DES310 Professional Project Personal Portfolio

Joseph Roper - 1901881

Development Diary

Week 1:

- Came up with game ideas, mainly an arcade style zoo game
- First meeting with client

Week 2:

- Made the anteaters' movement in Unity
- Set up source control on GitHub desktop

Week 3:

- Tested 2D prototype
- Discussed changes for moving into 3D
- Set the games' aspect ratio
- Source control fixes

Week 4:

- Fixed the git ignore, so the commits had less data to push
- Got the two designers Josh and Alana onto the source control using GitHub desktop
- Added collision for the car hitting the anteater, making it jolt back and forth like a real life emergency break.
- Updated clients on our progress, they were happy

Week 5:

- Accessed the Jira
- Split the highlight and label mechanics into separate scripts
- Made the tree win mechanic
- Got multiple ant eaters and anthills working
- Added the designers level designs into unity

Week 6:

- Made click and hold mechanic for the cars movement
- Worked on 3D asset integration
- Worked on getting the first playable ready for the client meeting
- The clients were impressed

Week 7:

- Continued 3D asset integration
- Researched a better source control, Fork
- Put the project on Fork
- Started Fork GitHub guide

Week 8:

- Finished for source control guide
- Got the team onto the source control
- Started the level menu system
- Started making an audio manager

Week 9:

- Integrated UI assets into the level menu system
- Added music to the pause screen
- Completed the pause screen
- Started making the game HUD
- Made the car alert mechanic
- Made the speed buttons mechanic
- Got the producer onto the fork

Week 10:

- Fixed merge conflicts for branches
- Cleaned up project folders
- Changed the speed buttons script to suit designers wants
- Added brightness slider functionality in settings menu
- Added tasks on Jira for programmers
- Changed the green overlay for when the game is paused to work better
- Added ant eater counter functionality

Week 11:

- Bug fixing with the car movement
- Added UI assets to counter script
- Added ant eater hit direction functionality for the two different death animations
- Added win state screenshot functionality
- Made the cars de-spawn after reaching the final waypoint at the request of the producer
- Resolved merge conflicts
- Started tech design document

Week 12:

- Resolved merge conflicts
- Finished tech design document
- Made commit log for submission

Startup

When the DES310 discord was created, I directly started messaging my peers from other courses, this led me to quickly form a team within the day and make the discord server where we would interact.

Idea generation for logo concepts started with my first design made in photoshop, this was quickly changed to a much improved logo design made by Daniel the 3D artist with his sausage dog as the mascot.

Game Research

From the first week, the team collectively knew that we were aiming for a simplistic pickup and play game. For example, games found on coolmath games and Friv.

(Images of game hosting websites[1][2])

Many ideas were discussed and curated within the first week to fit the theme of a Zoo arcade game, however after our first mentor meeting the team was informed that mini-games take more work than normal games as you have to make new mechanics and art for each mini-game. From this feedback we decided to settle on one idea, the one closest to the game Frogger and Crossy Road which were mentioned as inspiration in the client's brief.

---

[1] (Coolmathgames.com, 2019)
[2] (Friv.com, n.d.)

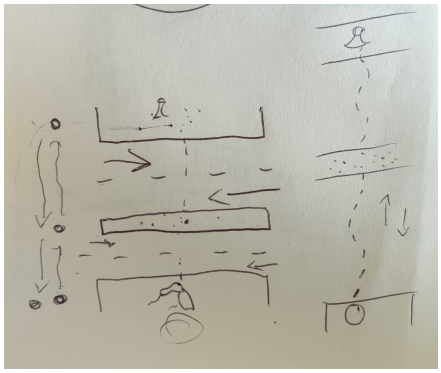
(Images showcasing games[3][4])

I personally am glad we decided to focus on one game idea looking back as it took a substantial amount of time to program the basic gameplay loop of the game.

After the first client meeting, a lot of questions were resolved about the brief, including the art style we could use and the fact that they wanted the game to focus specifically on giant ant eaters and spread the message of conservation whilst playing.

The initial game idea we had was based heavily off Crossy Road, where the user would control the anteater and avoid getting hit by cars crossing the road while trying to eat ants and anthills to improve their score. This idea was discussed with the mentor, and they said that because the user controls the ant eater it makes it seem like it's the anteaters fault that they are getting hit, as a team we thought about this and came up with the idea to reverse the controls making it, so the user has to slow down the cars on the screen to let the anteaters walk past. This new idea helps spread the message of conservation, as the user has to protect the anteater rather than control it.

Below is an initial gameplay diagram from my notebook, it shows that the anteater will spawn from the bottom of the screen and travels up the screen crossing the busy roads to get to the anthill. This diagram is still relevant to what the game is now.



[3] (www.youtube.com, n.d.)
[4] (Crossy Road - Endless Arcade Hopper Game, n.d.)

Mechanics Made

Gameplay Mechanics

To find out how the mechanics work on a technical level, see the technical design document.

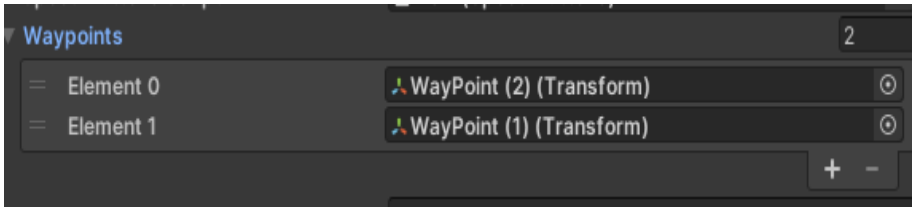
Anteater Mechanics

The anteater was the main staple of the project as it is what the clients wanted to be the main focus of the game, therefore it was a necessary addition to the project. I did not do much research on how anteaters move as this was the task for the animator, however I watched a substantial amount of unity tutorials on how to move 3D objects within a scene.

Waypoint System

The waypoint system is the mechanic used by all objects that move within the game. Initially, the anteaters were planned to use AI such as path finding to move around the scene. However, after some quick play tests, it was found to increase the difficulty of the game as the anteaters' path was too unpredictable. Because of this, we chose to have set paths for the anteaters on each level, which required researching a different system for object movement within unity.

This was the patrolling method where the movable object has waypoints within the scene which it travels towards. This was programmed for ease of use for the designers as it was coded in a fashion to allow the designer to change the amount of waypoints it has, insert new waypoints onto the anteater and change the order of the waypoints, all from the inspector window, so no change of code was needed. Whilst testing the game mechanic we wanted to see the waypoints in the scene to make sure the movable objects are performing the way they should, however in the finished product the team would want them to not be visible, for this I had to research how Unity's render pipeline worked so that I could turn of the waypoint prefabs rendering, thus making it invisible within the game.



This mechanic was chosen to stay in the game as it was easy to design levels with it as the designer could choose the set path in which the anteater should travel within the level and implement it into Unity seamlessly with the use of the inspector window. This mechanic was also used to move the cars within the game, as the designers also wanted the cars to have a set path.

For ease of use, I created a prefab for the ant eater and way points so that whilst making levels you could drag and drop them into the scene, with all the scripts and correct hit boxes pre applied. Prefabs were also useful as if a new script was added or the prefab was changed it would update every instance of this within the game, cutting down on development time.

The waypoint mechanic was changed slightly during the end of production within the car script. As the producer wanted the cars to de-spawn after going outside the screen boundaries. Previously we had the cars loop around the map continuously using waypoints, this was done at the designer's request and helped with bug fixing. However, after finishing the gameplay loop for the levels, the producer noticed that the anteaters finish the level before the cars could loop back around. To save on computational power by having all the cars constantly in the scene, it was beneficial to delete them, this would make the game run smoother. This was a quick change as I just made it so when the car reached the last waypoint set to them, it would de-spawn.

Collision

Collision was necessary for the game as the brief was about anteaters getting run over by cars, so this had to be implemented within the game. For this, research was done into Unity's physics engine to detect collisions. Unity has a lot of premade functions and presets for this, making the collisions more reliable within the game and easier to implement.

The anteater has a rigid body and box collider attached to it, these allow Unity to detect collision with other game objects which contain a collider. The collision works like the pseudocode below.

```
void Collision check (collision information)
    if collider is a Car
        despawn Anteater

    if collider is a Anthill
        despawn Anthill
```

This form of collision detection was used for all the collisions with the anteater, as it worked and was reliable.

Whilst making the collision scripts, I had to find out how to reference other scripts so that functions could be played and variables could be changed from the collision script. The method I used required creating a public variable of the script you want to reference, and then from the inspector window dragging that script or the game object it is attached to into it. This method has been used for nearly every single script I have made as referencing is a key part of hierarchical programming, it makes it so that code can be segmented and not clustered into one file.

Car Mechanics

Breaking

The breaking mechanic is used by the vehicles in the game, it allows the user to stop a car from moving with a mouse click. This was needed as it was an integral part of the game idea pitched to the clients, where the user would stop cars with their input to allow for the ant eaters to walk across the road to anthills for feeding. Research into how Unity detects mouse inputs was done, Unity has preset functions which detect mouse presses. The pseudocode below shows the basic premise of how the mechanic works.

```
void Mouse Down ()

    while car is not stopped
        slow down car
        wait for a milisecond
```

```
void Mouse Up()

    while car is stopped
        speed up car
        wait for a millisecond
```

Initially, the designers wanted to make it, so the user has to hold down the mouse button for the car to stop. This functionality was added to start with, however when testing it had a consistent bug where the car continuously speeds up past the car's max speed set in the inspector, breaking the game. This was due to unity struggling to detect when the mouse button is up. I discussed this with the designers and cut it out from the game. The new system in place is that the user can still click and hold on the cars however sometimes if the user clicks on the cars it stays stopped and doesn't speed up, again this is because unity fails to detect when the mouse button is up sometimes.

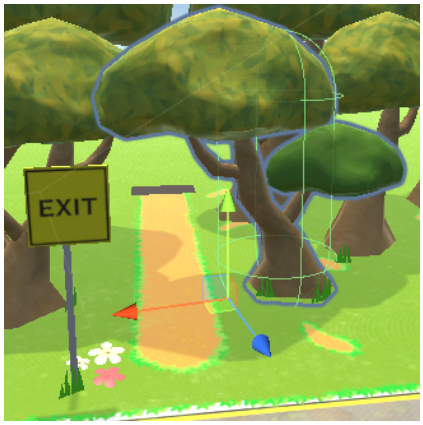
The new development didn't affect the gameplay too much as the team was worried the user could just stop all the cars allowing the anteaters to roam free making the game too easy, however when tested it was found that if someone tried to exploit this bug at least one of the anteaters would still run into the stationary car resulting in the player losing the level.

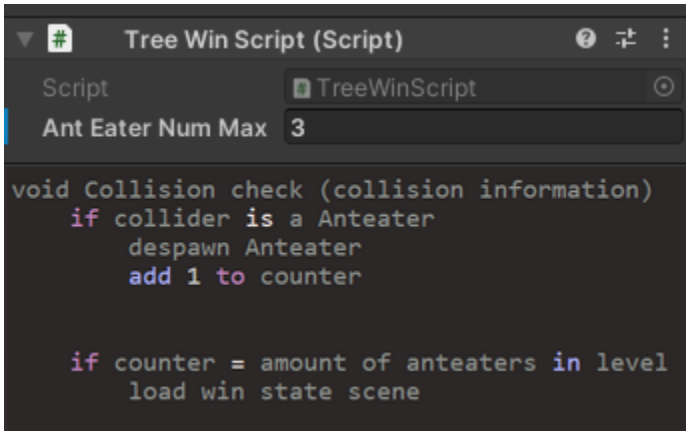
I tried many ideas to fix unity failing to recognize the input, however due to time restraints a solution was never found.

Level Mechanics

Win

The win mechanic was needed as we had to have a way for the user to complete the level. How it works is that each anteater will have a waypoint set at the end of the level. Once the anteater reaches this waypoint it is de-spawned, within the game object the win script is attached to, the designer can set the amount of ant eaters which are on the level. Once the last anteater reaches the end of the level using Unity's scene manager, it is set to load and display the level win screen. For this mechanic no research was needed as it used my prior knowledge and the help of the other programmer on the team Cameron Thustain who worked on the state manager for the game, he informed me on how to change scenes within unity.





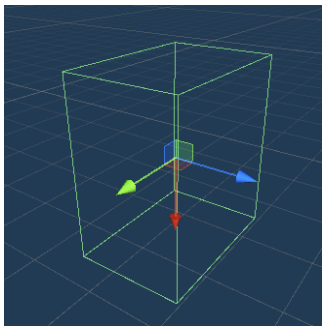
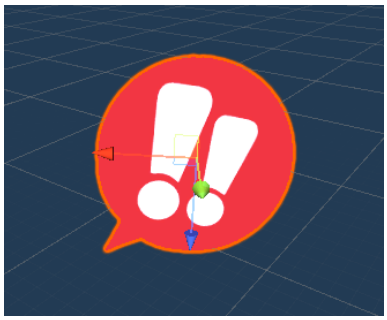
UI Mechanics

The UI art in the game was designed by Alisha the 2D artist within the team. Many of the UI mechanics presented were inspired from popular mobile games such as Angry Birds and Bloons Tower Defence as our game was designed to be one click play and family friendly just like most mobile games.

Car Alert

Whilst shadowing others testing the game, such as our mentor, the team noticed that they kept getting surprised by new cars quickly coming onto the screen, resulting in them losing. To solve this problem, the team came up with a car alert to display before a new car travelled onto the screen. This works by having a game object placed within the scene outside the camera's view perform a ray cast onto the road. If a car hits this ray cast, it instantiates an image onto the screen. For this, I needed to research the instantiating method common place in Unity. For the ray cast detection I used the same method as with the traffic collision designed by Cameron Thustain. Initially the mechanic rendered the car alert image in the 3D level space, however the producer requested for it to spawn in the 2D HUD space, which I changed promptly.


I created 2 prefabs for this mechanic, the first being the detection box so that when making a level it could be dragged and dropped into the scenes. The second being the car alert image so that the script attached to the detection box could access it for rendering onto the screen.

Speed Buttons

The speed buttons mechanic was inspired by the one in Bloons Tower Defence in terms of style and functionality. As the client wants the game to be playable for all ages as a team we thought we should allow the user to choose the difficulty they play at. This is achieved by increasing the speed of the game.

(Image showcasing Bloons Tower Defence game[5])

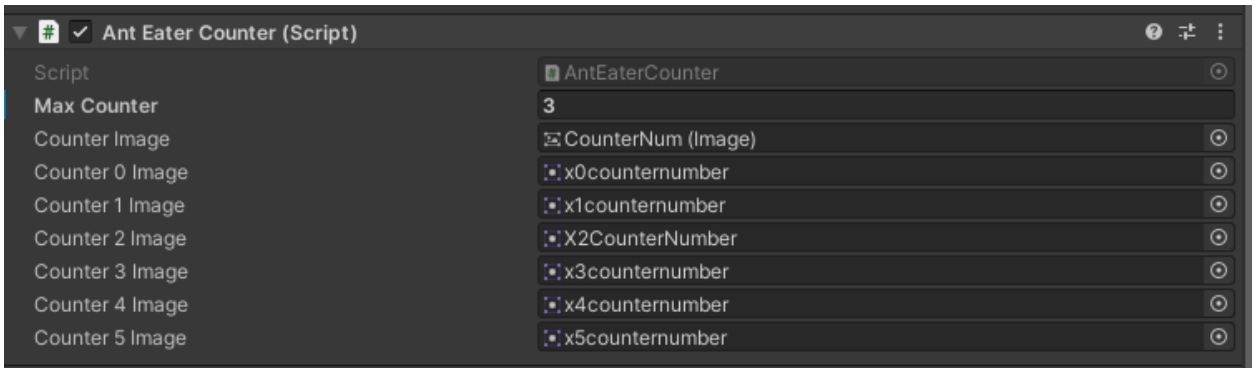
At first this was programmed by having each button set an integer named speed modifier, this was multiplied against the cars, and anteaters speed variable making them move faster.

[5] (Ninjakiwi.com, 2020)

However, when testing this proved to cause a bug for the game as the anteaters still ate the anthills at the same time because of this if the speed was faster the planned course for one anteater would get intercepted by another knocking them off their waypoint trajectory, breaking the game. To fix this I researched into how to speed up the game in different ways, after the research I found out that Unity allows the user to change the game's timescale and make it faster or slower. Now the buttons change a variable that is multiplied against the game's timescale to speed the game up.

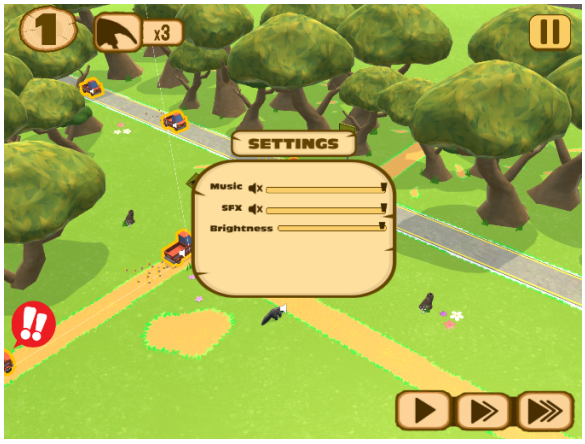
Ant Eater Counter

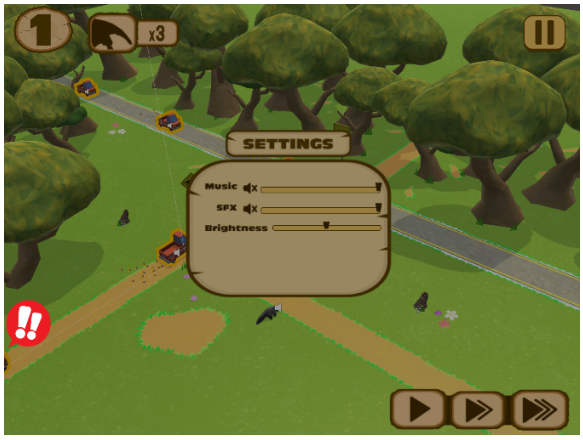
Whilst researching one click play and mobile games, we noticed that all the information the user should want is displayed on the game's HUD. Such as an indication of what's left to do in the game to complete the level. This is why the ant eater counter was made, the counter displays how many anteaters are left to eat within the game before completing it. To program this, I had to research how I could change an image's picture whilst the game is running through the use of scripts. To complete this mechanic it needed new UI assets to be made for the counter numbers , so I informed the 2D artist Alisha who made them quickly and put them on the shared Google Drive for me to integrate into the project. In order to use this mechanic the Level Menu System prefab would have to be applied to the scene, within this the designer would be able to set the starting value for the counter by accessing the anteater counter background image through the inspector window.



Brightness Bar

Whilst the team decided what the user should be able to control in the game, user accessibility was in mind. Because of this, we wanted to allow the user to be able to change the brightness to their liking. Once the mechanic was decided on I researched into controlling brightness in Unity. Although, whilst researching, I could only find guides on how to control the brightness of the lights within the scene. If this method was implemented, it would make it look like how you would control your brightness in your bedroom using a smart bulb. This is not what the team initially planned for, as we wanted it to behave in the same way as controlling your phone screen's brightness. Due to this, I went back to the planning stage and thought of a new way to implement this feature. This was to have an image overlaid onto the game's screen, of which the colour would be black. Using a slider bar, the player would be able to control the alpha value (transparency) of the image. I made it, so the user could only set the max value of the images' alpha value to 80% using the slider. As if the user was able to set it to 100 they wouldn't be able to see the game as it would be pitch black.





Music Sliders

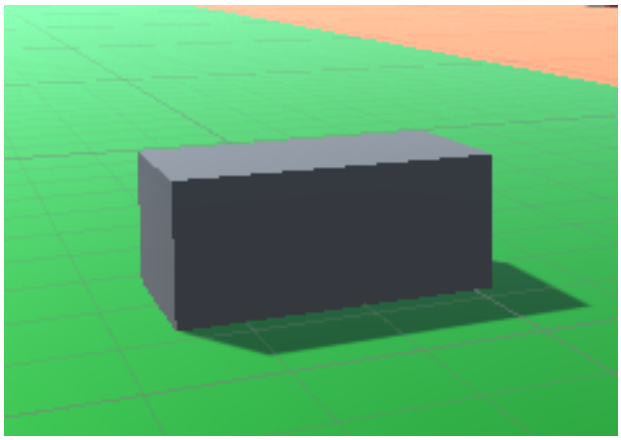
The music sliders are contained within the settings menu and were planned to be sliders because they are more user-friendly and easier to operate. For this, I did research into settings menus and how to build them within unity. Currently, the sliders only change the volume images as the other programmer Cameron was in charge of audio and due to time constraints it was never finished as it was seen as an additional feature.

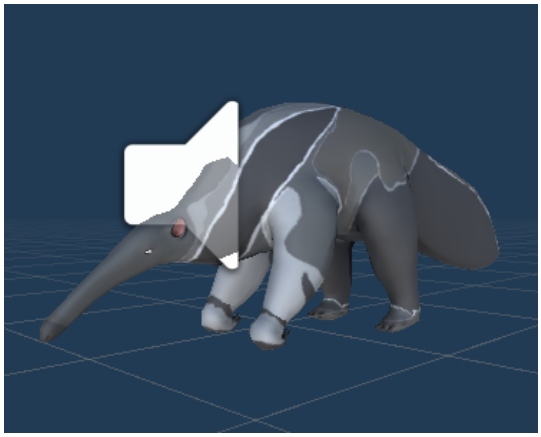
Prefabs

Prefabs are very useful and were used for all important objects within the game. Prefabs are a template which you can make and easily reuse by dragging and dropping it into the scene. For example, I can make a cube and change the colour of it and add different scripts onto it, save it as a prefab, and then I am able to copy that same cube easily into the project. Prefabs can also be re edited after use, and Unity will update any instances of that prefab in the project with the new changes.

Anteater Prefab

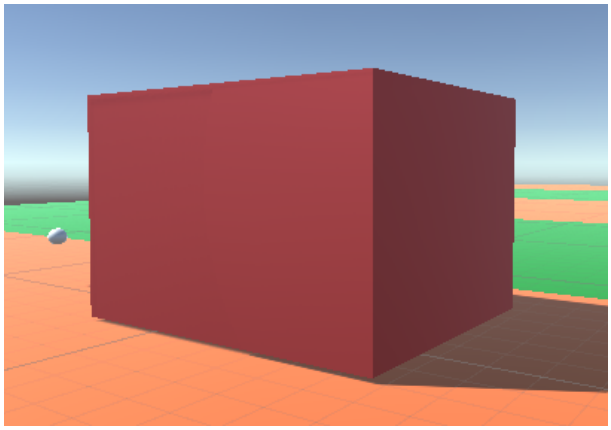
This prefab is widely used within the game as it is the main crux of the project. This prefab started off as a simple grey cube with the scripts to make it move applied to it. Once the 3D model of the anteater was created by Daniel the 3D artist, it was put onto the shared Google Drive. I then implemented it into the project and updated the prefab to fit the new model, this required unwrapping the texture, turning it into a material and then applying it onto the model. The box collider of the model also had to be updated to fit the model's size so that collisions with the anteater were accurate.





Vehicle

The first cars prefab started off as a red box when first testing the functionality of the game, once the 3D model for it was made I integrated it into the project updating the cars prefab this involved using the same method as with the anteater. I made a few changes to the other vehicle objects such as resizing the box colliders and moving the individual components of the model to make them fit together like the reference pictures provided by Daniel, however I was not the one to first implement the blue car and log truck. However, I did apply the necessary scripts to make them work within the levels to them.



Level Menu System

The level state menu system was an important aspect of the game as it contained the Pause Menu, Settings Menu and the in-game HUD which are key features of almost all games.

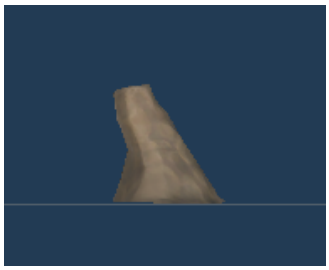
I decided to create all the in-game menus in a singular prefab to make the scripts attached to these systems allow for more efficient inheritance and justified hierarchy. Allowing for simpler and more understandable code between the mechanics. Research was done into menu systems within Unity so that all the menus could easily be merged together. Whilst creating the pause menu I learnt about how to affect Unity's in game timescale to stop and resume the game by the click of a button, this knowledge was later applied onto the speed buttons mechanic.

When implementing the UI assets which were made by the 2D artist Alisha I had to experiment with the placing and scale of the assets at first as I didn't have a complete reference for how the menus should look, resulting in the screens looking shoddy as I am not an artist. This was discussed with Alisha and in turn she created full screen mock-ups of the menus which I could then reference and use as a template, which was extremely helpful.

When the menus were first made they had a green overlay attached to them, however once they were implemented and tested it showed that the overlays layered on top of each other which went against the artist's intentions. To fix this, I removed the overlays from the menus using photoshop and asked Alisha to send me a full screen of the overlay. I then transferred the image into Unity and made it activate alongside the menus, separating the green overlay from the menu images gave the desired effect.

Anthill

I first created the anthill prefab as a white cylinder with a capsule collider and a tag named Anthill, as that's all what was necessary for testing. Once the 3D model was done, another team member implemented the new model and I adjusted the collider so that the anteater collision was still functional.



Source Control

Source control was needed for the project so that all the team members could access and make changes to the game on Unity all at once from their own personal devices. Cameron and I chose to use GitHub as the source control, which uses Git as it is industry standard and reliable. The first interface we chose to use was GitHub desktop, as we were familiar with it from using it through our university career on various modules. This interface was also recommended by the university/module leads. To set this up, Cameron and I started a GitHub organization and created a repository. Initially at the start of the project only me and Cameron used the source control, during this time many problems were found and resolved.

The first being that every commit with unity had 1000s of files, slowing down the source control functions. Due to this, I researched ways to fix this and found that git allows the user to create a text file called git ignore, which tells the source control to ignore saving selected files. I found one curated for Unity's miscellaneous files and added it to the project, this stopped the source control from crashing after every commit, thus saving a lot of development time in the future.

The second problem being as I was only used to using GitHub Desktop for individual projects I never noticed that it didn't have the capability of resolving merge conflicts which when using source control with a team were frequent, especially as Unity saves most of its work as binary files including its scene files, meaning that if 2 people worked on the same scene the two files could not be merged together and only one persons work could be saved into the project. Because of this research was done into other source control GUI's this was when fork was found which took a big learning curve to figure out how to set it up as it controls differently from GitHub desktop which i was used to. Fork allowed me to resolve merge conflicts within the interface rather than typing code in the console window, as well as many other functions, which were very useful when solving problems with team members who didn't know how to operate source control properly due to never interacting with it before the project. These features were mainly: cherry-picking, rebasing and resetting branches.

From researching the most about source control within the team I was put in charge of it this involved: resolving merge conflicts, providing maintenance on the Fork, overseeing each member when they first accessed the source control and merging the designers and artists work together.

To help the team members I created a guide on how to install fork, set up the source control and use it. This guide helped people a lot, however I still had to help them a little whilst they were first setting it up. After helping each member set up the source control, I would see where they struggled and update the guide to hopefully fix the confusion for the next member. The guide contains annotated pictures with each step, making it easy to follow.

The workflow loop for the source control which was made for the team followed like this. A new branch would be created for a Jira task from the main branch which holds the source project. The team members would complete their task on the branch they created and afterwards try to

merge their work into the main project. If the work could automatically be merged through git, the team member was allowed to do so. If there were merge conflicts, they would report this to me and I would rebase their branch onto the main and go through the merge conflicts and resolve each one by one so that the new work could be merged. With each conflict, I would communicate with the team member whose work it was being merged and the member whose work it would be replacing. To make sure one of them was comfortable with potentially redoing their work. Most of the time the conflict was created as a team member accessed a scene or prefab which is a binary file and accidentally committed, so no actual work was done in said file.

To stop merge conflicts as much as possible, I set out a structure for people working on the source control. Tasks would be assigned to members based on what scenes and prefabs they would have to access, so team member 1 would only get tasks to be completed on level 1 and team member 2 would only get tasks on level 2. At the start of the day, members would say what they would need to access in the project to do their work, making sure no one's work crossed. Once the member was finished with their branch, and it was merged into the main branch, they would inform the other team members so that they could then access the scenes and prefabs being used by the team member if needed.

To access this guide, it is located in the DES310 Anteater Highways Google Drive. And follow the file path: Documentation -> Programmers -> Technical Design Documentation -> Fork GitHub Guide

Tech Design Document Contribution

Sections completed by myself:

- Source Control
- Anteater Mechanics
- Anteater - Waypoint System
- Anteater - Collision
- Anteater - Car Collision
- Anteater - AntHill Collision
- Car - Waypoint System
- Car - Breaking
- Car - Collision
- Level Mechanics - Tree Win
- UI Mechanics - Car Alert
- UI Mechanics - Speed Buttons
- UI Mechanics - Ant Eater Counter
- UI Mechanics - Brightness Bar
- UI Mechanics - Music Sliders
- UI Mechanics - LevelStateMenu
- UI Mechanics - Pause Menu
- Prefabs - Level Menu System
- Prefabs - HUD
- Prefabs - Pause Menu
- Prefabs - Settings Menu
- Prefabs - Car Alert

Sections I contributed to:

- Front Page
- Coding Style & File Naming Conventions

Everything that I worked on within the tech design document was done for future development and understanding of the project. Meaning that if the project was picked up again with the current team members or new developers, they would be able to understand how the game functions by reading.

Project Role

The roles I fulfilled during the project were: Gameplay Programmer, System Administrator, UI Programmer and Game Designer. These roles are ordered in terms of relevance.

The role which I picked at the start of the project whilst forming the team was Mechanics Developer, this involved programming the gameplay functionality, a more encompassing term for this role is gameplay programmer. I believe I provided the work needed for this role, as most of the mechanics of the game were made by myself. Throughout the project, I fulfilled the role of a System Administrator by providing upkeep on the project and managing the source control, as well as assisting with the Jira. The UI programmer role was assigned to me as I was in charge of making the in-game level menu system which allows the user to interact with the game. The last role I had was game designer as I planned out most of the mechanics, integrated 3D and 2D assets into the project and implemented level 1. I believe I achieved what was needed for each role consistently throughout the project.

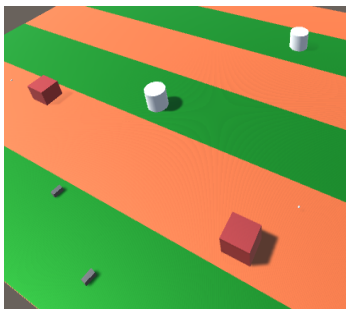
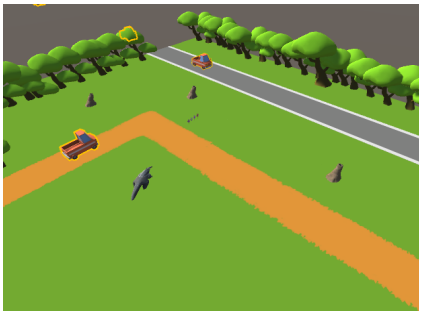
Jira Contribution

The Jira was created and mainly managed by the producer Ross. However, I assisted with the developer section of the management by updating current tasks for team members working on Unity, with more in-depth titles and descriptors as at first members got confused by what needed to be achieved with the task as it could lack context.

Whilst planning out what was needed for the project, I would create new tasks and assign them to members working in Unity by either their speciality or what fit the source control workflow. Once enough work was delegated for the week, the rest of the tasks were left unassigned for members to work on after they have completed their previous work. I would also reorder tasks in terms of priority for the project, helping with the teams' workflow.

Constructing Levels

The first level within the game was produced by myself and Cameron, we used Josh’s level design as a structure. For this we imported Daniel's ground assets and defined which areas would be grass and dirt, a box collider was then applied to it so that the anteater wouldn't fall through the floor. The rest used the prefabs curated by the two programmers and were implemented into the level. Testing was done on this level throughout the project by the designers and programmers, making small changes after each iteration to improve the user experience. Further changes were applied to the scene by the 3D artist such as the addition of flowers and signs as well as improved lighting, this was done to enhance the visuals of the level.


Reflective Summary

Project Requirements

The clients stated in the brief that they wanted a game which would inform and interest people about the current conservation project for giant anteaters in Brazil. The game would have to be for all audiences and most of all fun. The clients liked the idea of one touch play games such as Fruit Ninja as it would make the project more accessible. They also wanted the aspect of road crossing from the likes of Frogger.

I believe this was achieved with the final demo of the game as it used a one button play style as the whole game can be played with only a mouse. The theme and setting of the game is based in Brazil, using their vibrant dirt roads which have started to become common around the giant anteaters' habitat. The style of the game pays close attention to the art around the anteater enclosures within the Edinburgh zoo, where our clients are based. The mechanics of the game allow for a fun and intriguing game loop which inform the player about the conservation project using fun facts given to the player whether they win or lose the level. The message of conservation is spread throughout the game by having the main focus of the gameplay being to protect the ant eaters from getting hit by cars by slowing the vehicles down with user input.

Feedback Implementation

Overall, the clients were happy after every meeting and everything they suggested was already being implemented before the meeting. Our mentor on the project provided a lot of feedback throughout the project, I believe there were two main points made which changed the project altogether.

The first being to not create an arcade game as it is a lot more work than developing one idea, with this feedback we decided to develop the idea which fit closest to the brief in turn setting a definite and clear scope for our project. The second was to rethink our game concept to make it fit the theme of the project more, the concept we had at the time was close to the game Crossy Road where the player would control the anteater and scurry past cars to get a high score, after the feedback was given the team discussed changes and realized that the concept we had could give the player the opinion that it's the anteaters fault if they get hit by a car because they weren't skilled enough at avoiding them. To fix this, we decided to invert the controls so that the user slows down the cars, in turn changing the gameplay from surviving to protecting.

Strengths

The overall concept of the game was well grounded and achievable from early on, the game was adored by the clients, making them excited about every meeting. The artwork created for

the project is simple yet full of life and shows a lot of research into both the zoo and the conservation project in Brazil. It also fits the child-friendly theme frequent in one click games. The mechanics for the game achieve everything that was set out at the start when the game's design was first made, it does this with minimal bugs and allows for further expansion without difficulty due to the nature of the programming being focused on usability.

Improvements

During the project workflow was halted due to source control issues this was fixed by switching the GUI from GitHub Desktop to Fork, if this was done sooner it would've allowed for less work to have been lost at the start of production from merge conflicts between branches.

I believe the Jira could've been more organized with a clearer list of tasks which needed to be finished for each sprint, this would in turn motivate teammates to complete set work and allowed for the team to be informed more of where the project was at and where it was heading.

Whilst working on the project I noticed that I kept getting assigned work outside my role as a programmer and more fit for a designer, at the start of the module I believe the roles and what they entail should have been more defined so that the programmers could focus on polishing the game mechanics and even making more.

During the project two team members left due to illness a designer and an artist, when this happened the project should have been re-scoped, as I found that too much production time was spent on producing more features than showcasing the ones already made. For example the two designers time was heavily focused on producing more music for the game and a death animation for the anteater, this time could have been spent on creating more levels for the game as currently there are only 2 which I don't believe gives the player enough time to get a feel for the game. This would have allowed for more testing and refining levels, as I believe the two current levels in the game are quite difficult, especially if it is the first time playing.

Future Development

For future development of the project there are many ideas which the team discussed but were outside the scope given the small timescale available. The first being more levels added to the game, this would be straightforward given the nature of programming throughout the project, all what would be needed is a level design and then placing prefabs into the scene. The team would want an official ending for the game by introducing a nature bridge into the last level for the anteater to cross the road safely without help from the user. Nighttime levels would be introducing providing more gameplay, this game mode would limit the player's vision allowing` for a fun twist from the daytime levels. We would want to port the game over to mobile increasing the amount of player potential in turn spreading more awareness for the project.

Bibliography and References

Pictures used

Coolmathgames.com. (2019). Run 3. [online] Available at: https://www.coolmathgames.com/0-run-3.

Friv.com. (n.d.). Friv : Free Games! [online] Available at: https://www.friv.com/.

www.youtube.com. (n.d.). Frogger (Windows, 1997) Walkthrough. [online] Available at: https://www.youtube.com/watch?v=jGoivXOUA-0 [Accessed 23 May 2022].

Crossy Road - Endless Arcade Hopper Game. (n.d.). Crossy Road - Endless Arcade Hopper Game. [online] Available at: https://www.crossyroad.com/.

Ninjakiwi.com. (2020). Bloons Tower Defense 5 - BTD 5 - Ninja Kiwi, Creators of the best free online TD games on the web - Ninja Kiwi. [online] Available at: https://ninjakiwi.com/Games/Tower-Defense/Bloons-Tower-Defense-5.html.

Waypoint System

www.youtube.com. (n.d.). Super Easy Patrolling AI | Unity Tutorial. [online] Available at: https://www.youtube.com/watch?v=22PZJlpDkPE&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=10 [Accessed 23 May 2022].

www.youtube.com. (n.d.). FULL 3D ENEMY AI in 6 MINUTES! || Unity Tutorial. [online] Available at: https://www.youtube.com/watch?v=UjkSFoLxesw&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=12 [Accessed 23 May 2022].

www.youtube.com. (n.d.). Unity AI Patrol, Chase, Attack Tutorial in Less than 8 Minutes(Advanced AI Controller). [online] Available at: https://www.youtube.com/watch?v=ieyHlYp5SLQ&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=14 [Accessed 23 May 2022].

www.youtube.com. (n.d.). How To Make Easy Wander AI In Unity! [online] Available at: https://www.youtube.com/watch?v=aEPSuGlcTUQ&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=3 [Accessed 23 May 2022].

www.youtube.com. (n.d.). How to use Unity NavMesh Pathfinding! (Unity Tutorial). [online] Available at: https://www.youtube.com/watch?v=atCOd4o7tG4&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=6&t=177s [Accessed 23 May 2022].

www.youtube.com. (n.d.). A* Pathfinding in Unity. [online] Available at: https://www.youtube.com/watch?v=alU04hvz6L4&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=7 [Accessed 23 May 2022].

www.youtube.com. (n.d.). Unity Tutorials - How to make an object invisible. [online] Available at: https://www.youtube.com/watch?v=FaiScdvILWI&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=15 [Accessed 23 May 2022].

Collision

Detecting Collisions (OnCollisionEnter) - Unity Official Tutorials. (2013). YouTube. Available at: https://www.youtube.com/watch?v=QRp4V1JTZnM [Accessed 6 May 2020].

www.youtube.com. (n.d.). Unity - How to Detect Collision in C# [Using Colliders]. [online] Available at: https://www.youtube.com/watch?v=ZoZcBgRR9ns&t=270s [Accessed 23 May 2022].

www.youtube.com. (n.d.). How to make a Video Game in Unity - COLLISION (E05). [online] Available at: https://www.youtube.com/watch?v=gAB64vfbrhI&t=221s [Accessed 23 May 2022].

www.youtube.com. (n.d.). [Quick Tutorial] How to access Variables from another script - Unity. [online] Available at: https://www.youtube.com/watch?v=JJUnufMLUp0&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=19 [Accessed 23 May 2022].

Click And Hold

www.youtube.com. (n.d.). Unity3D How to : Detect Mouse Click on an Object. [online] Available at: https://www.youtube.com/watch?v=-0eqAUkKQpI&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=15 [Accessed 23 May 2022].

www.youtube.com. (n.d.). Unity SINGLE, DOUBLE and LONG mouse click Tutorial || SkillBased Tutorials. [online] Available at: https://www.youtube.com/watch?v=YrkzeeNDMDA&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=13&t=180s [Accessed 23 May 2022].

Car Alert

www.youtube.com. (n.d.). How to spawn any object at random positions in Unity. [online] Available at: https://www.youtube.com/watch?v=hViPCMZeQC4&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=18 [Accessed 23 May 2022].

Technologies, U. (n.d.). Unity - Scripting API: Object.Instantiate. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/Object.Instantiate.html.

Unity Forum. (n.d.). Change position UI image in Canvas. [online] Available at: https://forum.unity.com/threads/change-position-ui-image-in-canvas.586324/ [Accessed 23 May 2022].

answers.unity.com. (n.d.). Object Instantiate Image to Canvas, Object destroy, Image destroy? - Unity Answers. [online] Available at: https://answers.unity.com/questions/1375883/object-instantiate-image-to-canvas-object-destroy.html [Accessed 23 May 2022].

Unity Forum. (n.d.). Deleting a game object from within it’s self. [online] Available at: https://forum.unity.com/threads/deleting-a-game-object-from-within-its-self.92840/ [Accessed 23 May 2022].

Speed Buttons

Unity Forum. (n.d.). Input Manager Keys? *SOLVED*. [online] Available at: https://forum.unity.com/threads/input-manager-keys-solved.283725/ [Accessed 23 May 2022].

Technologies, U. (n.d.). Unity - Scripting API: Input.GetKey. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/Input.GetKey.html [Accessed 23 May 2022].

Level Menu system

www.youtube.com. (n.d.). Improved Prefab Workflows: Nested Prefabs, Prefab Mode and Prefab Variants. [online] Available at: https://www.youtube.com/watch?v=ibmdm_PoyMA&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=19 [Accessed 23 May 2022].

www.youtube.com. (n.d.). PAUSE MENU in Unity. [online] Available at: https://www.youtube.com/watch?v=JivuXdrIHK0&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=16&t=80s [Accessed 23 May 2022].

Sound sliders

www.youtube.com. (n.d.). SETTINGS MENU in Unity. [online] Available at: https://www.youtube.com/watch?v=YOaYQrN1oYQ&t=10s [Accessed 23 May 2022].

Unity Forum. (n.d.). How to change the size of a UI Image from code? (Trying to make a simple healthbar from this). [online] Available at: https://forum.unity.com/threads/how-to-change-the-size-of-a-ui-image-from-code-trying-to-make-a-simple-healthbar-from-this.265024/ [Accessed 23 May 2022].

Brightness slider

www.youtube.com. (n.d.). Unity Tutorial: Brightness Slider. [online] Available at: https://www.youtube.com/watch?v=MDvPNNgIu7k [Accessed 23 May 2022].

www.youtube.com. (n.d.). Changing Brightness of Light Using UI Slider in Unity. [online] Available at: https://www.youtube.com/watch?v=T2tyoB5iwT8&t=23s [Accessed 23 May 2022].

answers.unity.com. (n.d.). How to make a panel transparent - Unity Answers. [online] Available at: https://answers.unity.com/questions/1455358/how-to-make-a-panel-transparent.html [Accessed 23 May 2022].

Source Control

Fork - a fast and friendly git client for Mac and Windows. (n.d.). Fork - a fast and friendly git client for Mac and Windows. [online] Available at: https://git-fork.com/ [Accessed 23 May 2022].

www.youtube.com. (n.d.). How to use GitHub with Unity. [online] Available at: https://www.youtube.com/watch?v=qpXxcvS-g3g&list=PLdkgl0WszxzF2pQgMdEN-t7juMH9CpazM&index=10 [Accessed 23 May 2022].

GitHub (2018). GitHub. [online] GitHub. Available at: https://github.com/.

GitHub Desktop. (2019). GitHub Desktop. [online] Available at: https://desktop.github.com/.

Importing Assets

www.youtube.com. (n.d.). How to load obj and mtl files as a model unity. [online] Available at: https://www.youtube.com/watch?v=DdyvV8pjHVc [Accessed 23 May 2022].

Level Menu System

answers.unity.com. (n.d.). How to make a UI Image appear/disappear? - Unity Answers. [online] Available at: https://answers.unity.com/questions/1173902/how-to-make-a-ui-image-appeardisappear.html [Accessed 23 May 2022].

Unity Forum. (n.d.). Can’t add script. [online] Available at: https://forum.unity.com/threads/cant-add-script.627679/ [Accessed 23 May 2022].

answers.unity.com. (n.d.). How do I call a function in another gameObject’s script? - Unity Answers. [online] Available at: https://answers.unity.com/questions/7555/how-do-i-call-a-function-in-another-gameobjects-sc.html [Accessed 23 May 2022].