b. Models
3. NPC related
    a. Actions
        i. Talk system
        ii. Gifts
        iii. Routes
    b. Models
    c. Portraits
4. Subsystems
    a. Time Manager
    b. Inventory
    c. Shop
    d. Weather system
    e. Interfaces
    f. Game Saver
    g. Instructions screen

With a task division like that i could have done all the tasks, but I was greedy at the start of the project and could not handle all the work. I have learned to not underestimate little things while programming because every little thing counts when you are developing something and want it to be perfect.

As I mentioned on the previous point, I had to create some scripts and behaviours which I did not thought about, like the money system, the stamina system and the selling system.

Finally, on the technical proposal an ideal planning was proposed (point 1.4), but in the end the hours and tasks diverted from the original ones as the following tables explain:

*Table 10: Game art project deviations*

| **2D art/ 3D modelling and animations** | | |
|---|---|---|
| **2D art** | | |
| NPCs portraits | 10 | 15 |
| Object portraits | 6 | 6 |
| Interfaces | 4 | 7 |
| **3D modeling and animations** | | |
| Sceneries | 5 | 10 |

| | | |
|---|---|---|
| Characters models | 10 | 10 |
| Character animations | 20 | 15 |
| Object models | 10 | 13 |
| **Total hours** | **65** | **76** |

*Table 11: Programming project deviations*

| | | |
|---|---|---|
| **World related** | | |
| Adding sceneries and hitboxes | 2 | 4 |
| Map floor positions into a matrix | 20 | 20 |
| Time manager | 15 | 10 |
| **Player related** | | |
| Unity animators | 3 | 3 |
| Player movement and actions | 40 | 50 |
| **NPC related** | | |
| Create dialogs and talking system | 5 | 10 |
| Route creation | 20 | 10 |
| Friendship system | 10 | 10 |
| Shop creation | 10 | 15 |
| **Database related** | | |
| Database creation | 10 | 5 |
| Inventory system | 10 | 15 |
| **Testing and error solving** | | |
| Error solving | 30 | 40 |
| **Total** | **185** | **192** |

| *Table 12: Project documentation deviations* | | |
|---|---|---|
| **Documentation** | | |
| Technical proposal | 6 | 6 |
| GDD | 10 | 10 |
| Final memory | 25 | 30 |
| Presentation | 9 | 9 |
| **Total** | 50 | 55 |

## 6.2. Objectives

- ➔ Create 2D and 3D art appropriate for the game genre(farming/life simulator).
- ➔ Make a playable demo which faithfully represents the final game.
- ➔ Develop a game which provides a fun and dynamic game experience.

Regarding the objectives I had at the start of this project I think I accomplished them. I made lots of 3D models and 2D art. The sceneries in the project are fidel to the game genre. They are simple but pretty and there is a variety of them. Even though that initially was supposed that the art was going to be more important, during the development of the demo was found out that the artistic part was going to be more secondary than the programing and technical parts.

Moreover, I made a playable demo which faithfully represents the game that was thought at the start of the project. Event if there have been parts that could not be developed, the developed parts are a good representation of the game genre. I learned new things while developing the project, my knowledge branch has never been the programming branch, but developing this project I have noticed that my knowledge has improved a lot regarding this.

Finally, the game provides a fun experience, growing your crops and make friends with the villagers are the chore of all the games I wanted to honour.

## 6.3 Video and executable

Here the build and video to the project are provided:

- Build: https://drive.google.com/open?id=1j_Fe3IdRzIWtIo2xykpCnXuL7GFLj7Jt
- Video: https://drive.google.com/open?id=1HD7xEJdlwDALsO1p4YN6ze4zTd-QEQmf

# 7. References

[1] Ushi no tane. *Ushi no tane, a harvest moon help site*[online]. http://fogu.com/ [Last access: 17/05/2018]

[2] Steam. *Steam: Harvest Moon: Light of Hope*[online]. http://store.steampowered.com/app/585900/Harvest_Moon_Light_of_Hope/?l=spanish [Last access: 10/05/2018]

[3] Steam. *Steam: Stardew Valley*[online]. http://store.steampowered.com/app/413150/Stardew_Valley/

[4] Blender: https://www.blender.org/

[5] Inkscape: https://inkscape.org/es/

[6] Animal Crossing: http://animal-crossing.com/es

# 8. Annex

---

[1] CreateSoil.cs script:
https://github.com/HelenaOr/Mythrest-city/blob/master/Assets/Scripts/CreateSoil.cs

[2] TimeManager.cs script:
https://github.com/HelenaOr/Mythrest-city/blob/master/Assets/Scripts/Managers/TimeManager.cs

[3] ShowShop script:
https://github.com/HelenaOr/Mythrest-city/blob/master/Assets/Scripts/Shop/ShowShop.cs

[4] InventoryManager script:
https://github.com/HelenaOr/Mythrest-city/blob/master/Assets/Scripts/Inventory/InventoryManager.cs