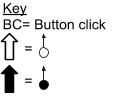
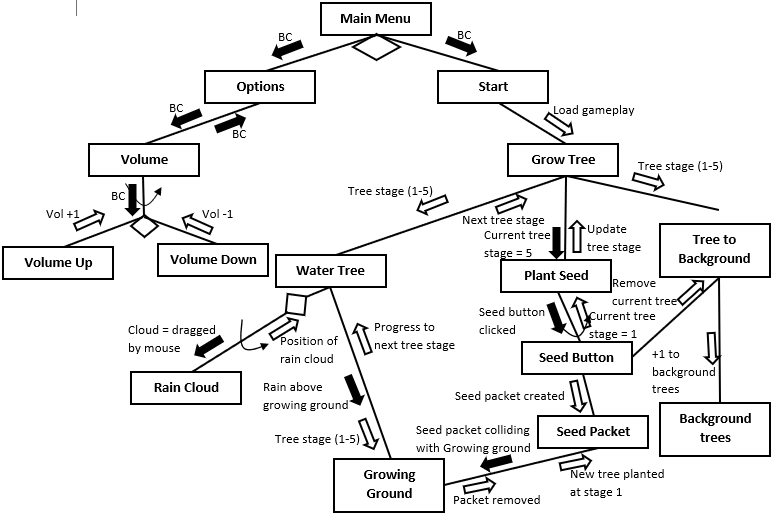
Data Dictionary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Format** | **Storage Size (bytes)** | **Display Size (bytes)** | **Description** | **Example** | **Validation** |
| Planted | Boolean | 0 1 | 2 | 1 | Used within GGPlanter animator to set to true when a seed is planted and set to false when animations are reset. | [True] | When set to true, allows for growing stages animations and coding to occur. |
| Finish | Boolean | 0 1 | 2 | 1 | Used within GGPlanter animator to set to true if SeedButton is pressed and set to false when a seed is planted. | [True] | When set to true, resets animations for the GGPlanter. |
| Deactivate | Boolean | 0 1 | 2 | 1 | Used within SeedButton animator to set to true if tree is not fully grown is pressed and set to false when it is. | [False] | When set to true, animates SeedButton to greyed out to display to the user that the button cannot be used. |
| Rain | Boolean | 0 1 | 2 | 1 | Used within Cloud animator to set to true if being held down and set to false when it is not. | [True] | When set to true, animates Cloud into storm cloud to display to the user it can rain. |
| OnMouseDown | Boolean | 0 1 | 2 | 1 | Used within multiple scripts to set to true when the mouse is clicked over the corresponding object/s. | [False] | Allows for functions (animations, code) within objects to occur when set to true. |
| OnMouseUp | Boolean | 0 1 | 2 | 1 | Used within multiple scripts to set to true when the mouse is released from the corresponding object/s. | [False] | Allows for functions (animations, code) within objects to occur when set to true. |
| OnMouseDrag | Boolean | 0 1 | 2 | 1 | Used within multiple scripts to set to true when the mouse drags the corresponding object/s. | [False] | Allows for functions (animations, code) within objects to occur when set to true. |
| OnClick | Boolean | 0 1 | 2 | 1 | Used within all buttons to set to true only when they are pressed. | [False] | Allows for functions (animations, code) within the button to occur when set to true without the need for other detection Booleans. |
| Water/WaterLvl | Static Integer | N | 2 | 2 | Created and updated within TreeGrow script, records how many times WaterDrop collides with GrowingGround and resets when SeedButton is pressed. | [15] | Allows NewSeed and TreeS (1, 2, 3, 4) scripts and the corresponding objects to access the water amounts in order to know when to update animations and coding. |
| Grow | Integer | N | 2 | 2 | Used within GGPlanter animator to hold a value equal to that of WatetLvl. | [15] | Updates tree stage based upon how high/low the number value is. |
| TreeScore | Integer | N | 2 | 2 | Created and updated within NewSeed script to hold the value of the number of times SeedButton is pressed. | [5] | Updates by 1 for every press, allowing for myText to understand the number of times a tree has been completed. |
| screenPoint | Array (Float) | N N N | 16 | 16 | Used within multiple scripts to record the value of the position of the object relative to vectors. | [134.5, -12.31, 74] | Allows for the objects position on the screen to be recorded in order to be referenced by other code. |
| Offset | Array (Float) | N N N | 16 | 16 | Used within multiple scripts to record the value of the movement of the object relative to their ScreenPoint and vectors. | [134.5, -12.31, 74] | Allows for an object to move to a position by recording the distance they must move from their ScrenPoint. |
| myText (UI text) | String | XXXX XXXXXXN | 22 | 26 | Used within NewSeed script to display and update the tree score on the UI using the TreeScore value. | [“Tree Score:0”] | Displays to the user the number of trees completed on the UI, in order to keep score. |

Structure Chart

IPO Chart

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Seed package (2D object), Growing ground collider | If the seed package collides with growing ground collider then run planting animation | Planting animation, current tree created at stage 1 at seed package location |
| Water drop (2D object), Growing ground collider, Current tree (2D object) | If water drop collides with growing ground collider and current tree exists then progress tree to next stage | Current tree grows to next stage, label appears, water splash animation |
| Seed button (UI element), Current tree stage | If seed button pressed and current tree is at stage 5 then run background tree animation and create new seed package | Seed package, background tree animation |
| Rain cloud (2D object) | If rain cloud pressed then run rain animation and rain cloud horizontally moves towards mouse location | Rain animation, rain cloud followed in mouse direction |
| Seed package (2D object),  Planting animation | If pressed and is not being animated then move with mouse | Seed package attached to mouse |
| Background trees right (2D object), background trees left (2D object), background tree animation | If background tree animation playing and background trees right != background trees left then add 1 to background trees right otherwise add 1 to background trees left | Background trees right or background trees left gained 1 tree |

Algorithms

BEGIN PlantSeed

IF Seedpackage = TouchingGG THEN

OUTPUT PlantingAnimation

Seedpackage = TreeStage(1)

LabelAnimation = 1

WaterLevel = 0

END IF

END PlantSeed

BEGIN WaterTree

INPUT LabelAnimation

INPUT TreeStage

INPUT GrowLevel

IF Waterdrop = TouchingGG THEN

WaterLevel + 1

OUTPUT WaterAnimation

REPEAT

CASEWHERE WaterLevel =

<3: WaterLevel = 3

3: TreeStage = 2; LabelAnimation = 2

6: TreeStage = 3; LabelAnimation = 3

9: TreeStage = 4; LabelAnimation = 4

12: TreeStage = 5; LabelAnimation = 5

ENDCASE

UNTIL TreeStage != <3

OUTPUT LabelAnimation

OUTPUT TreeStage

ENDIF

END WaterTree

BEGIN ButtonPress

INPUT TreeStage

IF SeedButton = Pressed THEN

IF TreeStage = 5 THEN

OUTPUT BackgroundTreeAnimation

OUTPUT NewSeedPackage

ENDIF

ENDIF

END SeedButton

BEGIN AddTree

INPUT BackgroundTreeAnimation

INPUT RightBackgroundTree

INPUT LeftBackgroundTree

IF BackgroundTreeAnimation = True THEN

IF RightBackgroundTree < LeftBackgroundTree THEN

RightBackgroundTree + 1

OTHERWISE

LeftBackgroundTree + 1

ENDIF

ENDIF

END AddTree

BEGIN Rain

INPUT RainCloud

INPUT Mouse

IF RainCloud = Pressed THEN

RainCloud y-Position = Mouse y-Position

OUTPUT RainAnimation

ENDIF

END Rain

BEGIN PackageMove

INPUT SeedPackage

INPUT Mouse

INPUT PlantingAnimation

IF SeedPackage = Pressed THEN

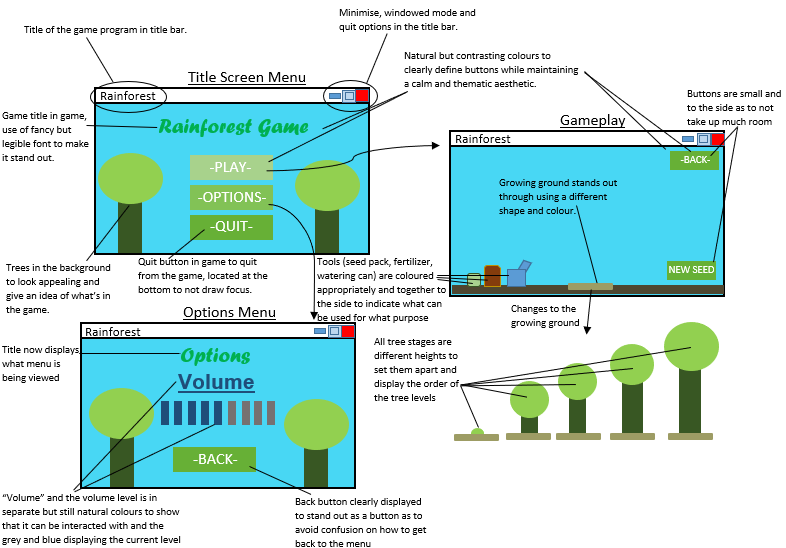
IF PlantingAnimation = 0 THEN

SeedPackage Position = Mouse Position

ENDIF

ENDIF

END PackageMove

Story Board (Miscellaneous Developments Tool)

Project Management Steps:

**Defining the problem**

The problem I have been faced with is the task to create an educational game targeted at a specific school grade. To solve this problem, I needed an audience and topic to base my game around, I chose rainforests as a topic and targeted year 8s. I chose year 8s as they study rainforests in both their science and geography subjects, meaning that my games educational content will be able to assist with their courses. I recorded the information regarding my games specifications goals for its future use as well as my aims for the content of the game. In doing so, I developed a competent understanding of what my game requirements are to be met, allowing me to create an initial Gantt chart. This initial Gantt chart was very simple, only displaying the 5 project development stages, without modules and with the timeline using the number of lessons. This Gantt chart therefor displayed which lessons would be used for which part of the process in order to plan and allocate time as at this point the goal was to only use lessons to complete the assignment.

**Planning**

The planning stage of my assignment consisted of the creation of pseudo code algorithms, an IPO chart, structure chart, story board, data dictionary and log. These methods of documentation allowed me to further develop my understanding of my solution to the problem by constructing a clear plan. While the majority of the planning stage did occur before I began the building stage, in order to use effective time management, I began the building stage before completing documentation I did not consider vital. The first thing I began was the log, as I started it in the stage of defining the problem in order to record the progress with decisions of topics, target audience, Gantt chart and brief. Then throughout the project I consistently updated my logs with the progress I make in order to understand my processes and to use in later documentation. My next step was to begin a storyboard that displayed the graphical interfaces, changes between scenes and the basic game progression. This story board allowed a clear plan for what my game content was going to be, allowing for me to then create an IPO chart that helped me to understand the processes to achieve my gameplay. In the process of creating my IPO chart, I decided to not use a fertilization bag as I decided that the use of water and seeds to grow a tree would be all that was necessary. From this point I used the IPO chart and story board to create a structure chart that displayed all interactions with the different components of my game to allowing me to understand its structures comprehensively. I decided I was ready to start writing my algorithms which I chose pseudo code for. As I was reaching the point that I needed to building my project, I decided to write algorithms and then implement them into my game before starting the next which allowed me to both make sure that my algorithms made sense and I could use them while I had a clear recent understanding of their purpose. I decided to create and update my Data dictionary as I created the fields within the game also to use my time to complete the aspects of my planning that were not vital to be done beforehand whilst creating my game to make sure it is done on time. By this time, I was working on aspects of the plan outside of class while creating my game inside of class to maximize my time efficiency.

**Building**

The building stage of my assignment consisted of the creation of sprites, scripts, animations and scenes to be used within Unity and took the greatest number of hours to complete. The initial parts of the building stage were creating the essential aspects of sprites for the background and algorithms. The sprites I made first were the background for the menu and the game, this allowed for the readjustment of cameras and the creations of scenes, allowing me to understand the scale of all other sprites as to make sure they would fit together on screen. This allowed me to then create the sprites for the buttons of the main menu so that I could place them on screen, by laying out the menu with sprites, it meant I was able to complete it quickly in order to focus on the game itself. When working on the gameplay scene, I began by creating the sprites for the growing ground and its different stages of being planted. In order to do this, I had to switch from MS paint to paint 3D, as MS paint did not support .png to create transparent backgrounds which had not been an issue for previous square sprites. This meant I had to learn new software for the sprites which was fairly simple and allowed me to continue on to finish the growing ground, tree stages and seed package. At this point I decided that without the bag of manure, it would be fair simpler and more intuitive to use a rain cloud to water the tree. This is because the cloud would be in the sky and the seed package would be on the ground, by separating the areas of use, it would reduce screen clutter and introduce a different aspect to the game. The cloud would introduce horizontal movement and procedural rain animation which would be more interesting to the player and better for the assignment as it displayed the ability to both re-use aspects of code as well as understanding code enough to edit it and use advanced methods for functions. The cloud and its storm state were the final sprites I completed. From this point, I added collisions for the ground (whose sprite was only part of the background), the growing ground and the seed package. This allowed me to create the movement scripts for the seed package, in order to learn the C#, I decided to use forums instead of YouTube videos as the forums had answers for more specific questions which allowed a deeper understanding. The seed package used its position and the position of the mouse to calculate how much it had to move, allowing it to follow the mouse when dragged and through the use of a simulated body, fell until it collided with the ground collider. Next, I wrote scripts for the growing ground to collide to change when colliding with the seed package to plant a seed through Unity’s animator, using Booleans to set values for collisions occurring. I implemented the cloud and recycled the script for the seed package but edited it so that the y-axis was unaffected by the mouse, meaning the cloud only moved horizontally, I also used a Boolean value to set it into a storm cloud state when dragged to show when it should be raining to the user. To create procedural rain animation, I created a separate rain entity from a capsule sprite and a capsule collider that had a simulated body so that it would fall. I then made the rain entity teleport to the cloud and lose velocity (so that it did not indefinitely pick-up speed) when it collided with the ground collider as well as deactivating and reactivating it depending on whether or not the cloud is being dragged/held by the user. I made it so that when the rain collides with the growing ground, an integer value for the water counter increased by 1, this then updated the grow integer in the growing ground animator, that when reaching 5, 10, 15 and 20, progressed the tree stage. The water counter only starts if the conditions of a seed planted are met, this is an example of a pre-test loop as it checked the condition before beginning the process. I then added sprites to the menu and gameplay buttons to change the scene on press and the seed button to remove the tree and add to a tree counter in the top left of the screen. The tree counter replaced the background trees so that the player still had some sort of incentive to continue playing the game. The seed button used a post-test loop to reset the tree as the process of tree growth happens before but it only has function once the tree is fully grown, otherwise the button can be pressed but it has no function, as indicated by its greyed out graphic. The final components I implemented was the labels to the tree which activated and deactivated depending on their correlating tree stage and while separate scripts, were the same with minor differences, then the options menu was changed to a how to play menu in order to instruct the user on the game.

**Checking**

The checking stage of my project consisted of desk checks, others testing the program and giving feedback and personally testing its limitations. The desk checks I performed were mainly on the growing integer code, to test if it all works with varying integers. I performed the tests to see the process of the animations working through the various inputs and if it all worked. The desk checks tested to see what would happen if negative numbers, border numbers and extremely large numbers worked appropriately. The tests all worked out the way they were supposed to, both in the desk check and on the program. This is because I used > to detect integer amounts instead of = as it means the number does not matter, as long as it is above an amount it will be detected by the program. The limitations I tested were of the collisions and the tree being reset. I tested the collisions with the seed package and cloud by attempting to move them out of the side bounds. As the ground collision goes slightly further than the screen, I could not drop the seed package over the edge, but I could push the cloud further enough that the water drop fell past the bounds and did not come back to the cloud automatically. This issue is not major though, as when the cloud is grabbed, it moves the drop back to its position, meaning that to get the drop back, all the user must do is let go and grab the cloud. During the creation of the seed button code, I had to repeatedly test whether or not resetting the tree caused any bugs. I discovered that when the tree was reset, I had it so that only the sprite reset and that the finish state and grow integer did not, meaning once the seed replanted, the tree was in an endless loop of moving through all its animations rapidly. The other problem I found was that when the labels were deactivated, their scripts did not activate, meaning they did not reactivate when the tree grew. When user testing occurred, while other users did not find any bugs or issues, it made me realize that the game was not color blind friendly. This is because while others could fairly clearly see the game title, I could not as its shade of green was difficult for me to see on the background. I did discover that the game was easy to understand and that the how to play screen worked effectively.

**Modifying**

My modifying stage consisted of implementing test feedback, fixing any bugs or problems found and the completion of my documentation and report. The test feedback I implemented was to change the color of the title to lighter green that I found much easier to read, making it color blind accessible. The bugs I found in the growth of the tree I fixed by resetting the finish Boolean and grow integer when the seed is replanted. This is done because when the tree is reset, it makes the finished bool true, and the planted bool false, so to be efficient and prevent any other unforeseen bugs, whenever the planted bool is set to true by the seed package colliding with the growing ground, the finished bool is automatically set to false. I fixed the labels by creating a control object for them that stays active while it holds all the scripts for the labels, the labels are then attached to the scripts in the object and it allows for the labels to appear with their corresponding tree stage. I completed my data dictionary once I had completed and finalized all aspects of my code as I had all data that I had used at that point. I also completed the final update on my Gantt chart, bringing it completely up to date and added my story board to the documentation. At this point I was able to comprehensively write my report as my project was completed.

Social and Ethical Issues

The social and ethical issues I encountered are making my software accessible, copyright and accuracy of the information provided. Making my software accessible doesn’t only mean to allow people who may be of disadvantage able to use it, it also means to make it usable and appropriate for my target audience. To make my software usable for my target audience of year 8s, I included educational content about rainforests in as to teach them content relevant to their science and geography courses. While doing so I also made it so that all aspects of the game are not completely educational as that would likely discourage 13-15-year old’s from playing the game, thus defeating the purpose. I also made the game simple in gameplay in both controls and content in order to increase accessibility. The game being simple means that a year 8 would be able to understand and play it quickly, being able to spend the time learning the tree layers but it also means that those with disabilities will also be able to play it. As the game only requires a mouse to play, those who might lack ranges of dexterity, preventing keyboard use, would also be able to play the game as long as they can read and use a mouse. I also made the game red-green color-blind friendly despite the wide use of greener shades, this allows it to keep its appealing look that keeps users engaged while also allowing its accessibility for red-green color-blind users. The game itself is accessible on unity, and as it is not in a currently published state, it is only accessible to those who use unity or have access to unity and they might need to update the game itself to use it on their version. To ensure that I have been ethical for copyright concerns, I have not copied code directly from the internet. I instead researched the different commands for C# and pieced together my research until I eventually understood enough that I was able to write the code for my game without the need for the internet. The accuracy of the content displayed in the game did require research to refresh myself on the tree layers within a rainforest and I myself learnt the layers in the process. By researching the necessary information, I ensured that the information provided would be factual and reliable in order to educate the users.

Justification of the Solution

The final solution to my problem is very similar to what I initially planned, this is because of my realistic scope, although some sacrifices were made to complete it. Just as initially planned, my game contained the educational content of the tree layers found within a rainforest that aimed at assisting year 8s for their geography and science courses. It does so by using the simple process of planting and watering a tree in order for the labels of the layers to appear, a key change is the use of a tree score instead of building a rainforest. The building of a rainforest by moving trees into the background was an essential part of the incentive to continue playing the game in order to see the progress made in a satisfying way. Due to time restraints, I decided I had to narrow the scope and instead replaced the building of the rainforest with a tree score that kept track of how many trees had been completed. This new method, while less satisfying, does present the player with some sort of reward for completing the game and displays that some aspect of progress occurs when growing multiple trees. Furthermore, another change that occurred early on was to remove the fertilizer bag and change the watering can to a rain cloud. These changes occurred to reduce the screen clutter of having multiple different tools that the player would have to drag over to the growing ground to grow the tree. Not only did this change reduce clutter, it also allowed for a more enjoyable game as it added variation and instead of boring the player with forcing them to do the same steps over and over, it makes it simpler. This increases engagement for the user as by only having to hold a rain cloud over the planted seed, it makes the process of growing a tree simpler yet more satisfying, and allows them to build up points faster. While the time limits meant that the implementation of a reward for the points does not exist, the points still allow the player to hold a personal goal and achievements within the game. The use of bright green and blue colors within the game also allow for an engaging experience as they appeal to a user, engaging them. This means that an aspect of enjoyment can be gained from viewing the pleasing aesthetics and layout of the games and menu. An aspect that was removed from the game was the options for adjusting sound. I made the decision not to include sound as I decided it was not essential to be included due to the time limits, and while it could have added enjoyment, thus engagement, it was unneeded for the gameplay. As there was no need for an options menu without anything to adjust, I instead replaced it with a how to play screen, easily accessible from the menu. The how to play screen provides comprehensive instructions that do not use language a year 8 would be unable to understand. The layout of the main menu itself displays the options of play, how to play and quit very clearly. By having the menu buttons with clear labels, it makes it easy to understand yet despite its appealing colors, is not distracting for a user as to give them ease of navigation. The final result is a game that consists of planting a seed by dragging over a seed package to a growing mound then watering it, with every 5 drops progressing it to the next stage with a label for its tree layer. This happens until it reaches full maturity at 20 drops in which case the seed button (Which is grey when the tree is not fully grown to indicate it cannot be used) becomes active and can be pressed, restarting the process but adding to the tree score.