BakersDozen Tech

Assessment 2 - Interim Project Report

OUA Building IT Systems (CPT111) SP1, 2021

The Team

Channon Harper | s3871491

Jessica Cramb | s3813728

Jacob King | s3858820

Joseph Tselios | s3858508

Matthew Smith | s3210455

Editing and Formatting by
Channon Harper and Jessica Cramb

Contents

Project Description

Our project will be the development of a 2D top-down farming simulator game, called “2D Farms – Live the Life”. This game will allow the player to go through the motions of crop production and will include aspects that tie into real world obligations such as debt repayment and upskilling. Development of this game will be performed through Unity Engine for PC.

Gameplay for *2D Farms – Live the Life* will start with a rundown house and limited resources and fields to work. The game will then involve the player controlling a farmer avatar to perform tasks such as crop planting, care and harvesting. The player will then sell their harvest yield to buy better tools and house upgrades as they progress to higher skill levels. Depending on the difficulty level chosen, the player will have the option to take out a mortgage on their farm, giving them an instant increase in cash for purposes such as purchase of upgraded tools, house or seeds. The completion of tasks such as crop production or farm maintenance will increase the player experience, facilitating the progress through levels to unlock more fields and upgrades. The player will have to ensure that the farm is profitable to avoid bankruptcy, doing so will stop the bank from repossessing the farm and ending the game. The game is a pure farm and sell product. No production lines exist within the game it is a simple plant, maintain, harvest, and sell.

The game is intended to be a real-life simulator, therefore there will be a realistic time lapse component, crop failure triggers and basic financial management aspect in the form of mortgage repayment obligation and sale for profit. Paying off the bank debt without sufficient income from the crops will result in a debt cycle that will destroy the farm. The only successful completion requires achieving all three goals, the bank, the house, and the sustainable farm size. This is the main goal of the game. If a farmer makes regular payments to the mortgage, they can continue the game and live a stress-free farm life. If the farmer cannot repay the mortgage, the bank will be knocking on their door, foreclosing on the farm and ending the game. There will be no second chance avenues to revive your farm, you will need to start again which gives the game a representation of a real-world situation.

In conclusion the game is successfully completed when the rundown house is repaired, the fields are a sustainable size, and the bank is paid. With all enjoyable games there needs to be an end in sight. Paying off the farms bank debt is the main goal of the game. If a user pays off the mortgage regularly, they can continue the game and live a stress-free farm life. If the user struggles to produce and repay their mortgage they will have the bank knocking on their door, and the bank will foreclose on your farm ending the game. There will be no second chance avenues to revive your farm, you will need to start again which gives the game a representation of a real-world situation.

Core Feature 1: Player Movement

Design

The fundamental function of gameplay for *2D Farms – Live the Life* is the ability to control the farmer's movements and actions. In this feature we will focus on the movement of the farmer and look closer at player movement in terms of the avatar changing location on screen depending on user input, the animation of the avatar to simulate fluid movement and the idle status of the player avatar. More specifically, this feature will manage the following gameplay functionalities:

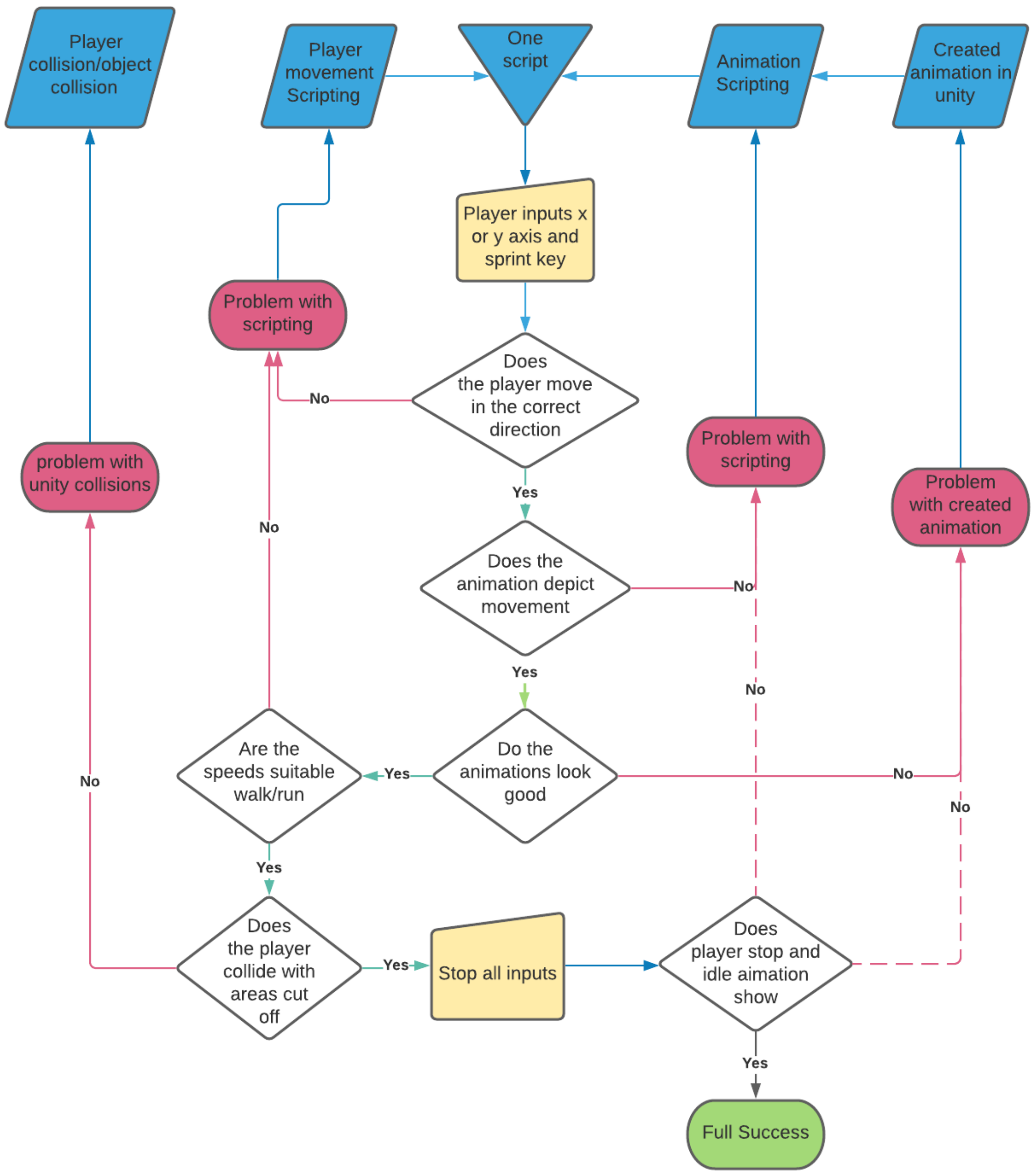
- Directional movement will be restricted to X and Y axis
- The game movement is controlled by the keypad along with a game controller e,g, 'W', 'A', 'S', and 'D' keys control movement between axis.
- The appropriate farmer animation will play for each movement direction.
- The appropriate farmer animation will be displayed when farmer is idle.
- Inclusion of a sprint function to increase movement speed allowing the farmer to get around faster with unlimited use.
- The farmer cannot move into locations where there is another object or frame border; that will cause a collision and change movement status to idle

Validation Testing

In essence there are three aspects that must perform correctly for the player movement feature to be successful, these include animation, movement of avatar and correct response to user input. The following table and flowchart show the testing procedure, relevant elements of player movement and outcomes that would be deemed as a failure.

Ideal Outcome	**Fail Indicator**
Idle status: when no keys are pressed and/or assigned control keys for direction are released the farmer ceases movement by not changing position on either axis. Farmer animation displays "idle" sprites which will be avatar facing towards camera but not changing.	a. Farmer moves to different location on screen. b. Animation from previous movement continues to play. c. Animation has stopped but avatar continues to face in previous movement direction.
Farmer moves in the correct direction as listed: ○ Up Y+ ○ Up Right Y+ X+ ○ Right X+ ○ Down Right Y- X+ ○ Down Y- ○ Down Left Y- X- ○ Left X- ○ Left Up Y+ X-	a. Farmer does not move at all b. Farmer moves in different direction.

Ideal Outcome	**Fail Indicator**
The animation changes to the correct set when changing direction as listed: ○ Up Farmer faces away from the user. ○ Up Right Farmer faces away from user. ○ Right Farmer faces to the right. ○ Down Right towards the user. ○ Down Farmer faces towards the user. ○ Down Left Farmer faces towards the user. ○ Left Farmer faces to the left. ○ Left Up Farmer faces away from user.	a. Animation does not change. b. Animation displayed is different from the one listed here.



Player collision/object collision
Player movement Scripting
One script
Animation Scripting
Created animation in unity
Player inputs x or y axis and sprint key
Problem with scripting
Does the player move in the correct direction
No
Yes
problem with unity collisions
Problem with scripting
Problem with created animation
Does the animation depict movement
No
Yes
No
No
Are the speeds suitable walk/run
Yes
Do the animations look good
No
No
No
Yes
Does the player collide with areas cut off
Yes
Stop all inputs
Does player stop and idle aimation show
Yes
Full Success

Engineering

The base of the movement will be a single script with float values that relate to delta time and move speed, modifying these with code for an input to be able to sprint. This process is made a lot easier with unity engine as most collisions are easily added and without the requirement of scripting. The animations are also made through the unity engine with the next section is required to make them. The following link will take you to the script that has been written.
https://github.com/Channon87/2dFarms/tree/main/2dfarms/Assets/Scripts/Movement

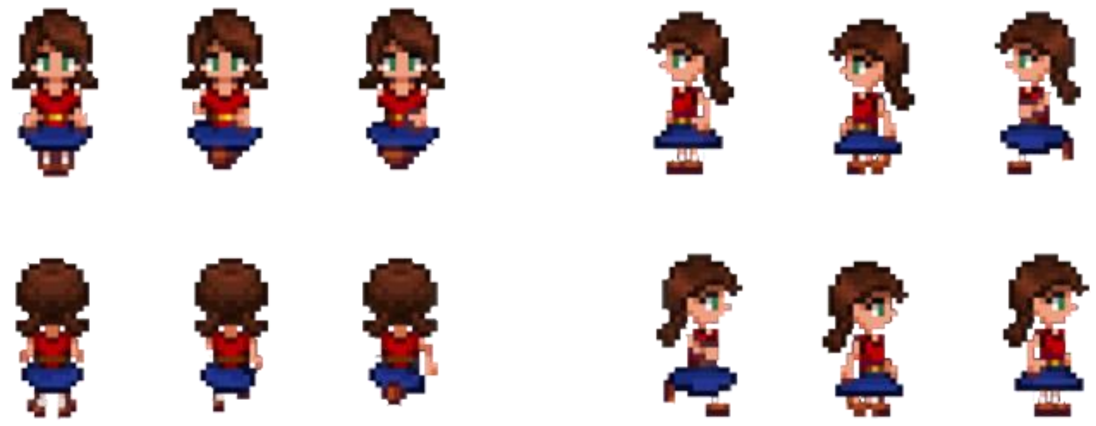
Production

Player movement basics will be covered upon the successful performance of the scripts and animations. Further to this, the collision rule applied to the player character will be tested each time a new object is added to the game. This further development for collision checking will ensure that the collision rules for all game objects in relation to the player avatar continue to perform as expected.

The following link provides a video of current player movement in action
https://channon87.github.io/2dFarms/#MVF1

Art

The visuals required to depict movement is multiple sprites, from which animations can be made. These animations change depending on user input or lack thereof and although relating to a different section these will also include change from idle to interaction when player is interacting. The following sprite sheet will be used to create our player in game and the animations necessary other sprites still need to be created with the use of GIMP for further content.



Authors: Channon Harper s3871491 | Matthew Smith s3210455 | Jessica Cramb s3813728

Create Date: 22 April 2021

Core Feature 2: Item Interaction

Design

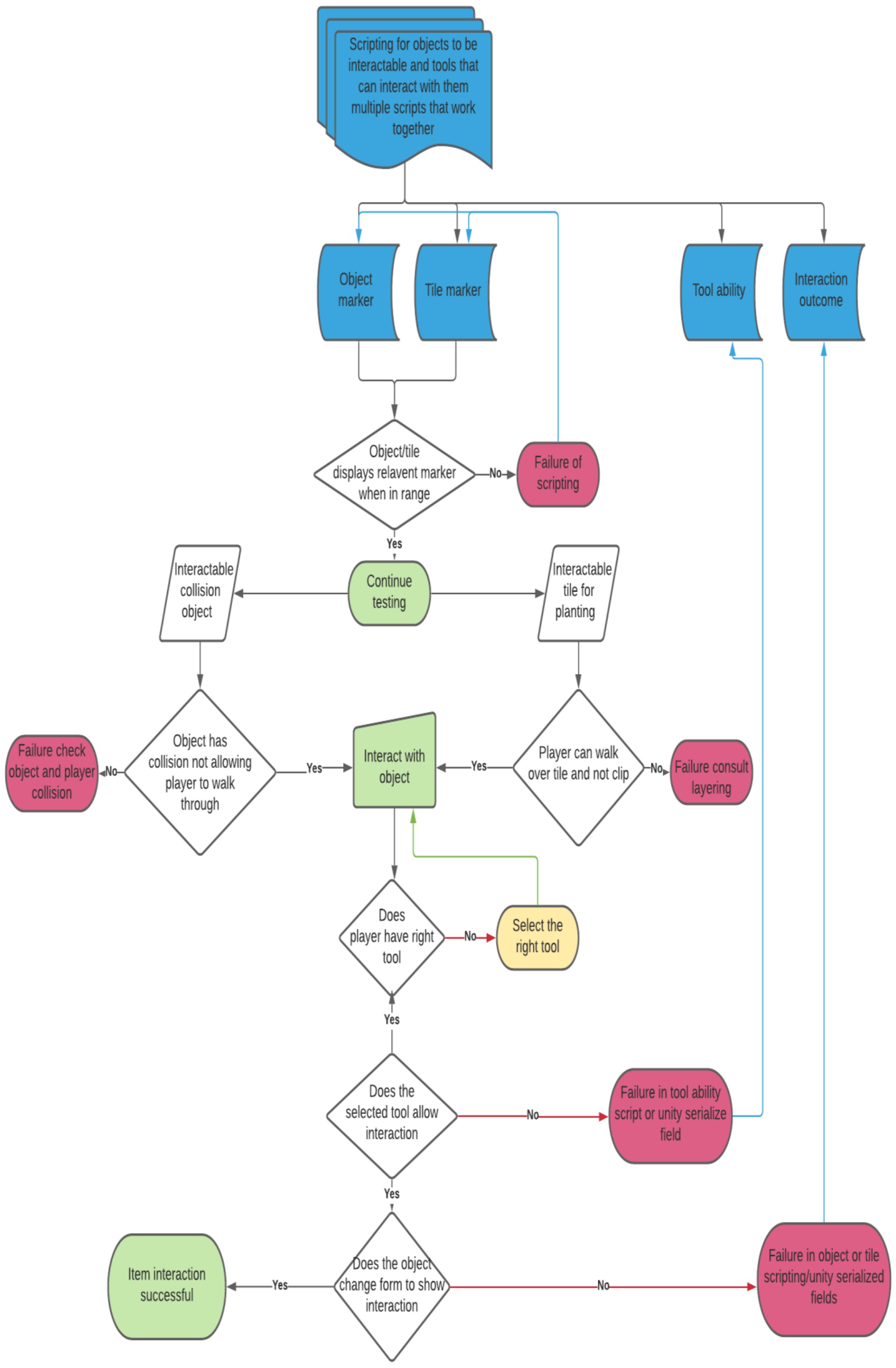
Initially the game will be incorporated with interactable surroundings be it objects in the form of chests, trees and stone to changeable tiles for farming. These objects will give the simulation of a better testing sandbox environment. Players will be able to interact with objects by moving close to them, once a player is close enough an icon will appear on screen, indicating that the object is interactive. To engage with the object the player will need to press an assigned 'action' key. This key will open the item inventory be it a physical inventory or change sprite dropping its contents. For interactable tiles the tile sprite will change to complement the interaction the player has chosen. In the following section you can find a flow chart that depicts the flow of these interactions. This description is broken down into the following points.

- Interactable objects and tiles.
- Has relevant marker to depict interaction.
- Objects have collision.
- Tiles have no collision.
- Each interaction has a tool or equip able item requirement.
- Changed visual to display interaction.

Validation testing

Most of the failings within this feature will be bug reports created through testing the game. Having a single person test the system to make sure each interactable item can indeed replicate the scripting behind them. After this point having other developers/users run the game and test over the same features identifying any problems that may occur. Each bug is tracked and fixed and retested until the feature is working as intended. Additional test objects may be introduced with failing systems built in to test the system. The following table and flowchart show the process of testing the features described previously.

Ideal Outcome	**Fail Indicator**
Object/tile displays relevant marker when player avatar is in range.	a. Marker does not appear on screen. b. Marker is always displayed, regardless of player location on screen.
Player is not able to walk “through” objects that have been assigned a collision value.	a. Player can walk through object
Player can walk over interactive planting tile and remain on top.	a. Player clips into tile and appears to be under the tile, either fully or partially b. Player cannot walk over tile at all due to invisible barrier
Interactive object only responds to correct tool e.g. trees will only chop if player has axe selected	a. Object changes regardless of tool selected. b. Object does not change at all when correct tool is selected/used.


Scripting for objects to be interactable and tools that can interact with them multiple scripts that work together
Object marker
Tile marker
Tool ability
Interaction outcome
Object/tile displays relavent marker when in range
No
Failure of scripting
Yes
Continue testing
Interactable collision object
Interactable tile for planting
Object has collision not allowing player to walk through
No
Failure check object and player collision
Yes
Interact with object
Yes
Player can walk over tile and not clip
No
Failure consult layering
Does player have right tool
No
Select the right tool
Yes
Does the selected tool allow interaction
No
Failure in tool ability script or unity serialize field
Yes
Does the object change form to show interaction
Yes
Item interaction successful
No
Failure in object or tile scripting/unity serialized fields

Engineering

As illustrated in the previous flowchart, multiple scripts will be required for the item interaction functionality and these must work together for this entire feature to work successfully. Object creation and maintenance can be accomplished through the unity engine and so it is assumed that object management can work with single scripting. This functionality is also reliant on the item inventory feature however, for testing purposes we can remove the need to reference the inventory feature scripts, should this be needed.

The following link will show most of the scripting to make this possible.

https://github.com/Channon87/2dFarms/tree/main/2dfarms/Assets/Scripts/Interaction

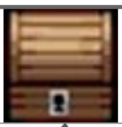
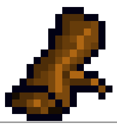
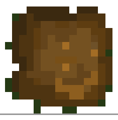
Production

Adding objects to the game can be achieved by essentially copying and pasting the code, with the only variances being the sprites and harvest yield factors. With that said, once all scripts are working for the base objects, we should be able to easily develop different objects from the source code by simply adding different sprites and yields, which will increase the games content. The following link shows a video of our current system working in its basic form with some elements remaining to be input.

https://channon87.github.io/2dFarms/#MVF2

Art

Sprites need to change depending on the interaction to let the player know that interaction has indeed occurred; this is true for interactable tiles and objects respectively. For example, a box will need to have a shut and open sprite, or a tree would need to turn into object material and disappear once harvested. The following are some of the artworks the project is currently using.

Item	**Initial sprite**	**Sprite after interaction**
Chest (default setting is closed) Interaction: player clicks chest to open it.		
Tree Interaction: player uses axe tool to chopped down tree.		
Cultivated Soil Interaction: player clicks on ground with hoe tool to plow the soil.	No sprite, just background art showing dirt area.	

Author: Channon Harper s3871491

Create Date: 22 April 2021

Core Feature 3: Time Elapse

Design

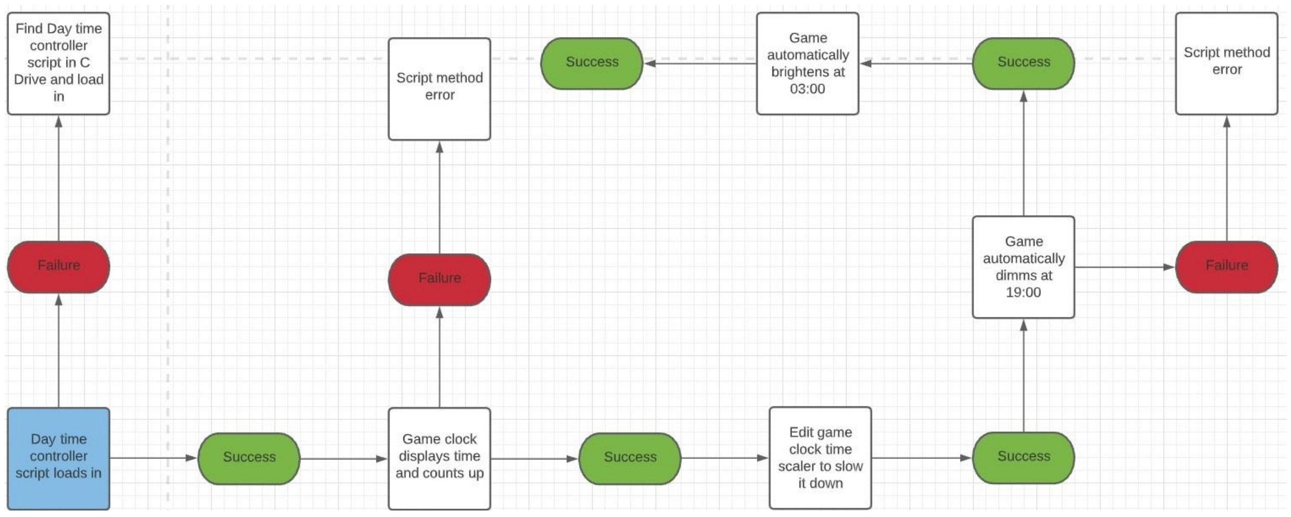
The backbone of *2D Farms – Live the Life* is the games capacity to simulate the real-life farming and a fundamental aspect of real life is that fact that time elapses, cycling between day and night. The time elapse feature for this game will simulate the change between day and night at an increased rate of speed and will be used to trigger actions on other various scripts in the game. This feature will establish the following aspects for gameplay:

- In game cycle between day and night
- Clock timer that is displayed as part of the game UI.
- Day Counter that will also be displayed as part of the game UI.
- Counter that will be utilised for crop growth and harvest feature.
- Counter that will be utilised for crop sale feature.
- Counter that will be utilised for mortgage repayment feature.

Validation Testing

This feature comprises of a central function that must operate correctly and several branched functions that feed into both the time elapse feature and other features, for example crop growth is dependent on having in game time pass a specified number of days before the crop advances to the next stage of growth. However, validation testing for other game features will not be performed as part of this feature and instead within their own respective features. To test that this feature is running correctly, we will run the game for at least three (3) in game days to test each of the following aspects:

- The game time elapsed during the testing period will then be compared against the pre-calculated figure to confirm it matches.
- Game changes colour shading to depict sunrise and sunset.
- Day counter is performing correctly, each time a new in game begins the day counter UI should also change.
- Following further development on EVF's, we will also test for in game continuation by checking that the game time listed at the time of saving is still the same when the game is reloaded.


Find Day time controller script in C Drive and load in
Script method error
Success
Game automatically brightens at 03:00
Success
Script method error
Failure
Failure
Game automatically dimms at 19:00
Failure
Day time controller script loads in
Success
Game clock displays time and counts up
Success
Edit game clock time scaler to slow it down
Success

As displayed in the above flowchart, there are points in gameplay that can be accessed for validation testing. These points are explained in greater detail in the below table:

Ideal Outcome	Fail Indicator
Value of timer variable continually increases by one(1) whilst game is in play	c. Timer variable value does not change at all. d. Variable value is being increased at an incorrect scale e.g. counts up by 3 instead of 1. e. Variable value is decreasing.
Value of day counter increases by one(1) each time the timer counter reaches a set number. This is then reflected on screen in the player UI.	a. Day counter does not change at all. b. Day counter increases at the incorrect rate. c. Player UI does not display correct value.
In game day to night cycle is depicted by changing the colour of game objects and background to a darker shade/tint. This process will commence when in game time reaches 1900.	a. Colour shading/tinting does not change at all b. Change in colour shading does commence at in game hour 1900, but instead starts at a different time. c. Colour shading is made lighter instead of darker.
In game night to day cycle is depicted by changing the colour of game objects and background to a lighter shade/tint. This process will commence when in game time reaches 0300.	a. Colour shading/tinting does not change at all b. Change in colour shading does commence at in game hour 0300, but instead starts at a different time. c. Colour shading is made darker instead of lighter.
Game UI displays time in the same format as a standard clock. This will include the clock changing with each increased minute and hour. The clock will also reset to 00:00 at in game midnight.	a. UI clock does not change b. Clock shows incorrect time (this will be checked against console reporting that displayed timer variable value) c. Clock does not reset at midnight d. Clock counts backwards.

Engineering

To add a level of realism to 2D Farm - Live the Life, the game will have a running time clock that will run faster than the real world for accelerated gameplay. The significance being that gameplay will cycle through day and night at an increased speed. Once the in-game time lapse speed has been confirmed, an estimated game time minutes/hour per each real time minute will be calculated and added to scripting. To facilitate this across multiple hardware setups we will utilise Delta Time; which gives us the ability to take current framerate of the user and calculate equations to give the same result, so no matter the hardware the player is using the timescale and other features will all function the same. The timer used to facilitate the day and night cycle will also be made available to other features that have timed events and actions. Specific mechanics of this particular feature include the following:

- Core function will be the timer counter, this will increase a timer variable at intervals of one(1). The rate at which this variable will be changed is yet to be determined, but in our initial stages we have set the acceleration at 1 real world second = 1 in game minute.
- Timer counter is the base function that will feed into other features such as day counter, day/night cycle etc.
- Day counter will increase once the timer counter reaches a specified value, again this is yet to be confirmed.
- Sunrise occurs at 0300
- Sunset occurs at 1900
- Access to timer variable is available for other scripts.
- Following further development, the time elapse feature will ensure game saving and loading lets player leave and return to game at the same point of time.

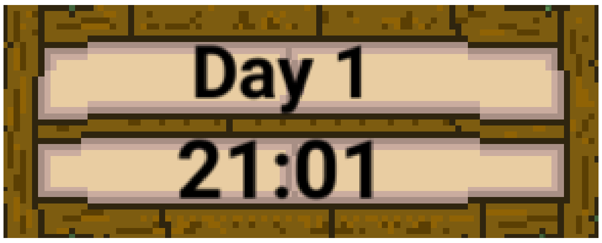
Production

Development of the time elapse feature is solely done through Unity as the majority of the work involved is performed in scripting, with a few Unity GUI inputs such as colour hue for changes between night and day. Scripting for time elapse found here https://github.com/Channon87/2dFarms/tree/main/2dfarms/Assets/Scripts/TimeLapse is contained in a single file that allows developers to set the following variables:

- Accelerator speed; sets how fast the time will pass in game compared to real world
- Sunrise and sunset times respectively; this will include commencement time and length of time that will pass before the event ends
- Lighting for the day/night cycle

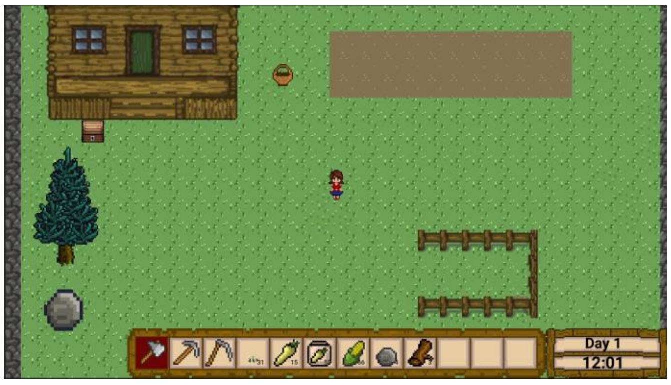
Art

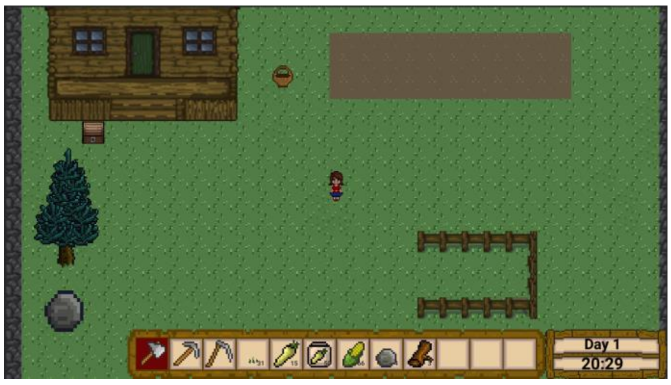
Concept art for this feature includes the game clock and day counter, that will be displayed on the game UI. The clock will show the player what time it currently is in-game and will show up on the bottom right of the screen while in play. The clock is animated in digital time.

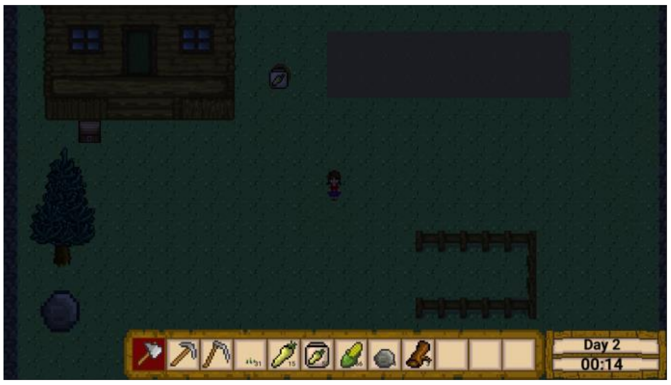


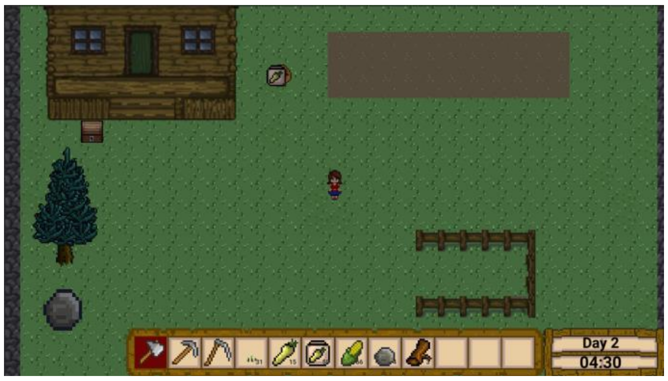
Day 156
11:46

Further, this feature controls the day and night cycle of the game. As previously mentioned this will work by changing the shading of the game objects and backgrounds. Examples of lighting during the day and different times of the day are shown below:









Author: Joseph Tselios s3858508 | Jacob King s3858820 | Jessica Cramb s3813728

Create Date: 22 April 2021

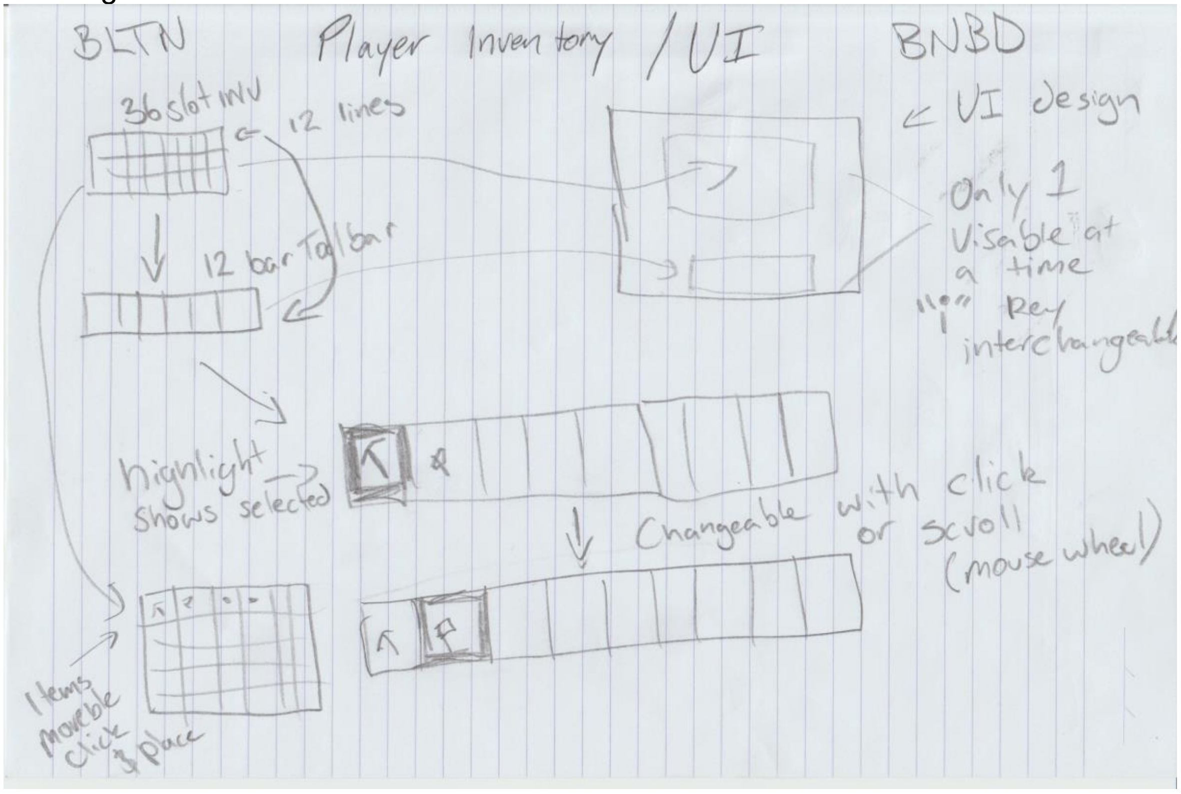
Core Feature 4: Player Inventory/UI

Design

The design for the inventory will be a thirty-six slot ‘bag’ in which the player will be able to move around scriptable objects within to store them to the player. This will be attached to a toolbar of twelve slots that is directly derived from the first twelve slots within the thirsty-six slot inventory. The toolbar will be selectable by the player and the selected item is equipped to the player for interaction purposes. The items are selectable in the toolbar by a click or scrolling of the mouse wheel, with a highlight showing the current selection. The dot points show a breakdown of this explanation.

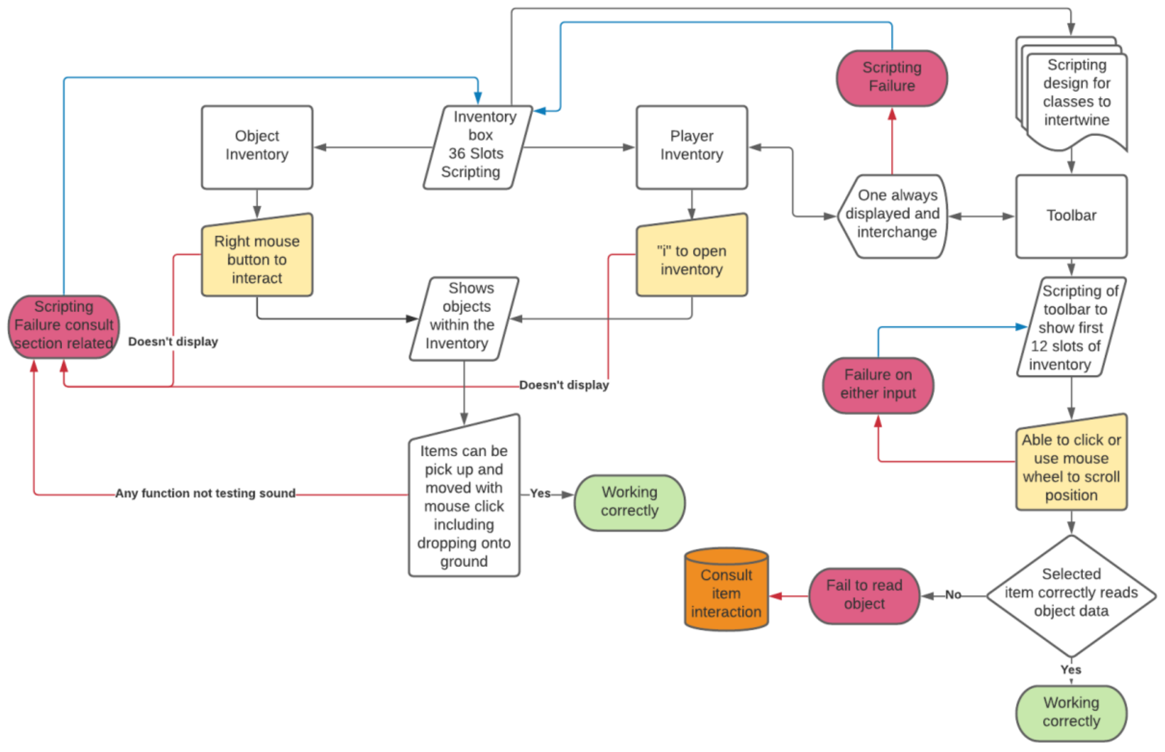
- Thirty-six slot inventories.
- Twelve slot toolbar.
- Toolbar depicts inventory (is not another bag)
- Changeable toolbar layout through the inventory
- Selectable mouse click or scroll wheel.
- Toolbar highlight shows equipped item.

The following extremely rough sketch shows the initial design concept from first meetings.



Validation testing

Most of the failings within this feature will be bug reports created through testing the game. Having a single person test the system to make sure each scriptable object is moveable, and they run the scripting desired to interact with the other objects created in item interaction. After this point having other developers/users run the game and test over the same features identifying any problems that may occur. Each bug is tracked and fixed and retested until the feature is working as intended. Addition test objects may be introduced with failing systems built in to test the system also.


Object Inventory
Inventory box 36 Slots Scripting
Player Inventory
Scripting Failure
Scripting design for classes to intertwine
Right mouse button to interact
"i" to open inventory
One always displayed and interchange
Toolbar
Scripting Failure consult section related
Doesn't display
Shows objects within the Inventory
Doesn't display
Scripting of toolbar to show first 12 slots of inventory
Failure on either input
Able to click or use mouse wheel to scroll position
Items can be pick up and moved with mouse click including dropping onto ground
Any function not testing sound
Yes
Working correctly
Consult item interaction
Fail to read object
No
Selected item correctly reads object data
Yes
Working correctly

Ideal Outcome	Fail Indicators
Pressing assigned key will open player inventory, default is “i”	a. Nothing happens, inventory does not open b. Pressing the assigned key triggers a different action e.g. pauses game instead
Right clicking with object selected triggers correct action e.g. holding a hoe and right clicking soil will turn it into cultivated soil.	a. Nothing happens. b. A different action is triggered.
Objects are displayed in inventory windows and toolbar	a. No items are present b. Item graphics do not show c. Graphics displayed do not match item
Items can be picked up and moved using mouse click, drag and release	a. Items are not added to inventory if dragged in b. Items are not removed from inventory when dragged out c. Item movement animation is not shown d. Dropped items do not appear on the ‘ground’

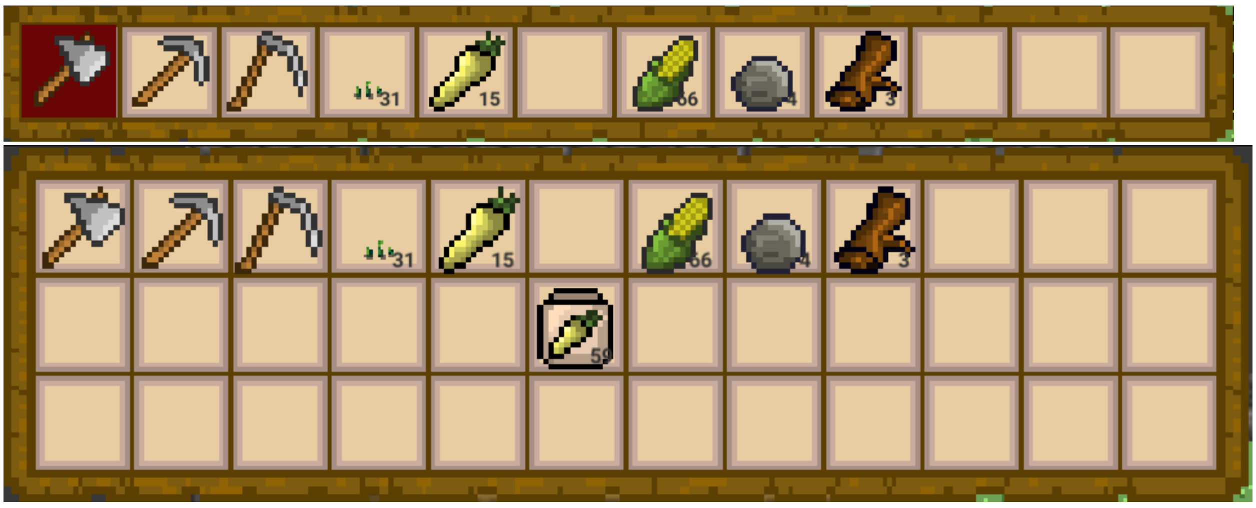
Ideal Outcome	**Fail Indicators**
Toolbar only displays the items in the first twelve inventory slots.	a. No items appear in toolbar but are present in at least one of the first twelve inventory slots b. Items displayed are in inventory slots thirteen and above.
Scrolling mouse wheel or clicking in specific toolbar slot changes selected tool	a. Item selection does not change. b. If scrolling using mouse wheel, the selection change goes in the wrong direction

Engineering

Technically a lot of scripting will be required for this design to be implemented. Both the inventory and toolbar need pockets that change depending on the scriptable object that is within. The functionality to read the object and then attach it to the player the scripting will need to be readable through multiple areas, the UI itself must also scale to render quality of the selected screen size. The initial scripting will be for the bag itself with at least two viable objects that can test the placement and changes. The following link displays full detailed scripting that has been worked on to make this possible.
https://github.com/Channon87/2dFarms/tree/main/2dfarms/Assets/Scripts/PlayerInventory

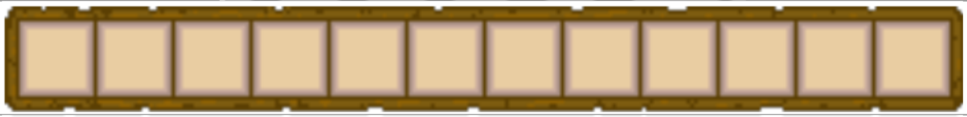
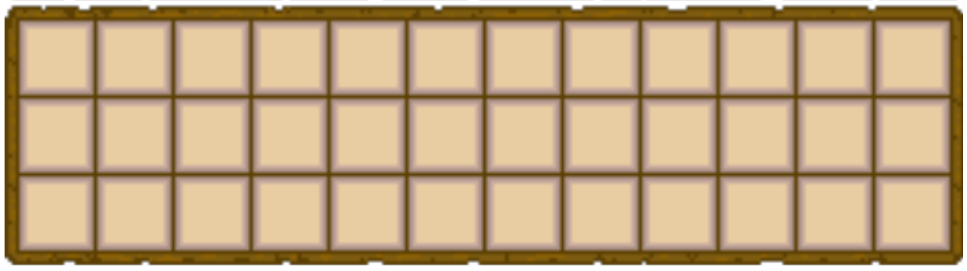
Production

Putting all the above into practice we will be able to develop the UI through the unity engine, customizing as we see fit the editable scriptable objects and inventories. The following screenshot is the base model of the working inventory and UI with the scripting done thus far on the project. Following the link will show a video capture of this in practice.
https://channon87.github.io/2dFarms/#MVF4



Art

To assist in object selection clarity, each scriptable object must have a unique image. This will also help the user differentiate the objects they have in their inventory slots. Currently we are using placeholder artwork that has been attained as part of an asset pack. In further development we hope to create personally create artwork. The following screens are an example of sprites used, following the link will take you to our GitHub repo where all sprites can be seen the following is a link for the UI. https://github.com/Channon87/2dFarms/tree/main/2dfarms/Assets/Sprites/UI

Item Description	**Concept Art**
Various tools that will be used by player.	
Toolbar showcasing top twelve items in player inventory	
Full player inventory displaying all thirty-six slots.	

Author: Channon Harper s3871491 | Jessica Cramb s3813728

Create Date: 19 April 2021

Core Feature 5: Growing and Harvesting

Design

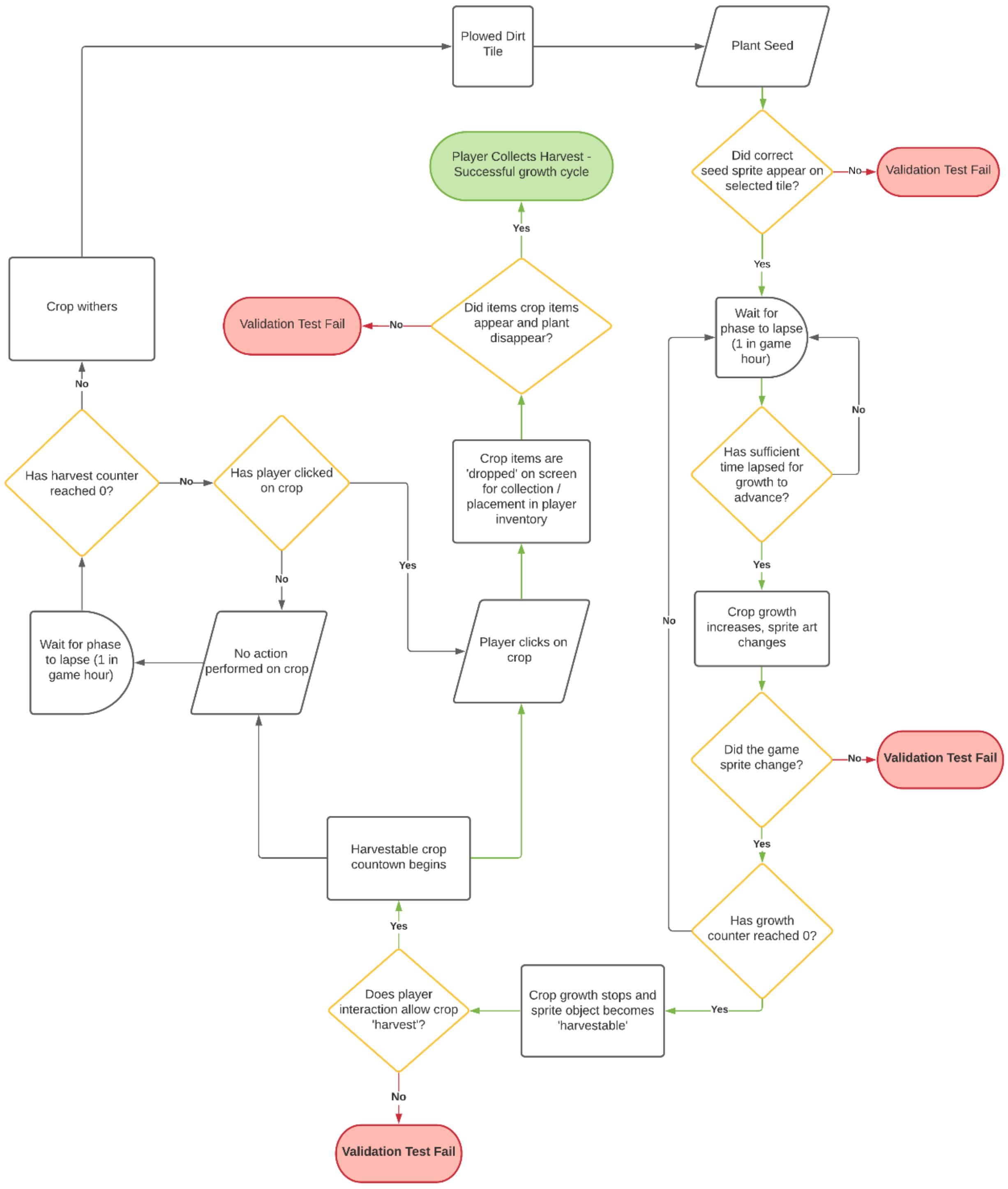
The core aspect and primary gameplay of *2D Farms – Live the Life* is the cultivation and sale of crops, including both food and raw materials e.g. cotton, bamboo. The ultimate goal of this game will be for the player to pay off the character's mortgage and eventually turn a profit from their sales. As such, the growing and harvesting features of the game are imperative to the success of a gameplay for *2D Farms – Live the Life*. The feature covers a range of gameplay aspects, such as:

- Preparing the soil for planting e.g., using a hoe to cultivate the soil.
- Planting seeds
- Caring for the crops; water and/or fertilize
- Harvesting

Validation Testing

There are several aspects to that must work for this function to be an overall test; as shown in this chart, the growth function is heavily dependent on the changing of sprites to show that crop growth is occurring. The following table and flowchart represents the overall process of a growth cycle from planting to harvest. This shows a general overview of the points in the process where a validation test will be performed.

Ideal Outcome	**Fail Indicators**
The crop art changes at specified times to give the appearance of growth.	a. Sprite artwork does not change at all. b. Artwork is incorrect for growth stage.
Growth stage options for player interaction change depending on the stage of the crop growth e.g., water, fertilize	a. No interaction options appear at all. b. Incorrect options made available e.g., option to harvest crop before it reaches the 'mature' growth stage. c. Option selected does not perform correct action. d. Options appear too early or too late in growth cycle
Harvest option for player interaction is only one available once growth counter reaches 0	a. Options for growth cycle still appear. b. Harvest option does not appear at all. c. Selecting 'harvest' does not perform action as anticipated.
Harvesting a crop removes plant sprite and inserts fruit/vegetable sprite	a. Crop sprite remains on screen. b. No fruit/vegetable sprites appear on screen.
When harvested crop is picked up the player inventory is increased correctly	a. Inventory is not adjusted to include new items. b. Items remain on screen as 'dropped'


Plowed Dirt Tile
Plant Seed
Did correct seed sprite appear on selected tile?
No
Validation Test Fail
Yes
Wait for phase to lapse (1 in game hour)
Has sufficient time lapsed for growth to advance?
No
Yes
Crop growth increases, sprite art changes
Did the game sprite change?
No
Validation Test Fail
Yes
Has growth counter reached 0?
No
Yes
Crop growth stops and sprite object becomes 'harvestable'
Does player interaction allow crop 'harvest'?
No
Validation Test Fail
Yes
Harvestable crop countown begins
Player clicks on crop
Crop items are 'dropped' on screen for collection / placement in player inventory
Did items crop items appear and plant disappear?
No
Validation Test Fail
Yes
Player Collects Harvest - Successful growth cycle
No action performed on crop
Wait for phase to lapse (1 in game hour)
Has harvest counter reached 0?
No
Has player clicked on crop
Yes
No
No
Crop withers

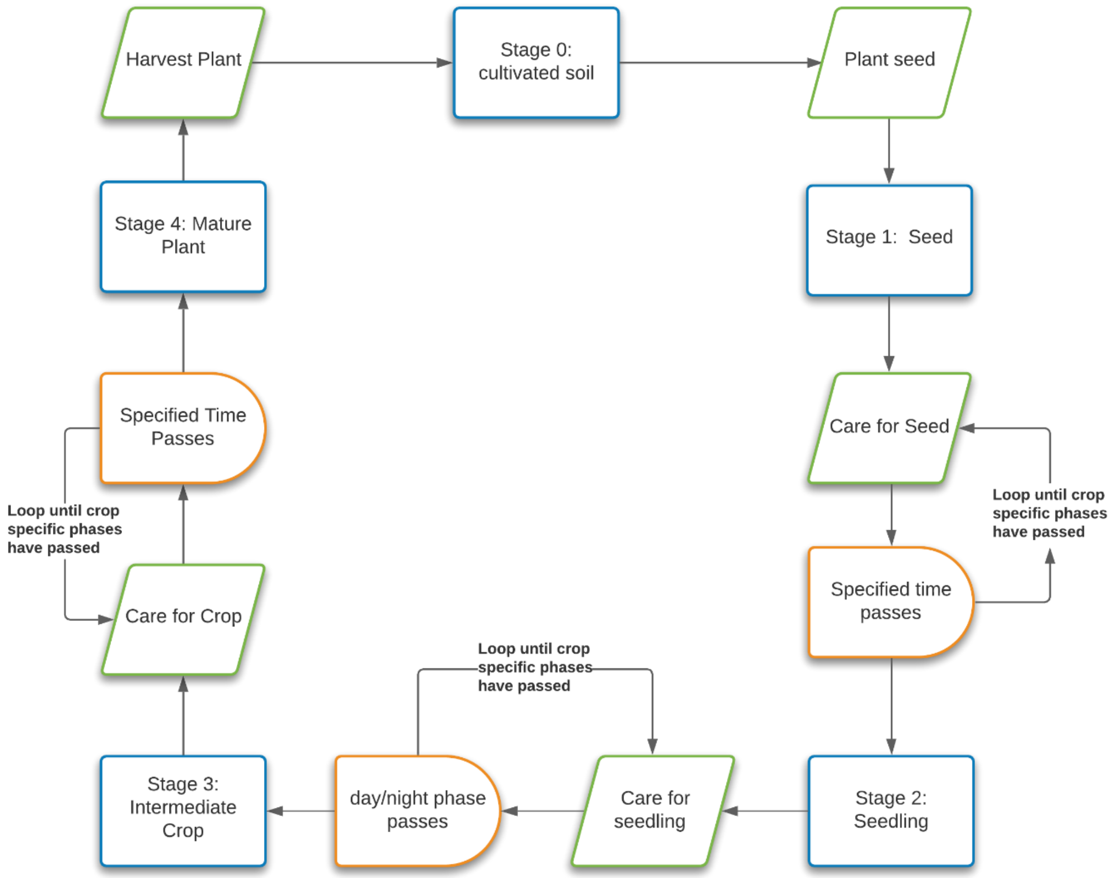
Engineering

Early concept ideas of *2D Farms – Live the Life* established that the game would, at its core, being a farming simulator game. As such, the player would spend, hopefully, hours of gameplay cycling through the stages of crop growth as mentioned above. In this section we will take a closer look at the game mechanics and how we visualise the final game to perform with regards to growing and harvesting gameplay.

To give a level of urgency around crop cultivation there will be a cycle that each crop will go that will include planting, growth windows and harvest windows. To view the growth cycle in action please visit https://channon87.github.io/2dFarms/#MVF5. Specific mechanics for the growth and harvest function will include the following:

- Progression through growth cycle will rely on the in-game timer. Once the growth counter has reached 0 another counter will begin.
- Progression from mature plant to wilted plant will be tied to in game timer. If the harvest counter reaches 0, the crop will no longer be available for harvest and instead the player must replant the field with a new crop.
- We will set rules around how much in game time should pass before progressing to the next stage of the growth and harvest cycles, respectively.
- Artwork/sprites will change for each stage of growth.
- Amount of produce awards for harvesting mature crops will be set individually for each crop type.
- Crop care in game objects that will increase growth counter by set multiplier, such as fertilizers and watering equipment.
- Following further development on some EVFs, there will be progress indicators for each crop to indicate stage of growth or time left to harvest. This will be included as a UI in the corner of the screen if the player hovers over a crop.

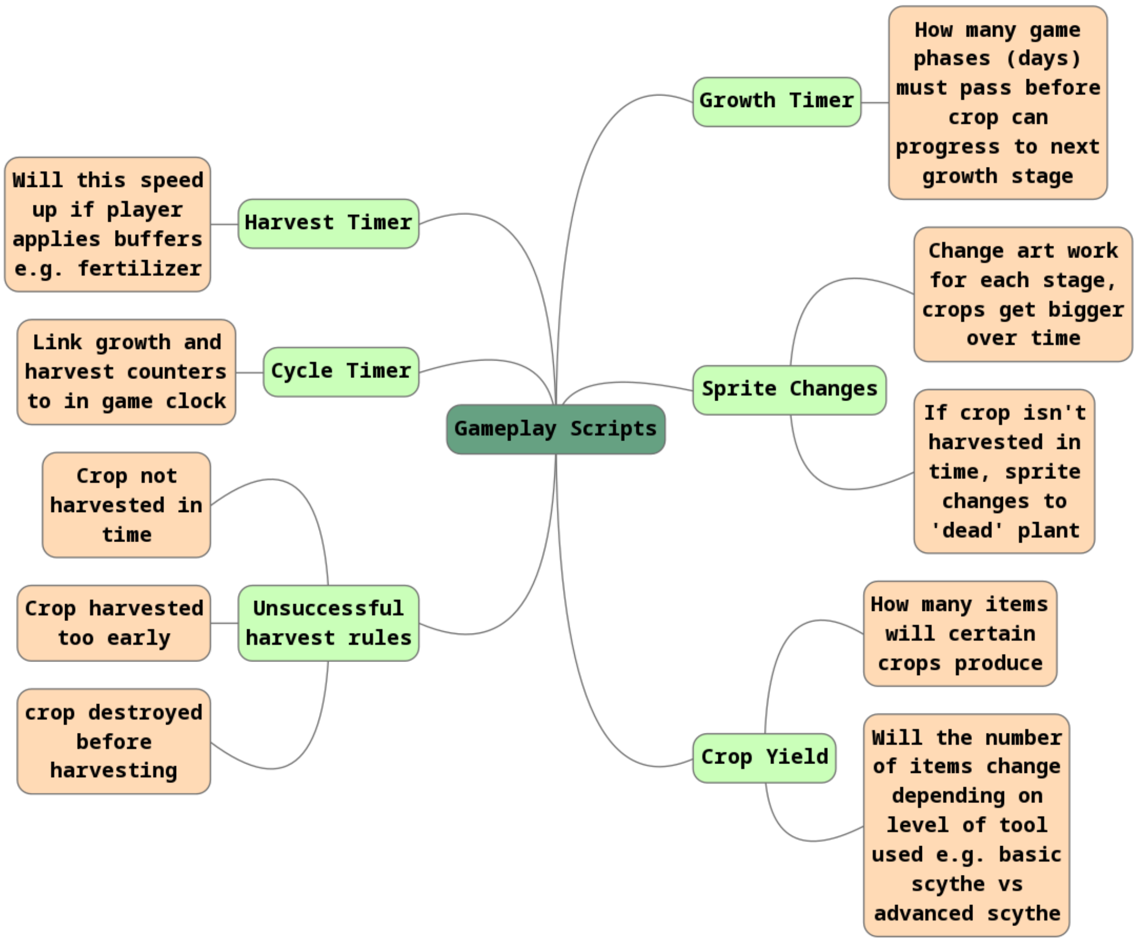
The following flowchart provides a vary broad overview of a successful growth and harvest cycle.



Production

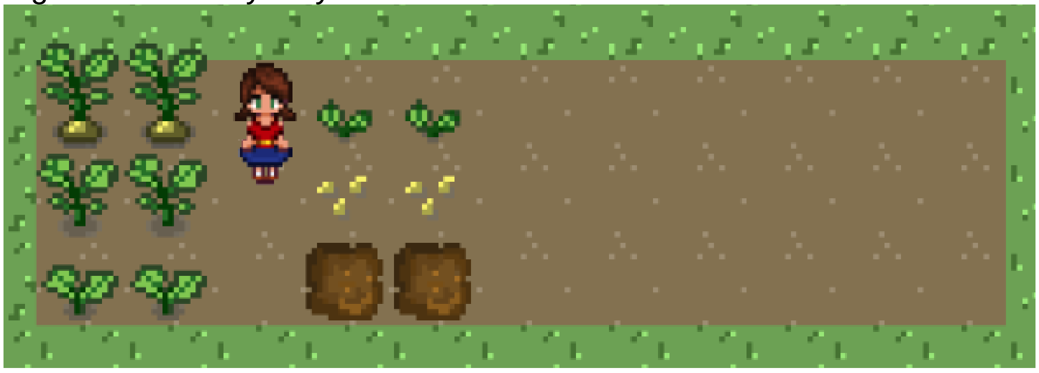
Development of the growing and harvesting feature will require the use of a number of tools. Unity will be the primary tool as this provide a graphic interface hub to help pull together all aspects of the game function and associated objects. Visual Studio will allow us to develop scripts for the gameplay, this is the default software used by Unity. Lastly, sprite creation and modification will be performed using GNU Image Manipulation Program (GIMP). Scripts for this feature will include aspects such as:

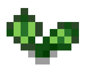
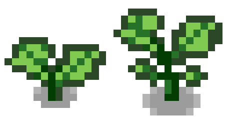
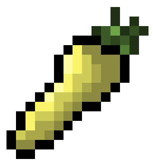
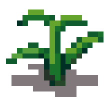
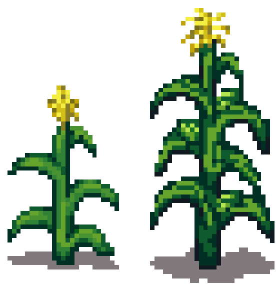
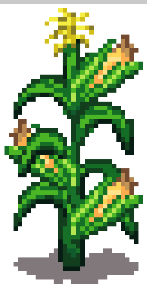
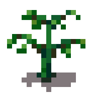
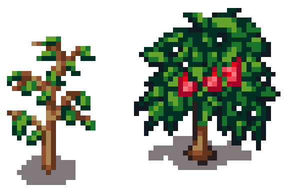
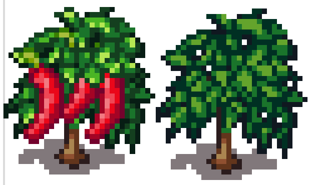
- Growth time rules: how many game phases (days) will it take for a crop to progress to the next stage of growth.
- Harvest time rules: how many game phases (days) will the crop be available for harvest before it is deemed 'dead'.
- Sprite changes: each growth and/or harvest stage will have an assigned sprite artwork to give the illusion of crop growth e.g., when crop changes from 'seed stage' to 'seedling stage' the sprite will change from small to medium artwork.
- Cycle timer: instruction on how to draw on game timer to progress growth and harvest timer.
- Crop yield: determines how many items are 'dropped' per crop type when harvested.
- Non-mature harvest rules set rules for what happens to a crop should it be harvested or destroyed before reaching the mature stage of growth e.g., crop destroyed with no yield.


Gameplay Scripts
Growth Timer
How many game phases (days) must pass before crop can progress to next growth stage
Harvest Timer
Will this speed up if player applies buffers e.g. fertilizer
Cycle Timer
Link growth and harvest counters to in game clock
Sprite Changes
Change art work for each stage, crops get bigger over time
If crop isn't harvested in time, sprite changes to 'dead' plant
Unsuccessful harvest rules
Crop not harvested in time
Crop harvested too early
crop destroyed before harvesting
Crop Yield
How many items will certain crops produce
Will the number of items change depending on level of tool used e.g. basic scythe vs advanced scythe

Art

Crops each have stages they go through, seedling, intermediate, mature and expired. As time goes forward the crops transition between these states never going back to the last. Crops will only drop produce when harvested in the 'mature' state. The amount of items dropped will be specified per crop and possibly depend on tool used by player e.g. advanced scythe yields more wheat.



Crop Name	Seedling	Intermediate	Mature	Expired	Harvested Item
Parsnip				Coming Soon	
Sweetcorn				Coming Soon	Coming Soon
Chilli				Coming Soon	Coming Soon

Author: Jessica Cramb s3813728

Create Date: 19 April 2021

Project Estimation

Group Estimation Summary

	Clark Lavery	Channon Harper (TL)	Jessica Cramb	Joseph Tselios	Jacob King	Mathew Smith	**{Average Hour}**
MVF 1 - Player Movement	20.00	9.00	10.00	10.00	9.00	20.00	13
MVF 2 - Item Interaction	35.00	28.00	26.00	30.00	27.00	35.00	30
MVF 3 - Time Lapse	20.00	11.00	20.00	15.00	10.00	20.00	16
MVF 4 - User Interface	30.00	31.00	29.00	35.00	28.00	30.00	31
MVF 5 - Growing and Harvesting	30.00	26.00	34.00	25.00	29.00	30.00	29
EVF 1 - Levelling System	25.00	29.00	36.00	30.00	32.00	25.00	30
EVF 2 - Unlockable Items	25.00	34.00	36.00	37.00	32.00	25.00	32
EVF 3 - Menu	15.00	14.00	20.00	20.00	21.00	15.00	17
EVF 4 - Game Fail	40.00	60.00	35.00	55.00	32.00	40.00	44
						Total	240

	Research	Learning	Designing	Coding	Testing	Total Hours
MVF 1 - Player Movement	2.00	1.00	2.00	2.00	2.00	9.00
MVF 2 - Item Interaction	2.00	5.00	3.00	10.00	8.00	28.00
MVF 3 - Time Lapse	2.00	2.00	1.00	4.00	2.00	11.00
MVF 4 - User Interface	4.00	5.00	4.00	12.00	6.00	31.00
MVF 5 - Growing and Harvesting	4.00	4.00	2.00	12.00	4.00	26.00
Extended Feature 1 - Levelling System	4.00	4.00	3.00	12.00	6.00	29.00
Extended Feature 2 - Unlockable Items	4.00	4.00	6.00	10.00	10.00	34.00
Extended Feature 3 - Menu	2.00	2.00	4.00	4.00	2.00	14.00
Extended Feature 4 - Game Fail	6.00	4.00	10.00	20.00	20.00	60.00

Justification description	
MVF 1 - Player Movement	For this item I chose that the research and learning was minimal as in our project the movement is basic. Which in turn the designing coding is also minimal as the coding is readily available and many tutorials exist. As for testing there may be some circumstances where it may not work however the testing of simple player movement up down left right with the ability to sprint should be easily testable and any problems would be incurred by a further feature. Please note that for this feature and the rest that follow these are done on the principal that I have zero previous experience with unity or with C# however by the looks of it C# relates quite well with Java which I do have experience with. At writing these estimations I had already done research into each one giving me a better understanding as to what was needed.
MVF 2 - Item Interaction	For this item research as to how interaction should be done should be relatively easy. Learning how to do it on the other hand will take longer hence the extended time. Designing will take an average time as we decide as to what kind of interaction we wish to incur. Coding and testing have a longer time as I believe the process of making multiple scripts to work together will take a lot of know-how and if a test fails to work extra coding will take place. Since I have never tried to make a game with item interaction the learning curve and implementation is higher than others and thus resulting in these figures.

MVF 3 - Time Lapse	Research, learning and designing on this is minimal as I see that there would not be extended ways to create a timescale within the game we are making. Coding and testing may take a bit longer but currently believe it would only need 1 script that may need some working to make it how we want it so will a bit of playing around with the testing and coding.
MVF 4 - User Interface	Research, learning and design of this aspect have an average time as there are multiple ways to introduce a UI to the system, I have never made a UI system for a game and hence the numbers I have given. Coding and testing are also very extended times are there would be multiple script to write alongside many things that could possibly go wrong with the system, therefore more time in coding after testing procedure. Since the UI also has many different windows available to display multiple different aspects this will also extend the time needed to implement this.
MVF 5 - Growing and Harvesting	Research, learning, and design have an average time as there may be a couple ways to implement this feature but once we found one that suites our game design should not take as long as the research. Coding and testing on the other hand may take time as there would be multiple ways to implement a scale of growth as well as multiple different plan table with different items to harvest the times, I given I believe to be on the low side for simple basics of 1 plan table. Testing the feature is lower than the coding as can scale time and see if the coding works however multiple times the coding may need to be changed to produce the desired outcome.
Extended Feature 1 - Levelling System	As before research learning and designing are of average length as finding the system, we wish to use may be difficult however once found designing of that system should be relatively easy. Coding and testing on the other hand would be difficult as each previously made script would need to have an experience additive added to it as well as testing to make sure each of these items produce the same coding. These numbers may be way off and may require a lot more time however basing off the system of minimal items to make the game playable.
Extended Feature 2 - Unlockable Items	research and learning should take an average time as there are multiple tutorials available for this, however it will be a learning curve has said previously I have no experience in game development. designing has a high time as once we have a method to unlock items, we then need to design a flow with the levelling system as to what items will unlock at certain levels and this will take a lot of time. Coding and testing are also relatively high as would need to play game through to see if testing worked, and the coding would also need to be done through multiple scripts assuming making it a laborious task.
Extended Feature 3 - Menu	Unlike the other I believe that research learning and development of this system is minimal I do not believe there are many ways to implement a menu system and any of them should work with our game. Coding and testing are also low as from my current knowledge of videos I have watched it is not a hard task and to add an extra scene through unity pausing time is quite doable testing as always half time of coding as problems may incur.

Extended feature 4 - Game Fail	Probably the most extensive feature we have in our system research of this would take time as to find the right coding that would suite or game. Learning is medium as I believe once we find the right tutorial can be learnt in this time frame. Designing will take a lot of time as the game fail system is huge, we would need to maths a lot including how much a bank takes off and how much each crop makes so the game is playable but not too easy. Coding will also be over extensive as the game fail itself is a multitude of features rolled into one the timeframe given is to get the basics working as a level one player would require a lot more work as levels progress. Testing will also take a long period of time as it will require gameplay for testing and with each level to money aspect even with a sped-up timer may take upwards of 1 hour to complete.

Jessica Cramb

	Research	Learning	Designing	Coding	Testing	Total Hours
MVF 1 - Player Movement	2.00	2.00	2.00	3.00	1.00	10
MVF 2 - Item Interaction	3.00	3.00	5.00	10.00	5.00	26
MVF 3 - Time Lapse	3.00	3.00	2.00	8.00	4.00	20
MVF 4 - User Interface	5.00	5.00	7.00	7.00	5.00	29
MVF 5 - Growing and Harvesting	7.00	7.00	5.00	10.00	5.00	34
EVF 1 - Levelling System	7.00	7.00	6.00	10.00	6.00	36
EVF 2 - Unlockable Items	5.00	5.00	10.00	10.00	6.00	36
EVF 3 – Menu	3.00	3.00	5.00	6.00	3.00	20
EVF 4 - Game Fail	8.00	6.00	6.00	10.00	5.00	35

Justification description	
MVF 1 - Player Movement	Character movement for gameplay is such a core aspect to all rpg games, meaning that the functionality is tried and tested. As such I believe that the overall time to complete this feature to be minimal as there was bound be a vast number of tutorials freely available, meaning that there was a good chance a refined, quick process for player movement would be available.
MVF 2 - Item Interaction	I did not have any experience with game development prior to starting this subject, although have put in countless hours of game play. With that said I know from a player perspective that item interaction in game can intricate depending on the level of detail and game objectives. Since we are developing a game that will have many interactive objects, and each with their own set of rules I anticipate that development on this feature would be intensive. I have nominated most of the time to coding as I imagine that getting the code right will be an iterative process that could take a while, given my lack of knowledge in this area.
MVF 3 - Time Lapse	I have allocated a medium amount of time for research and learning to this feature, as I have not had experience in developing a feature like this but did have a few ideas in mind on how the coding could work by applying analytical thinking during the concept stage. I have dedicated the bulk of time allocation to coding as I imagine there would be a lot of working parts involved in this feature e.g., in game timer, colour hue changes in game to depict different times of the day, like a blue tint for night.
MVF 4 - User Interface	I have spread the time allocation for this feature fairly evenly as I believe that developing a user interface will be a highly iterative project and so we will most likely find ourselves cycling through all phases quite often.

MVF 5 - Growing and Harvesting	This feature has more time allocated to it and the bulk of this is in coding. I have gone with a higher figure as I envisage this game having a decent variation in crops and so having to include these will take quite a while form art to coding.
Extended Feature 1 - Levelling System	Including a levelling system is a daunting prospect for me and having no experience in these matters I have given this EVF a higher time allocation for research and learning, in addition to coding. I imagine this will require a fair amount of intricate coding.
Extended Feature 2 - Unlockable Items	I have dedicated a large portion of this EVF to designing and coding as this will involve the development of a range of ‘tools’ for game play. Between the design and coding this would be the bulk of the work.
Extended Feature 3 - Menu	I believe that the development of a menu would be too intense as this is a component used by all games, as such there is bound to be a number of informative tutorials.
Extended Feature 4 - Game Fail	The game fail feature will require the development of a thorough script that looks for a range of variables and so I have estimated a longer time for completion of this EVF. The time needed to research would take longer as a result of the various moving parts.

	Research	Learning	Designing	Coding	Testing	Total Hours
MVF 1 - Player Movement	3.00	1.00	2.00	2.00	2.00	10.00
MVF 2 - Item Interaction	2.00	5.00	3.00	10.00	10.00	30.00
MVF 3 - Time Lapse	6.00	2.00	1.00	4.00	2.00	15.00
MVF 4 - User Interface	10.00	3.00	2.00	10.00	10.00	35.00
MVF 5 - Growing and Harvesting	2.00	2.00	1.00	15.00	5.00	25.00
Extended Feature 1 - Levelling System	5.00	5.00	3.00	12.00	5.00	30.00
Extended Feature 2 - Unlockable Items	4.00	3.00	10.00	10.00	10.00	37.00
Extended Feature 3 - Menu	5.00	5.00	4.00	4.00	2.00	20.00
Extended Feature 4 - Game Fail	5.00	5.00	5.00	20.00	20.00	55.00

Justification description	
MVF 1 - Player Movement	I estimated 10 hours as I believe a lot of the coding for movement is already created in unity projects.
MVF 2 - Item Interaction	Item interaction can be very detailed depending on how much effort is input. I put enough time allowed to really take the time to learn which codes will need to be implemented and allow time to find the correct images to act as the item
MVF 3 - Time Lapse	Time lapse is a concept that I have never fiddled around with, I do know thanks to Channon that the concept is straight forward when implemented, just learning how to implement is the tricky part
MVF 4 - User Interface	Surprisingly enough, I am a perfectionist with these types of things, user interface is such an important part in games since players can be overwhelmed very easily and give up on a game if the interface is too confusing, I dedicated enough time to this part of the game as I believe it is such an important part.
MVF 5 - Growing and Harvesting	Growing and Harvesting is quite linked to item interaction, we will just need to link the script to the time lapse feature to really give the real time feel to the game.
Extended Feature 1 - Levelling System	The leveling system is one that I have never found myself working on, the learning curve for me will be very large, a lot of research will need to be put in for me to even understand where to start.
Extended Feature 2 - Unlockable Items	Again, I have never found myself working on a feature where items are locked away from the user until they get something or do something to unlock the item. Another feature that sounds good in theory, but I will need to research a fair bit on how to implement this feature.
Extended Feature 3 - Menu	I have estimated the menu to take 20:00 hours as I believe it is another important part of user interface. Having clarity and simplicity is key, but it still needs to be engaging with the user, a fine balance.

Extended Feature 4 - Game Fail	Finding fixes to game fails is a lengthy process, whether it be finding out why a code is not working as intended or how to implement a new feature that for some reason is not working correctly for an unknown reason, all game fails can be lengthy when troubleshooting.

	Research	Learning	Designing	Coding	Testing	Total Hours
MVF 1 - Player Movement	2.00	2.00	1.00	3.00	1.00	9.00
MVF 2 - Item Interaction	3.00	5.00	5.00	12.00	2.00	27.00
MVF 3 - Time Lapse	2.00	2.00	2.00	3.00	1.00	10.00
MVF 4 - User Interface	4.00	4.00	5.00	11.00	4.00	28.00
MVF 5 - Growing and Harvesting	4.00	4.00	3.00	12.00	6.00	29.00
Extended Feature 1 - Levelling System	6.00	3.00	5.00	12.00	6.00	32.00
Extended Feature 2 - Unlockable Items	6.00	3.00	5.00	12.00	6.00	32.00
Extended Feature 3 - Menu	3.00	3.00	5.00	4.00	2.00	17.00
Extended Feature 4 - Game Fail	6.00	6.00	8.00	8.00	4.00	32.00

Justification description	
MVF 1 - Player Movement	Player movement is a simple mechanic. Most games have this feature, so researching will not take much time. Tutorials are easy to find and there are so many of them, and this feature is so old in gaming, they will be concise and simple to follow. Time to complete this feature will not require much in the way of testing, as it is very easy to do so.
MVF 2 - Item Interaction	Researching another staple of gaming will not take much time but learning how it all works together will take more time. Design will take a similar amount of time to get it just right. Getting the mechanics of this feature how we want will take some design time. Coding will by far take the longest. This is just to get the base of this feature working and tweaking the system. Testing will not take much time because of how simple the feature really is. Fine tuning it will take the most time hence the longer 'coding' section.
MVF 3 - Time Lapse	Time lapse feature is a simple feature to have in a game, making in-game time run faster than the real world does to ensure a better gameplay experience. Luckily for us, this is also a common feature in videogames. Finding tutorials and researching the feature will not take up much time because of how ubiquitous a feature it is. The design of this feature will not take long as it is a simple calculation of how fast we want the game to be. Coding will take the longest to complete because of how many smaller interactions this feature is linked with. Testing will not take much time – just seeing if time flows faster in game compared to the real world.

MVF 4 - User Interface	Researching User Interface will take longer than the features mentioned above – but still not very long. User Interface is also a feature in every game so finding how to make one will not take long. Neither will learning how to make one. Design will take more time because of how ‘arty’ this process is. Coding the User Interface will take much longer because of the intricacies of how it functions. This feature interacts with lots of others in game. Multiple scripts will have to work together to make this feature function properly. Testing will not take a long time because the design is straight forward – if a button does not work as designed to, then it needs to be fixed. Finding what is wrong is the easy part of this, fixing it will take longer than identifying issues.
MVF 5 - Growing and Harvesting	Researching and learning how to make Growable and Harvestable in game will not take much time as it is in so many other games. Finding tutorials and following along with them will be straight forward because of this common game feature. Design will also no take very long because of the rather simple desired functionality of this feature. Coding will take far longer than the other parts because of how many systems this feature needs to interact with. Many scripts will need to work together to function. Testing this is also take time because of the same reasons. If something is not working properly it could be because of the other features and scripts not working nicely together. Finding what is wrong with this feature will be time consuming.
Extended Feature 1 - Levelling System	Researching a leveling feature will take time to do so. This is not a unique feature to gaming but researching how it works will take time. Learning how this feature works will not take as much time because it is a simple enough concept to grasp. Design will take a long time because of the way it should be integrated into the game, there are many ways to do so and doing it well will take time. Coding this feature depends on how complicated the design is. Luckily, this feature does not rely too heavily on the other features. Testing this feature can take time also because of the complexity of the design. If it is hard to achieve in-game, it will take longer to test.
Extended Feature 2 - Unlockable Items	Unlockable Items will take little time to research and learn, as tutorials are readily available online and the concept is simple enough to grasp. These two factors lower time needed for research and learning. Design will take more time to design because of the complexity of the system we want. Coding and testing will take a moderate amount of time. Coding because of the features needed, and all the different unlocks to code in. Testing will take a long time because of the gameplay time required.
Extended Feature 3 - Menu	The menu system is similar to some features we have already. Researching and Learning will not take long as it is essentially coding in buttons that do different things. Design will take the longest because deciding how each sub menu interacts with the game will take a long time. Coding will not take a very long time because most of the script will be repetitive. Testing will be easy to do and straight forward – so little time is needed.
Extended Feature 4 - Game Fail	This extended feature will take a long time overall. Researching and learning this system will take time to get done properly, as this system will need to be tailored to the game. Design and coding will take even longer because of the fine tuning needed to get right. Testing will only take this long because of the play time needed to test the system.

Matthew Smith

Given the following justification, values within the group estimation have been worked off Clark's figures. Since our mentor has more experience than any of us in game development the figures given would be more accurate than the ones group members had come up with therefore the secondary column helps to determine the actual average timeframe required to complete these features.

Justification Description

As I have just joined this group and am unfamiliar with the progress of the development, I cannot answer these questions with meaningful values. Any attempt to hypothesis the outcomes of unknown tasks would only serve to hide my ignorance at this point. This ignorance has been imposed by the late joining of the group, and other time commitments in the in the three (3) days since I joined the group.

Collaborative workspaces

Discord -

Discord is an easy program to use that can make communicating amongst the team reliable and consistent. There is a mobile version that will allow people to communicate while they are away from home or their PC. Our team uses discord for all direct communications, we opted for this option as it allows easier notification and scroll back for any missed messages. Can also directly message any team member if its talk not needing for the entire group to see, i.e., a question or query about something the other person is working on. The following link shows how we have used discord so far in the project although not all communication is shown here as we have separate channels this is for indication purposes only.
https://channon87.github.io/2dFarms/video/discordGrab.html

Microsoft Teams

Microsoft Teams is a more professional platform that we will be using for communication and documentation. Since it has a direct link to our Trello board and can have shared documents that any user can access and collaborate on it makes deliberation on items easy. We also use it for all meeting we hold including group and tutor, we can record each meeting which is then able to be viewed later for any participants that did not attend. We use a private channel that cannot be viewed unless authorized so a link will be lacking in this regard.

Trello

Trello allows the group to clearly communicate exactly what needs to be done, when it needs to be completed and who has dedicated themselves to completing ‘x’ task. So far, our use of Trello is related back to our meetings with what is expected for each team member to achieve within a week. Each member then adds these tasks to their personal swim lane so everyone can see the progress. The other section of Trello is dedicated to the allocation of main tasks; team members are expected to join a card to indicate that they will work on it . Each of these have checklist that need to be done prior to moving it into the done section, we still have some work to do on our board to neaten it up however currently the system we use is working for us.

Github

Access to Github is free for all users, however as students of RMIT we have access to the student pack. This has a web application alongside a desktop app known as GitHub desktop. This will be used for the project creation itself. Currently we use GitHub as a backup of the game development alongside using it to keep all created sprites and a website that allows people to see artefacts of the game itself. Collaboration of the board is not set yet since currently one person takes care of the system backups and web design and although not included within the main resources of this project as the website is not a foregone necessity I choose to use Atom for html coding. You will find it here:
https://github.com/Channon87/2dFarms

Software

Unity

The version we are using is 2020.3.1f1 and with the benefits of being a student at RMIT the professional version is free for 4 years. Alternatives would be Godot Engine or GameMaker Studio 2. Each one has its perks however a decision was made for unity since it seemed to have the most tutorials and since none of us had used unity before seemed to be the easiest to learn. After using it now for 6 weeks the project progress is very good, with its ability to collaborate through their servers and roll back the game on big mess ups have not ran into any problems with it yet. When choosing a game engine for the future I believe that unity is where I will start again. As previously input into the document links to video artefacts, even after a brief period it seems to be able to do everything we intend for our project.

GIMP

Gimp or better known as a free/poor man's photoshop, we decided on this tool for that reason it is free. With creating a 2D game there are a lot of sprites and artworks that need to be made, with gimp we can make our 16x16 tile sprites and add in alpha channels for the transparent background with ease. Couple this up with a graphics tablet and you have an artwork studio to create anything your head desires. So far some of the asset sprites used to make the game have not had a transparent background or shadows which we did not want so using Gimp we had the ability to crop and change these to our intention.

Microsoft Visual Studio

Although other tools for coding in C# may be available MVS is attached to unity engine and offered on initial download. Given that they have features that work together the decision was easy to use this program. Our game being 2D does not have assets readily available that includes scripting, so all the scripting had to be done manually. I found that it much like many other tools like Atom have intelligent autocomplete (sometimes annoying) and the features within were easy to use once a script was saved it would automatically update unity to show this.

Tools

Lucid Charts

Like many successful projects planning is important and to have this in a visual manner is much desired. Lucid Charts allows us to map out our features with failures giving a rundown of how each of them should work in the end. So far, we are using Lucid Charts for all diagrams related to the project, future use will include script designing for the programming side of things allowing easy knowledge of class design and how each variable relates to one another.

YouTube

Probably the go to system of learning things you do not know apart from google. YouTube has provided many tutorial videos on aspects of our game that has allowed us to learn the intricacies of C# as well as Unity design. If you want to learn something there is usually a video made for it finding the right one can, sometimes take time. Most of us being more visual learners found the videos more relatable than any text applicable websites available although some of these were helpful also.

Resources

These resources helped in our game with not only the production of our C# scripting and the unity development but with access to usable sprites and information on how one should go about game development. The following headers are all linkable to the relevant sites that they depict following this an explanation of said resource and how we found this useful.

Greg Dev Stuff

The playlist of making a Stardew Valley like game in unity was very informative helping with the script workings and how to make a farming game. There are limited farming game videos that give a rundown as good as this one. The fact that he covers all the aspects that we wanted to add to our game with scripts working together to achieve this, was not the easiest channel to find but when we did it was like a diamond in the rough.

Greg., n.d YouTube – Greg Dev Stuff [online] Available at: <*https://www.youtube.com/c/GregDevStuff/playlists* > [Accessed 24 April 2021].

Brackeys

Anything you need to know about unity there is probably a video in here somewhere Brackeys channel is helpful for all things unity. Although not producing any more videos and some of the tutorials being outdated a lot of what is taught is still relevant. Moving forward creating the EVF's this will be a resource we will depend on for learning.

Thirslund, A., n.d. YouTube - Brackeys Game Dev Tutorials*. [online] Youtube.com. Available at:* <*https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA/playlists*> *[Accessed 24 April 2021].*

University of Washington GAMER GROUP

The best information we could find on how to develop artefacts for game design was helpful as it broke down how to develop a game in good detail. From start to finish it covers all aspects of what is expected throughout the development process.

Cpb-us-e1.wpmucdn.com. 2021. [online] Available at: <https://cpb-us-e1.wpmucdn.com/sites.uw.edu/dist/2/3760/files/2019/09/Taxonomy-of-Video-Game-Development-Artifacts-.pdf> [Accessed 24 April 2021].

Unity Learn Tutorials

While using their program they provide a lot of tutorials themselves. Who better to learn the basics of unity then the people that made the development software themselves? Within their tutorials page you can find a multitude of videos ranging from beginner to advanced, this was our first steppingstone into learning how to develop our project.

Unity Learn. 2021. Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn. *[online] Available at:* <https://learn.unity.com/tutorials> *[Accessed 24 April 2021].*

Unity asset store

Starting a project there are many areas to cover, although looked down upon on a release of the package assets are proven valuable as any starting point to a project. Within the asset store they offer many assets however for our project we opted for the obvious free ones that gave us some sprite to start with, so we did not have to worry about creating concept art which takes time prior to getting into the coding.

Unity *2021.* Unity Asset Store *[online] Available at:* <https://assetstore.unity.com/> *[Accessed 24 April 2021].*

Open Game Art

Much like the previous stated asset store OpenGameArt offers sprites created by users to be used. There are many items within various games that can only really been drawn one way so why not take the opportunity to access these without the need to draw it yourself. Its all about time management in a project and having pre-done open-source sprites to be used helps get production rolling. Although once again upon game release it is expected that art is redone especially if it has a monetary value added to it, no-one likes an asset flip.

OpenGameArt.org. 2021. OpenGameArt.org. *[online] Available at:* <https://opengameart.org/> *[Accessed 24 April 2021].*

Author: Channon Harper / s3871491 **Create Date:** 23/04/2021