

# Natural Selection

## Evaluation Report

Group 16: AverageDevs

COLTON MATUTE GARCIA - 190061933  
RAEES HUSSAIN - 180196582  
GARNET SAMUELS - 180136865  
BENNY NUAMAH - 190122865  
SAYF ASLAM - 180051616  
BENJAMIN AGYEMANG - 170016319  
ARHAM ZUBAIR - 190220828  
ARUN MAHAY - 180107083  
SAID KHALFAN - 190046413  
SHREY PATEL - 190125534

<http://t16naturalselection.co.uk/>

<b>Introduction</b>	3
<b>Project Management</b>	3
MVP	3
Planning	3
Implementation	4
Reflection on MVP Project Management	4
Final Demo	4
Planning	4
Implementation	4
Evaluation on Project Management	4
<b>Software Development</b>	5
Planning	5
Implementation	5
Software development teams	5
Networking Team	5
Gameplay Team	5
Aesthetics Team	6
Code Distribution	6
Software Development Teams Evaluation	6
Website	6
Game	6
Networking	6
Lobby System	7
Online Feature	7
Gameplay	7
Turn-Based	7
Shooting Mechanics	7
Health system	8
Aesthetics	9
Maps	9
Sprites	9
Testing	10
Software Development Evaluation	13
Codebase	13
Software Product	14
<b>Teamwork</b>	14
General overview	14
Steps taken to improve performance and engagement	14
<b>Conclusion</b>	15
<b>Appendices</b>	16

## Introduction

Natural Selection as a game meets the fundamental criteria of our deliverables. It is a multiplayer, 2D game with a retro theme. It also meets the expectations of our target audience which further reinforces the fact that we have achieved the quality we aimed for originally. It has been produced using Unity, which is a cross-platform game development engine. The Unity development environment is modern, robust and has a large online following which enhances the quality of software produced. The user-interface design was researched; our target audience required it to be clear, concise and attractive. We have ensured we met this expectation by making the UI simple, easy to navigate and applied with our signature retro theme. A key feature of the game is the multiplayer functionality. Users can play with other players in real-time, via a turn-based system. Furthermore, exhaustive testing has ensured that the game is downloadable and playable on any user's PC. Rigorous testing procedures were applied, which are documented below. The code produced is modular and reusable which allows us as well as other developers to manipulate the code and add further functionality to our game. This report evaluates the extent to which the software product, the way our team worked and the way the project and software development were managed to a high quality.

The development of this game utilised the agile methodology where demands and deliverables evolved through the collaborative effort of self-organizing and cross-functional sub-teams. The group was divided into sub-teams based on their strengths. The teams involved: software developers, web developers and report writers. Weekly, the entire team met to update each other on the progress made and whether they needed assistance. The development process was rigorous and entailed consistent testing by sub-teams within the group. All these tests helped us ensure different aspects of our game, whether it's a particular function or the installation, work together seamlessly as a game of high quality would. Moreover, we created different channels on our Discord group which was our primary method of communication. Within this group, we used channels to keep our communications organised. For instance, there were different voice channels for subteams to work together and a larger voice chat channel where we all came together to work collaboratively. The text channels also aided our development procedure. These channels included a "resources" channel where we posted tutorials and documentation which helped the development process. There was also an "announcements" text channel where we informed each other of deadlines or major changes to the game.

## Project Management

To deliver the MVP demo and the final demo to a high standard, we decided to split them into two significant milestones. The work done for both the demos was divided into three phases: Planning, Implementation and Evaluation. This approach made working on the milestones methodical. We used the MVP as a point to evaluate our current project trajectory, success and management to implement changes toward our final demo. We believe this was beneficial as it meant that we were constantly reviewing our project and making necessary changes accordingly. This is what we believe was a major part of the reason why we were able to deliver our final demo. This section will show how we managed our project and how we believe we could have improved it as a whole.

## MVP

### Planning

To plan toward our MVP, we dedicated our first group meeting, on 23/10/20, to decide on specific roles and delegated tasks toward the planning phase of our project and set out the tasks to be completed (see appendix 1.1). To effectively layout the tasks and the roles delegated we decided as a group to produce a Work Breakdown Structure (see appendix 1.2). The Work Breakdown Structure enabled us to see step-by-step the plan of what needed to be done which helped us remain productive. In our second group meeting, on 27/10/20, the Work Breakdown Structure was presented to the group and further tasks were distributed (see appendix 1.3).

## Implementation

We took a decomposition approach for the implementation of the project, thereby, splitting the various tasks into subtasks and dividing them between the respective groups which were created. This approach allowed us to save time as the tasks that needed to be completed could be done at the same time. For example, the design of the website was worked on at the same time as the planning for the documentation. This, therefore, increases the chance of the intended project being delivered on time.

## Reflection on MVP Project Management

The feedback we received for our MVP was disheartening as our project was on the trajectory of a failing grade. As a group, we recognised that the shortcomings of our MVP came from our inadequate level of teamwork. After the two initial planning meetings, we remained in our respective groups and didn't work together effectively, thereby creating a lack of accountability which delayed the timing of the deliverables of our projects.

## Final Demo

### Planning

To plan toward our Final Demo, we dedicated our first group meeting, on 11/01/21, to reflect on and discuss the feedback we received for our MVP and began to take steps toward it (see appendix 1.4). In our meeting, we decided to develop a new Work Breakdown Structure which decomposed the project into smaller pieces to understand what needs to be done (see appendix 1.6). However, after taking into consideration the issues we faced as a group working toward the MVP, we decided to also create a Gantt Chart (see appendix 1.7). The Gantt Chart was necessary and beneficial because it provided a sense of accountability in respect to the deadlines for individual deliverables. The Gantt chart allowed us as a group to know when certain deliverables were needed and when others could start. This was a key factor in the team being able to deliver the final project. Additionally, the Gantt Chart helped the team know what work was being done at any given time, which meant the team could switch between tasks and prepare for others such as learning a specific coding language or feature for the game development. The addition of the Gantt Chart, upon reflection of the MVP, helped improve our teamwork overall. In our second meeting which took place on 15/01/21 was used to present and solidify the Work Breakdown Structure and the Gantt Chart. It was also used to further discuss the steps moving forward and to delegate the tasks as per the Gantt chart (see appendix 1.5).

### Implementation

We continued to follow a decomposition approach to our implementation toward the final demo. However, the difference in our approach was that we ensured that we had weekly meetings to review our progress. We also used a google doc where we tracked what work was done between each meeting. This tracker was used to keep team members accountable while working on the deliverables. Following the created Gantt Chart, Work Breakdown Structure and regular meetings we were able to deliver our end product to a satisfactory standard.

## Evaluation on Project Management

We believe as a group we managed our project adequately, though there was room for improvement. We effectively took on board the feedback and implemented them to deliver the end product. We believe that the aspect of our project management that influenced our success was the introduction of the Gantt Chart after the MVP feedback. This helped us become more organised and accountable. It also meant that we had a clear understanding of what needed to be done to deliver the end product.

In hindsight, we believe there were a few aspects in which we could have improved to manage the project better. One aspect that could have been improved is we could have created a set of goals from the beginning such as S.M.A.R.T Goals. An issue we faced in our project is that we had a very abstract and unclear view of success and this, in turn, affected not only our motivation as a group but also meant we did not have a clear bar to meet. If we had implemented S.M.A.R.T goals we believe the level of success would have increased. Another aspect that could have been improved on was the further decomposition of tasks to allow for a more detailed and specific understanding of the deliverables needed. What was found throughout the project was that there was a lack of clarity at some points in what needed to be done. We tried to mitigate this by the use of the Gantt Chart but there was still a lack of clarity in some aspects of the project leading up to the final demo presentation. A final way to improve the project management that we found was to address problems early on. Upon reflection, it is clear we tended to wait until the last minute to address problems in respect to attendance, communication, deliverables and overall implementation of the decomposed aspects of the project. If we had addressed these issues early on, as we discussed in our designed alliance at the start of the project, the overall outcome of our project would have been a lot better than what was produced.

## Software Development

### Planning

To properly understand the project, we decided that it would be helpful to make a diagram that would illustrate how the game would function. We made a use case diagram (see Appendix 17) to highlight the key aspects of the system, making it easier for the software developers to be on the same page and understand the structure of the game. The use case diagram was used as a reference which helped the development of the game.

### Implementation

At the start of the project, we held a meeting to decide how we will work on the project, and we decided to go with the decomposition approach for the following reasons; different schedules made it difficult to work together, ten members are too many to work with simultaneously and the project was too large to tackle at once, therefore it needed to break into sections. For the aforementioned reasons, we broke down the team into three groups: Networking, Gameplay, and Aesthetics, then we broke down gameplay into further teams, for example; Combat mechanics, Player, Turn-based system, etc.

## Software development teams

### Networking Team

The networking side of the project was one of the most intricate yet necessary aspects for the final product, so we decided as a team to have a separate team to focus on that feature of the game. Their work was to allow the game to be hosted by a player and let other players join their session to play together, as well as making sure that all the features work online such as the movement and shooting mechanics, therefore they had to work collaboratively with the other teams.

### Gameplay Team

To make our pitch idea possible, the gameplay team's focus was to implement the features that we as a team agreed on and that we believe it will make the game feel unique. From gameplay mechanics to balancing, the team will ensure that the game is both fair and fun to play. Individual members of the team were selected to complete key tasks, based on their interests in games and their strengths.

## Aesthetics Team

Visuals are important for portraying the themes of a video game, therefore this team's goal was to make the game UI such that it met the retro theme criteria we pitched. This entailed making sure 'Natural Selection' looked and sounded like a retro game with character models, gun models and map designs.

## Code Distribution

Our main objective was to use the Unity Collaborate feature which allows multiple people to edit the same project which would have been ideal, but sadly the basic version of Unity only allows up to 3 people on the same project. There was a way to increase the cap, but it could take months to get access to that feature so we decided to have 3 people from different teams on the main project and use a GitHub repository to share the project with others. The main project was composed of leaders of each team who worked closely together when implementing features from their teams into the main project. Important details discussed in their meetings were then passed on to the teams through their respective leaders to ensure everyone was informed on the current state of the game.

## Software Development Teams Evaluation

Our approach to software development allowed each team to make fully functional prototypes in a separate environment and then incorporate all of the prototypes into the main project. This allowed us to be more organised with our prototype development and teamwork. We believe this was an effective approach as it allowed us to produce work efficiently with our vastly different schedules (Appendix 1.7). However, a downside of this approach was that some members were reliant on their respective team leaders, which caused some bottlenecks throughout the project. Upon noticing this issue, we decided as a team to increase the number of meetings so that updates would be given more frequently.

## Website

Overall, we were pleased with the website as it was minimalistic yet eye-catching, and it fulfils its purpose of engaging the user. During the development of the website, we faced many challenges. The main challenge was communication, due to the pandemic we had to adapt to a new style of communicating, which sometimes was difficult since different people had different schedules and commitments. Therefore aggregation on a time to meet was difficult. To improve the website, we would reduce the amount of time spent on the meeting by talking about the task that needs to be done for the week, which will give us more time to develop and enhance the website aesthetically and functionally.

## Game

### Networking

As a group, our knowledge and experience of implementing multiplayer are limited as none of the members have created a multiplayer game before. So it would have been reassuring to be able to use Unity's own multiplayer solution 'Unet'. However, to our disappointment, Unet was deprecated two years ago. With Unet not being an option available to us, this meant we had to use a third party multiplayer implementation available on the Unity asset store. After thorough research, we had narrowed down the choices available to us to only two, Mirror and Photon (PUN2 - Photon Unity Networking). Overall Mirror and Photon are quite similar as they are both still being updated and refined frequently and they both share the same syntax as Unet. In the end we chose Mirror as our networking solution as it was the most similar to Unet out of the two. Since it is the most similar that meant we could use new mirror tutorials and old Unet tutorials as the Syntax and methods differ only slightly. This is extremely helpful as it meant that there were more resources available to us, therefore leading to a more robust codebase that could do complex networking functions in a relatively small amount of code.

## Lobby System

To improve clarity, we wanted the lobby system to be clear and straightforward to properly let the users know the state of the connection between players so we made the lobby system easy to use and understand. The game uses a client-server connection for the online functionality where the players can host a lobby that fits the nature of the game, this is the most suitable method as the majority of games would be private, and we want users to play the game whenever they want, so the game does not rely on an outside server to host the games, but rather allows players able to host their games.

A downside of this approach is that the game relies on the host's connection speed and if it is poor, the lag is very noticeable while playing. This is an outside factor that we did not have any control over so we decided as a team to focus on creating the core of the game. If we had more time to work on the project, we would've liked to add another connection method so that players with a slower internet connection would have a better experience playing.

## Online Feature

One of the most difficult aspects of the project was the online gameplay system because of our lack of experience and how intricate it is. When implementing certain features to the game such as the gun mechanics or the player movement, it felt like re-doing them from scratch as we had to revise which parts will be handled by the client/host so that they function as intended, therefore, online mechanics of the game was a time-consuming aspect.

## Gameplay

### Turn-Based

To implement the turn base system we used the singleton coding pattern. The singleton pattern is a software design pattern that forces an instantiation of a class to only a single instance. It is a coding convention to use a manager class to command various complex objects to ensure a smooth functioning programme. Our PlayerController is a particularly complex class, as is evident in Appendix 8 and 7, that needs to be managed properly so that the game is aware of each player and can switch between turns. We chose to use the singleton pattern because it meshed well with the usage of the mirror framework. Due to Mirror's constraints, all of the code to control the player is contained within our 'PlayerController' class. However, we can decouple the management of the player from the player behaviour and this is what is done in the player-manager singleton class.

Input Blocking is a crucial aspect of a Turn-based multiplayer game, as it is the only thing that separates it from being a Real-Time multiplayer Game. The nature of the singleton pattern means Input blocking is already implemented. This is another pivotal reason we chose the singleton pattern over any other design pattern as this would allow us to delegate our finite time and manpower to other more critical components of our game.

### Shooting Mechanics

The shooting of each gun differed between our MVP stage, later development (completed separately to the main project) and our final game. Initially, we created a shooting script that was attached to each weapon that was also responsible for the gun firing a bullet, as well as the gun's behaviour. This is shown by the shoot functions and variables in Appendix 20. In addition, we created a WeaponScript script that was responsible for switching between each weapon as shown in Appendix 15. However, implementing the prototype guns within our turn-based system, and making them fully functional online in our main project resulted in some key changes.

Implementing the gun prototypes to the main project was planned by all three 3 Game sub-teams to follow the structure in Appendix 9 which was incrementally improved and discussed whilst we gained more knowledge on how to incorporate our guns in an online turn-based game. The behaviour of each gun was made to be different in the final version of 'Natural Selection' by having a class called weapons on each gun Unity GameObject which contained variables to represent each gun's specific characteristics (As displayed in Appendix 16). These variables were made public and could be changed during runtime which made the process of balancing easier as we could change a variable in the Unity IDE and instantly see how it affects the gameplay.

The shooting of each gun was handled in the Player Controller script to make the action of the bullet firing local to each player. This was done by having the player telling the server when they have fired and the server displaying this for all players. The code for this is shown in Appendix 5 and more specifically CmdShootRay. Furthermore, Appendix 5 shows how each player instantiated a bullet into the map and applied a shooting force to the bullet in the RpcFireWeapon() function. The FlipWeapon() function shown in Appendix 8 was responsible for flipping the sprite of each 3 guns in the weaponArray to make sure the gun was always responsive and facing the correct way.

The final weapon system that we were able to create in comparison to the weapon system we had planned, did meet our initial objectives of having: a responsive retro-themed array of weapons that the player could at any point cycle between. Successful implementations of these features allowed the game to whilst being turn-based feel responsive and fast on each player's turn further achieving the genre of game we were trying to draw inspiration from 'Worms'. This method of development of having prototypes separate from the main project was an effective use of an Agile SDLC that our team agreed upon.

Depending on user feedback, we could have added the feature to include an ammo counter as displayed in Appendix 10 as a menu game mode option. This feature would diversify how players could interact as they would be given the option of having to either remember the number of bullets and play strategically as in the first iteration of our game or play in a more real-time fast pace game setting where each player will always know how much ammo they possess.

One improvement that could be made to the weapon system is having a larger variety of weapons with more distinct characteristics. Such as this bow and arrow which has variable firing power that depends upon how long the fire button is pressed before being released. As shown in Appendix 10. Also, the trajectory of the arrow follows a parabola to show the forces of gravity acting on the arrow.

Another weapon we could have added in the future is a rocket launcher as demonstrated by Appendix 11 and 12, the rocket launcher can destroy other objects as shown by the destruction of one remaining arrow that hit the red enemy cube in Appendix 12. With further development, we could implement a destructible map environment to make the rocket launcher feel more realistic, impactful and exciting to use.

## Health system

Combat in the game is one of the main features of our game and having an efficient health system was one of our priorities. We made the health system as simple as possible, so it would not hinder the visuals of our game, so to display the current health of a player, we had a slider on top of their head with a 'heart' icon so that the players could tell that the slide represents the health of the player. An issue that we figured out was that when a player has a very small amount of health, it is not clear if they are alive or dead, as it is hard to see if the player has any health left, because the icon is slightly on top of it. If we had more time to work on the project, we would have liked to slightly update the health display so that it is more clear to the players which would avoid confusion.



## Aesthetics

Our main objective for the design of the game was to make it clear and simple to understand, so on our approach to the visuals of the game, we made it minimalistic so that the game is not cluttered with information. We chose this approach because we wanted the players to spend more time thinking about the game and strategies rather than be distracted by unnecessary values or over-designed visual features.

## Maps

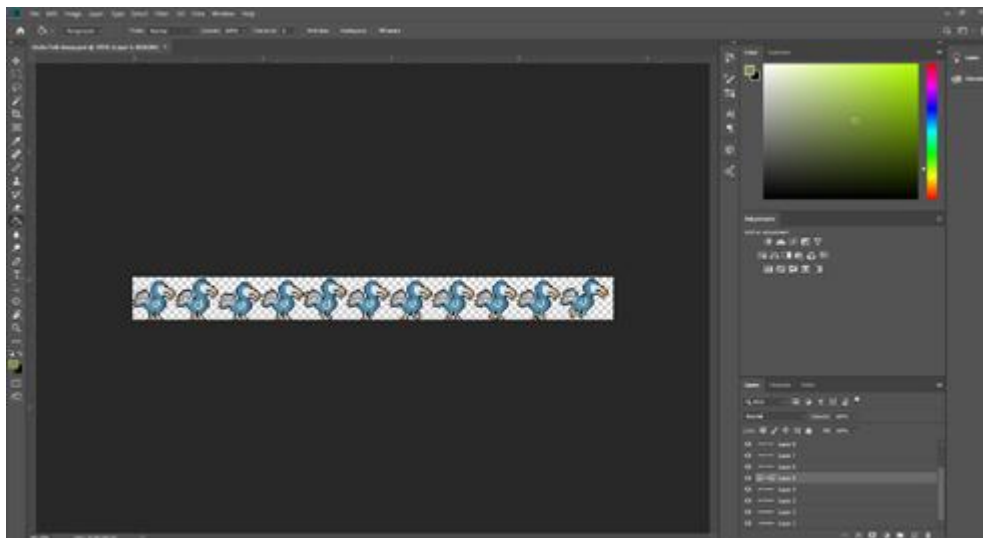
Different map designs encourage different player behaviours. This is a principle we have taken from prominent platformer games that also incorporate combat. Such as *Hell is other demons* and *Hollow Knight*. For example in Appendix 18 the map design incorporates many steep hills that can not be shot through by the player, therefore the players have to strategically approach each other so they do not end up in a compromising position where they can be easily killed by a player waiting at the top of the hill.

Appendix 19 directly contrasts the design of Appendix 18, instead of multiple steep hills the map peaks in the middle thus splitting the map into two halves that can be contested and controlled by players. We believe this to be an aspect of high-quality map design and, consequently, software quality. The architecture of the maps dictates player behaviour so that it does not lead to the same game-winning patterns which can lead to matches becoming repetitive and ruining the game's replayability.

To make our maps even more high quality we used symmetry to not only make the maps look pleasing to the eye but also a way to balance out the game. Balancing out the game is extremely important as any unfair advantages players may have had has been removed so only the players' strategies, skills and intuition are tested.

## Sprites

For the character design, we concluded that the dodo would be the best. We wanted to make the players more invested by getting the dodos to match the tileset of the levels so they don't look out of place and make the game more immersive. Unfortunately, there were no premade assets for a dodo that fit the tile-set we had. So, we decided to make our own characters from scratch and add all the animations manually to have full control of how we wanted the character to look and move.



Here you can see all the different images for the blue coloured dodo put together to make an animation sprite sheet which was imported to unity and rendered to make an immersive character sprite sheet. We made 4 coloured versions of the Dodos to not only add variety to the game but to also let the players

know which specific character they are controlling. This was done to avoid confusion between the players while simultaneously adding more enjoyment to the game.



These were the 4 colours of the dodos we ended up with, each with its own animations. We used the unity animation controller to decide which animations to play depending on what buttons the user pressed.

In the future, if we had more time we would have liked to add different species of characters to the game to add more variety, for example, we wanted to add dinosaurs and reptiles but we didn't have enough resources and time to add that many as we would also have needed to make animations for each of those.

#### Testing

To effectively test our game, we have employed a variety of different testing approaches as shown in the table below. The purpose of testing is to allow us to document, detect and solve any errors we may have within our game, thus ensuring quality control. We have used: Functionality testing - to ensure that our end product works as specified; as well as Blackbox and Whitebox testing. Performing an exhaustive amount of testing ensured that our end product is mostly bug-free, fully functional and stable. Screenshots of the results of the tests are shown in the Appendix.

Test Description	Expected Outcome	Actual Outcome	Action Taken
The game allows a user to host the lobby over the network. [Appendix 21]	When a client decides to host a game, they should be able to host a lobby and players should be able to join the lobby by inputting the IP address of the host.	When a user does decide to host, players can connect over the network and join via the network, a room is created specifically for that host.	We have ensured that the host can create a room and people can join the room, this was a key criterion in our design brief and we have ensured that through our code we have met it.
The game allows clients to join a lobby hosted by other players over the network. [Appendix 22 and 23]	Up to 3 players can join a hosted lobby.	Using the IP address of the host, up to 3 players can connect over the network to the host.	According to our design brief, we made sure that the game supported up to 4 players with one of the players hosting the game.

<p>The game only allows the game to start, once everyone is in a ready state.</p> <p>[Appendix 13 and 14]</p>	<p>The start game button should only be clickable by the host once all the players are ready.</p>	<p>As shown in the screenshot in the appendix, the start button is only clickable by the host once every player is ready, otherwise, the game will not start.</p>	<p>The start button is disabled until the players that have readied up are the same as the players connected in the lobby.</p>
<p>All the buttons within the game work and are easy to follow.</p> <p>[Appendix 13 and 14]</p>	<p>All buttons should have a logical design and work as intended.</p>	<p>We have ensured that all buttons within the game are clear in what they do and do what they are supposed to do.</p>	<p>The buttons have been made to ensure that the text is in a suitable font and size, ensuring that they are easy to follow. The buttons have been designed intuitively, to ensure the quality of the game and each button leads to the relevant screen.</p>
<p>The game ensures that the turn-based system works correctly.</p> <p>[Appendix 26]</p>	<p>Only one player - when it is their turn - should be able to move/or shoot. Once this time has ended, the player should be frozen.</p>	<p>Whichever player's turn it is, only they can move or shoot. We have given the player 20 seconds to do so. The turn-based system works mostly as intended as it switches between all players one at a time after a given time and if the player fires a gun. However, sometimes the player's turns are cut short by a few seconds.</p>	<p>We initially found that having 20 seconds was too long and slowed the game down and thus we changed it to 10 seconds. We have also changed the mechanics such that when a player does shoot, they have two seconds to move before their turn ends.</p>
<p>Once a player's health points have reached zero they should die.</p> <p>[Appendix 25]</p>	<p>Once a player has died a popup screen should appear notifying the user that they have died and/or lost.</p>	<p>Once the player's health bar has reached zero, then the player has died making them unable to move or shoot. A 'lost' screen pop-ups letting the player know they have lost.</p>	<p>We have made sure that when the health bar does go down to zero, that the player is defeated. A screen pops up showing that the player has lost.</p>
<p>The game should ensure that there is a winner.</p> <p>[Appendix 24]</p>	<p>When there is only one player left, they should be declared the winner.</p>	<p>We have ensured that the network manager always keeps a track of a winner.</p>	<p>We have used the concept of 'last man standing' to ensure that the game will always have a winner by counting the dead players, if the count of dead players is one less than the count of total</p>

			players, then the player who is not dead is the winner.
Ammunition should hit the player and the system recognises it. [Appendix 27]	When a player has been hit, their health should be decremented.	When playing the game, the system can recognise when the bullets hit a player.	Initially, we had a bug where the health did not decrease when the player was hit. In the health system, we modified the update method so that it checks every frame if the player has been hit by a bullet.
Checking that each weapon deals different amounts of damage. [Appendix 27]	Each weapon should do different amounts of damage.	The weapons spawn different types of bullets, with this feature, the game can check which bullet hit the player, and take away health accordingly.	We made different prefabs for the bullets the weapons shoot since the early stages of development since the damage feature was important to our game. The health system checks the names of the bullet prefab to check how much damage it should do.
Ammunition for guns should decrement logically and correctly. [Appendix 5]	When a gun has no ammunition, the gun should not be able to shoot. After each shot, the ammo for that gun should be decreased by 1.	The gun does correctly shoot when there are bullets in the gun. The gun doesn't shoot out bullets when the bullets have run out. But we found out that the ammo goes down by 2 instead of one.	We found out that the line that takes away one from the ammo count is written twice on different parts of the code. The fix was to delete one of the lines.
The sound effects work as intended. [Appendix 5]	The sound effects in the main screen, when shooting and end screen should all work as intended.	The sound effects and music throughout the game are fully functional.	We have added music in the lobby, main game and at the end screen.
The maps should load correctly on the game screen. [Appendix 18 and 19]	The game should cycle randomly through the three maps created.	On any given match, a random map is assigned to the game.	First, we got one map to load up correctly, once we had accomplished that we were able to add different maps to the game.
The dodo should be	The dodo should be	Initially, we did have an	The issue with the

able to move as intended.  [Appendix 26]	able to move and jump correctly.	issue that meant that the Dodo did not jump correctly. The basic player controls such as movement did however work as intended.	double jump was in the map, we did not set the ground correctly so that dodo could not tell it was grounded.
The weapon aiming should work correctly  [Appendix 9 and 10].	The weapons should follow the mouse and flip the sprites when the player turns around.	The weapon follows the mouse locally but does not show it to the other players. The sprite of the weapon flips when the player turns around but when changing weapons, the next weapon is unflipped so it does not work correctly.	To fix these issues, we added a Network Transform Child to the weapon canvas so that it could where each player is aiming online and to fix the weapon flipping, we made a method that would flip all the weapons when the character flips so that it stays consistent even when switching weapons.

## Software Development Evaluation

### Codebase

While natural selection is a fully functioning turn-based shooter with real-time elements, there are many critiques to be made in regards to the codebase so that the overall architecture can be made more maintainable and robust. However, there are also many high-quality aspects of the code that we will be highlighting.

For example, in regards to Appendix 4, we have descriptive variable names and any unfamiliar programmer will be able to recognize exactly what their functions will be. The variable 'speed' quite clearly correlates to the speed at which the player character moves, and this convention continues throughout our whole codebase. We believe this to be an aspect of high-quality software.

An example of Mirror (the multiplayer framework that we used) limitations causing our code to not be as high quality as we wished is contained in Appendix 5. This is because special mirror methods designated with Command and ClientRpc keywords can only be declared in the player object which in our case is 'PlayerController'. These special methods are vital for updating the server state and ensuring that all players remain synchronized. They can become fairly dense however we can not separate them into their own specific scripts.

In Appendix 3 you can see we have commented out sections of code that we have not included in the final version of our game. Sometimes experimental features of the game did not work as intended or were not required for our purposes in the end, however, we chose to only comment out the code in case we chose to revisit the feature at a later date. In the completed version of the codebase, these sections should have been deleted to make the codebase more readable and, consequently, more high quality overall. This also leads to redundant methods as seen in Appendix 2. In the process of testing and tweaking our game we removed the main snippet of code from the 'NextPlayer' method leading it to lose its purpose of having the server deal with changing players, instead, it became redundant as a single

line of code would have executed the same function without an unnecessary middle man. These are examples of artefacts left over from the process of game development.

Code comments are another way we could have elevated the quality of our codebase. Appendix 6 showcases the way we have used code comments to help in our development. As a group, we used decomposition to break down the core aspects of our game and delegate out these main components to each of our development teams. So code comments were used to reference who contributed which methods and variables. This makes it easier when collating our works together and when testing our project. We used some code comments throughout the codebase however it should have been constant throughout the development process as it is a good, established coding practice.

## Software Product

For the final product, we as a team believe that we met our initial objective of making a strategic retro-themed, online multiplayer game. Through testing the game, we found the game to be fun and refreshing, and we are proud of that. We were focused on ensuring the game does not feel like a carbon copy of our reference game; 'Worms'. Therefore, to ensure that our game is unique we spent an extensive amount of time thinking about our USP (Unique Selling Point). In the end, we feel we have successfully differentiated our game as it is more fast-paced without losing the strategic factor. This is due to the shorter turn times and smaller maps. These two elements lead to more action and tension throughout the game, thereby making the game more competitive.

Despite the game meeting the fundamental criteria we aimed for, the final product comprises some issues which we would like to have worked on if we had more time. For instance, the most noticeable yet occasional issue is lag. We believe the game is fully playable, but it could be optimized in a way that would improve the lag compensation, making the game feel smoother and clearer.

## Teamwork

### General overview

At the start of the first term, each member of the group contributed towards a Designed Alliance which stated our intentions and commitments for each team member to undertake within the group. We also created several means of communication that allowed us to keep up to date and work collaboratively. For example, we have created a Discord server to allow us to complete work simultaneously, share resources easily and document our progress whilst also using WhatsApp to schedule meetings, issue simple reminders about our specific team deadlines and ensure any module wide announcements were reiterated.

Despite having all of these resources in place, the general level of teamwork amongst the group was not always fluid and consistent. The first 4-5 weeks into the project were very productive, with meetings taking place weekly and each member of the group was completing their task by their given deadline. However, the 5-6 weeks approaching the MVP were unfortunately much less effective. Weekly meetings became fortnightly meetings and the absence of members at meetings echoed throughout the team with fewer and fewer people turning up to each meeting. The lack of organisation and firmness throughout meant that the final weeks leading up to the MVP were rushed and led to us being ultimately punished by producing a poor product.

### Steps taken to improve performance and engagement

The shortcomings of our MVP acted essentially as a wake-up call for the team and was the starting point for change. Firstly, we decided upon a group leader who was responsible for ensuring everyone

within the team was aware of their roles and responsibilities and to ensure that tasks were completed on time. The leader also divided tasks to group members based on their strengths and weaknesses. Those with better programming experience and with a deeper understanding of games were assigned roles related to coding specific aspects of the game. Members with skills relating to web development were assigned to the website and finally, those with better-written skills were tasked with completing report based assignments. We also decided upon a given day and time of the week for everyone to meet. We kept this the same for each week since we had already figured out that it was a time everyone would be available. In the rare case of a member being unable to attend the meeting, an available member would fill them in on our discussion and what is expected of them in the upcoming week.

## Conclusion

To conclude, we believe the software products, which entail the game and the website, were delivered to a high standard. Though we have met the fundamental criteria of the product deliverables, there remains room for improvement. Our work and therefore progress up until the MVP demo was inadequate to help us achieve a passing grade which was strongly reflected in the feedback we received post-demo. Specifically, the poor project management and lack of effort in the team altogether foreshadowed the quality product delivered as the MVP. The feedback received helped us gain a true perspective of the project's trajectory. This ultimately changed our ad-hoc approach to the project and pushed us to become much more organised. The frequency of team meetings and collaborative work, motivation and accountability all improved which helped us deliver completely functional products.

This project has taught us valuable lessons which will help us in both our professional careers as well as our personal lives.

## Appendices

### Appendix: 1.1

## 23/10/20

---

### Logistics

Time:	15:00-15:45
Date:	23/10/20
Attendees:	Benjamin Agyemang, Raees Hussain, Said Khalfan, Arun Mahay, Colton Matute Garcia, Benny Nuamah, Garnet Samuels, Arham Zubair
Please Bring/Read:	
Meeting purpose	To plan towards the submissions needed for the hack

### Agenda

Item	Time	Agenda Item	Presenter
1	15:00	Decide on concrete roles within the group moving forward in the project	
2	15:30	Discuss the tasks needed to be completed for the upcoming hack	
3			
4			

### Closed Actions

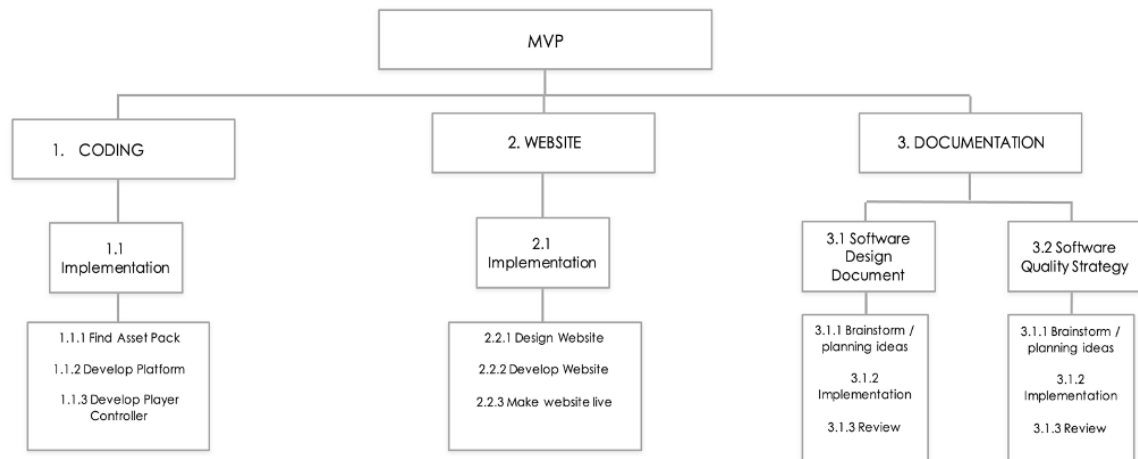
No	Action	Who	When	Status or comment
1.	Concrete roles established	Whole Group	23/10/20	Teams Established: Coding Website Documentation
2.	Tasks distributed toward the hack	Whole Group	23/10/20	Completed

### Open Actions

No	Action/Discussion	Who	When	Status or comment
1.	Create a work breakdown structure for the upcoming hack	Benjamin Agyemang		In Progress
2.	Search for asset packs to fit our retro theme	Coding Team		In Progress
3.	Begin designing the website for the game	Website Team		In Progress
4.	Brainstorm ideas for managing software quality and the design of the software	Documentation Team & Coding Team		In Progress



Appendix 1.2:



## Appendix 1.3:

## 27/10/20

---

### Logistics

Time:	15:00-16:00
Date:	27/10/2020
Attendees:	Benjamin Agyemang, Raees Hussain, Said Khalfan, Arun Mahay, Colton Matute Garcia, Benny Nuamah, Garnet Samuels, Arham Zubair
Please Bring/Read:	
Meeting purpose	

### Agenda

Item	Time	Agenda Item	Presenter
1	15:00	To show developed work breakdown structure	Ben
2	15:10	To show potential asset packs	Raees
3	15:25	Vote for asset pack	
4	15:30	Present ideas for managing software quality and software design	Documentation & Coding Team
5	15:50	Discuss next steps moving forward	

### Closed Actions

No	Action	Who	When	Status or comment
1	Work Breakdown Structure Accepted by the group	Ben	27/10/20	Completed
2	Asset pack voted on	Raees	27/10/20	Completed
3	Next steps established			Completed

### Open Actions

No	Action/Discussion	Who	When	Status or comment
1.	Begin to write up designed alliance and software quality documents	Documentation Team		In Progress
2.	Create survey to understand target audience	Documentation Team		In Progress
3.	Continue designing the website for the game	Website Team		In Progress
4.	Implement ideas for managing software quality and the design of the software	Coding Team		In progress

# 11/01/21

---

## Logistics

Time:	17:00-17:45
Date:	11/01/2021
Attendees:	Benjamin Agyemang, Raees Hussain, Said Khalfan, Arun Mahay, Colton Matute Garcia, Benny Nuamah, Garnet Samuels, Arham Zubair
Please Bring/Read:	
Meeting purpose	

## Agenda

Item	Time	Agenda Item	Presenter
1	17:00	Discuss and reflect on the feedback given from the MVP	
2	17:30	Discuss next steps moving forward for the final demo	

## Closed Actions

No	Action	Who	When	Status or comment
----	--------	-----	------	-------------------

## Open Actions

No	Action/Discussion	Who	When	Status or comment
1.	Draw up the Gantt Chart and Work Breakdown Structure for next session	Documentation Team		In Progress

Appendix 1.4:

# 11/01/21

---

## Logistics

Time:	17:00-17:45
Date:	11/01/2021
Attendees:	Benjamin Agyemang, Raees Hussain, Said Khalfan, Arun Mahay, Colton Matute Garcia, Benny Nuamah, Garnet Samuels, Arham Zubair
Please Bring/Read:	
Meeting purpose	

## Agenda

Item	Time	Agenda Item	Presenter
1	17:00	Discuss and reflect on the feedback given from the MVP	
2	17:30	Discuss next steps moving forward for the final demo	

## Closed Actions

No	Action	Who	When	Status or comment
----	--------	-----	------	-------------------

## Open Actions

No	Action/Discussion	Who	When	Status or comment
1.	Draw up the Gantt Chart and Work Breakdown Structure for next session	Documentation Team		In Progress

## Appendix 1.5:

# 15/01/21

---

## Logistics

Time:	17:00-17:45
Date:	15/01/2021
Attendees:	Benjamin Agyemang, Raees Hussain, Said Khalfan, Arun Mahay, Colton Matute Garcia, Benny Nuamah, Garnet Samuels, Arham Zubair
Please Bring/Read:	
Meeting purpose	

## Agenda

Item	Time	Agenda Item	Presenter
1	17:00	Present Work Breakdown Structure & Gantt Chart and discuss	
2	17:30	Discuss next steps moving forward for the final demo	
3			

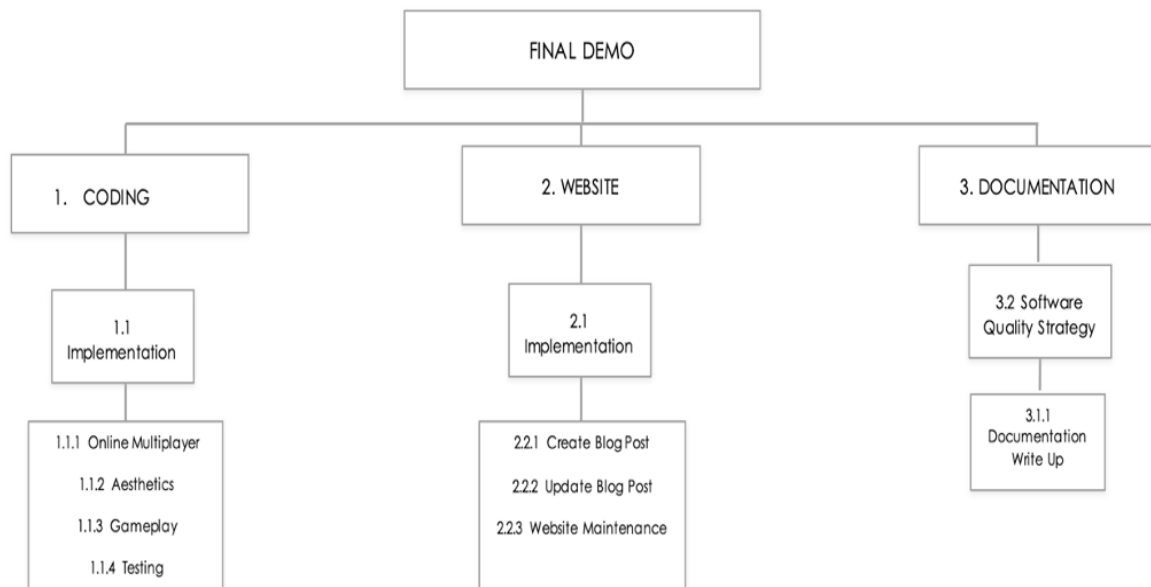
## Closed Actions

No	Action	Who	When	Status or comment
1	Work Breakdown Structure and Gantt Chart accepted			Completed

## Open Actions

No	Action/Discussion	Who	When	Status or comment
1.	Begin to work as Gantt Chart and Work Breakdown Structure shows	Whole Team		In Progress

## Appendix 1.6:



TASK ID	TASK	START DATE	END DATE	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11
1	Game Development	18/01/21	22/03/21											
1.1	Multiplayer	18/01/21	22/03/21											
1.1.1	Online													
1.2	Aesthetics	18/01/21	22/03/21											
1.2.1	Maps													
1.2.2	Sprites													
1.3	Gameplay	18/01/21	22/03/21											
1.3.1	Turn Based													
1.3.2	Shooting Mechanics													
1.3.3	Health System													
2	Website Development	18/01/21	22/03/21											
2.1	Blog uploads													
2.2	Website Maintenance													
3	Documentation Development	15/01/21	31/03/21											
3.1	Project Documentation													
4	Testing													

## Appendix 1.7:

## Appendix 2:

```
//if (!isServer)
//return;
currentTurnTime -= Time.deltaTime;

if (currentTurnTime < 0)
{
    NextPlayer();
}

public void NextPlayer()
{
    //if (!isServer)
    //return;

    StartCoroutine(NextPlayerCoroutine());
}
```

## Appendix 3:

```
/*
void MoveCamera(int _Old, int _New)
{
    currentPlayer = _New;

    if (_New < 0 || _New >= Players.Length)
        return;

    playerCamera.SetParent(Players[currentPlayer].transform);
    playerCamera.localPosition = Vector3.zero + Vector3.back * 10;
}*/
```

#### Appendix 4:

```
public float speed;
public float jumpForce;
private float moveInput;

private Rigidbody2D rb;

private bool facingRight = true;

private bool isGrounded;
public Transform groundCheck;
public float checkRadius;
public LayerMask whatIsGround;

private int extraJumps;
public int extraJumpsValue;
```

#### Appendix 5:

```
[Command]
void CmdShootRay()
{
    RpcFireWeapon();
}

[ClientRpc]
void RpcFireWeapon()
{
    /*
    //bulletAudio.Play(); muzzleflash etc
    var bullet = (GameObject)Instantiate(activeWeapon.weaponBullet, activeWeapon.weaponFirePosition.position, activeWeapon.weaponFirePosition.rotation);
    bullet.GetComponent<Rigidbody2D>().velocity = bullet.transform.forward * activeWeapon.weaponSpeed;
    if (bullet) { Destroy(bullet, activeWeapon.weaponLife); }

    */
    //ammo

    //PlayerManager.singleton.NextPlayer();

    activeWeapon.PlayAudio();

    isFiring = true;
    activeWeapon.weaponAmmo--;
    isFiring = false;

    GameObject BulletIns = Instantiate(activeWeapon.Bullet, activeWeapon.ShootPoint.position, activeWeapon.ShootPoint.rotation);
    BulletIns.GetComponent<Rigidbody2D>().AddForce(BulletIns.transform.right * activeWeapon.BulletSpeed);
    if (BulletIns) { Destroy(BulletIns, activeWeapon.weaponLife); }
    activeWeapon.gunAnimator.SetTrigger("Shoot");
    //CameraShaker.Instance.ShakeOnce(magnitude, roughness, fadeInTime, fadeOutTime);
}
```



## Appendix 6:

```
//endgame
public bool isDead = false;
public bool hasWon = false;
public bool isPlaying = true;

//end game stuff
[SerializeField]
private GameObject lostGame;

[SerializeField]
private GameObject wonGame;

//turn based
public bool IsTurn { get { return PlayerManager.singleton.IsMyTurn(playerId); } }

public int playerId;

public bool hasShot = false;

//sfx
public AudioSource jumping;
public AudioSource weaponSwitch;
```

```
void Start()
{
    if (isLocalPlayer)
    {
        extraJumps = extraJumpsValue;
        anim = GetComponent<Animator>();
        rb = GetComponent<Rigidbody2D>();
        CmdChangeActiveWeapon(selectedWeaponLocal);
    }
}

// Update is called once per frame
void FixedUpdate()
{
    if (!isLocalPlayer) return;

    if (!IsTurn) return;

    if (isDead) return;

    if (hasWon) return;

    isGrounded = Physics2D.OverlapCircle(groundCheck.position, checkRadius, whatIsGround);
    anim.SetBool("isGrounded", isGrounded);

    moveInput = Input.GetAxisRaw("Horizontal");
    rb.velocity = new Vector2(moveInput * speed, rb.velocity.y);

    if (moveInput == 0)
    {
        anim.SetBool("isRunning", false);
    }
    else
    {
        anim.SetBool("isRunning", true);
    }

    if (facingRight == false && moveInput > 0)
    {
        Flip();
    }
    else if (facingRight == true && moveInput < 0)
    {
        Flip();
    }
}
```

---

Appendix 7:

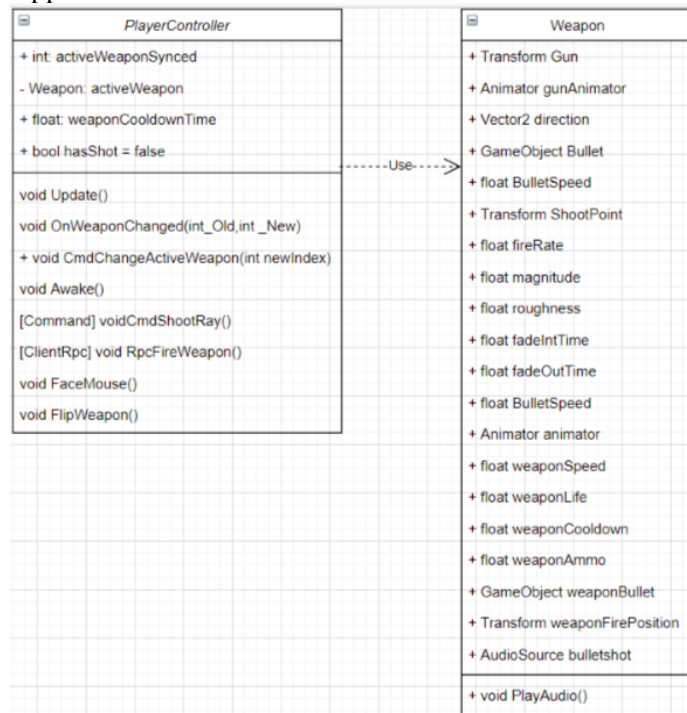
## Appendix 8:

```

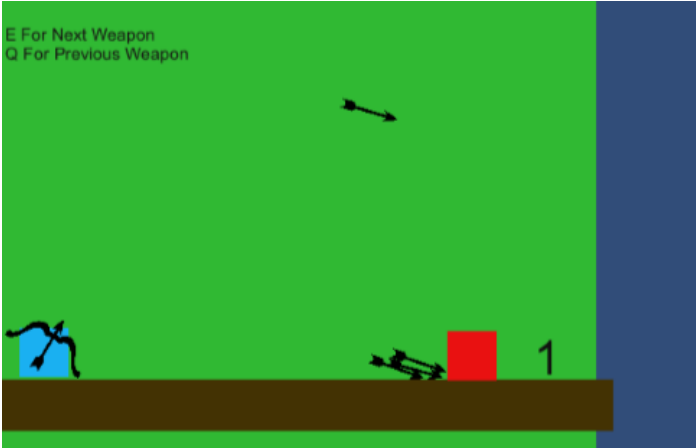
277 //gun faces mouse
278 void FaceMouse()
279 {
280     activeWeapon.Gun.transform.right = activeWeapon.direction;
281 }
282
283 void Flip()
284 {
285     if (!isLocalPlayer) return;
286     //Switch the way the player is labelled as facing.
287     facingRight = !facingRight;
288
289     /*
290     Vector3 Scaler = transform.localScale;
291     Scaler.x *= -1;
292     transform.localScale = Scaler;
293     */
294
295     transform.Rotate(0f, 180f, 0f);
296     canvasBoard.transform.Rotate(0f, 180f, 0f);
297     FlipWeapon();
298 }
299
300 void FlipWeapon()
301 {
302     Transform weaponOneTransform = weaponArray[1].GetComponentInChildren<Transform>();
303     weaponOneTransform.localScale = new Vector3(weaponOneTransform.localScale.x, weaponOneTransform.localScale.y * -1, weaponOneTransform.localScale.z);
304
305     Transform weaponTwoTransform = weaponArray[2].GetComponentInChildren<Transform>();
306     weaponTwoTransform.localScale = new Vector3(weaponTwoTransform.localScale.x, weaponTwoTransform.localScale.y * -1, weaponTwoTransform.localScale.z);
307
308     Transform weaponThreeTransform = weaponArray[3].GetComponentInChildren<Transform>();
309     weaponThreeTransform.localScale = new Vector3(weaponThreeTransform.localScale.x, weaponThreeTransform.localScale.y * -1, weaponThreeTransform.localScale.z);
310 }
311 }

```

## Appendix 9:

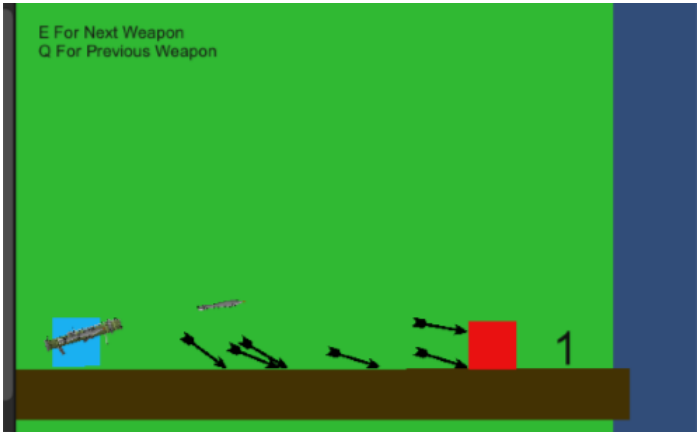


Appendix 10:



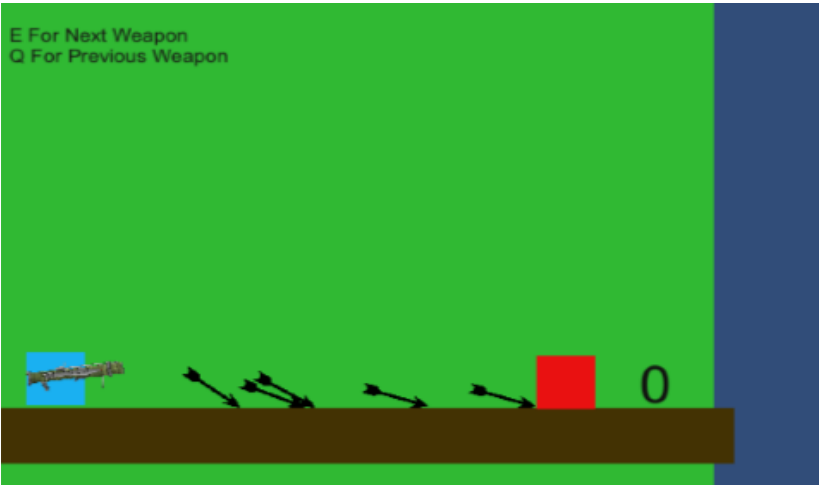
The diagram shows a green rectangular area representing a field. On the left side, there is a blue square. On the right side, there is a red square. A black arrow points from the blue square towards the red square. In the top left corner, the text "E For Next Weapon" and "Q For Previous Weapon" is displayed. To the right of the red square, the number "1" is written. The entire scene is set against a dark blue background on the right side.

Appendix 11:



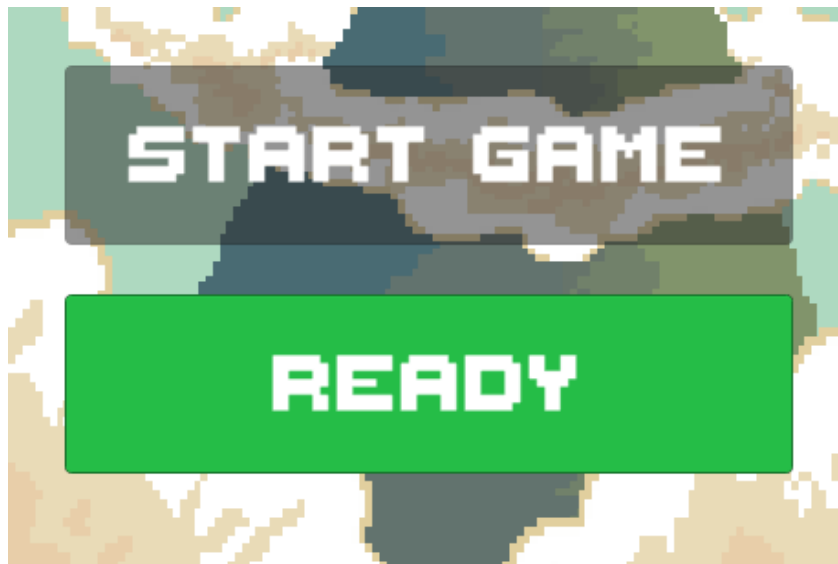
The diagram shows a green rectangular area representing a field. On the left side, there is a blue square. On the right side, there is a red square. Multiple black arrows point from the blue square towards the red square. In the top left corner, the text "E For Next Weapon" and "Q For Previous Weapon" is displayed. To the right of the red square, the number "1" is written. The entire scene is set against a dark blue background on the right side.

Appendix 12:

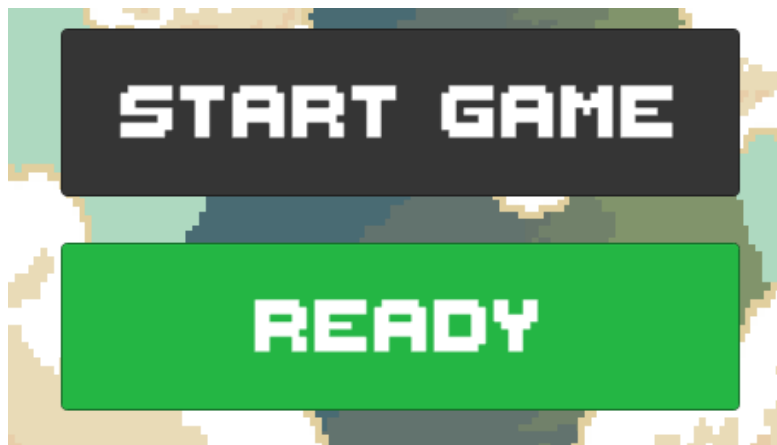


The diagram shows a green rectangular area representing a field. On the left side, there is a blue square. On the right side, there is a red square. Multiple black arrows point from the blue square towards the red square. In the top left corner, the text "E For Next Weapon" and "Q For Previous Weapon" is displayed. To the right of the red square, the number "0" is written. The entire scene is set against a dark blue background on the right side.

Appendix 13:



Appendix 14



## Appendix 15:

```
// Update is called once per frame
void Update()
{
    //if (isLocalPlayer)
    //{
        if (Input.GetKeyDown(KeyCode.E))
        {
            //next weapon
            if (currentWeaponIndex < totalWeapons - 1)
            {
                weapons[currentWeaponIndex].SetActive(false);
                currentWeaponIndex += 1;
                weapons[currentWeaponIndex].SetActive(true);
                currentWeapon = weapons[currentWeaponIndex];
            }

            /*if (currentWeaponIndex < totalWeapons - 1)
            {
                Destroy(Points.GetComponent<ShootBowScript>());
            }*/

            if (Input.GetKeyDown(KeyCode.Q))
            {
                //next weapon
                if (currentWeaponIndex > 0)
                {
                    weapons[currentWeaponIndex].SetActive(false);
                    currentWeaponIndex -= 1;
                    weapons[currentWeaponIndex].SetActive(true);
                    currentWeapon = weapons[currentWeaponIndex];
                }
            }

            if (Input.GetKeyDown(KeyCode.E) && Input.GetKeyDown(KeyCode.Q))
            {
                if (currentWeaponIndex < totalWeapons)
                {
                }
            }
        }
    }
```

## Appendix 16:

```

using Mirror;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using EZCameraShake;
using UnityEngine.UI;

namespace Game
{
    public class Weapon : MonoBehaviour
    {
        public Transform Gun;
        public Animator gunAnimator;
        public Vector2 direction;
        public GameObject Bullet;
        public float BulletSpeed;
        public Transform ShootPoint;
        public float fireRate;
        private float ReadyForNextShot;
        public float magnitude;
        public float roughness;
        public float fadeInTime;
        public float fadeOutTime;

        //ammo display variables
        public int ammo;
        public bool isFiring;
        public Text ammoDisplay;

        //ammo and reloading variables
        private int currentAmmo;
        public float reloadTime = 1f;
        public int maxAmmo;
        //private bool isReloading = false;
        public Animator animator;

        //new weapon
        public float weaponSpeed = 15.0f;
        public float weaponLife = 3.0f;
        public float weaponCooldown = 1.0f;
        public int weaponAmmo = 15;

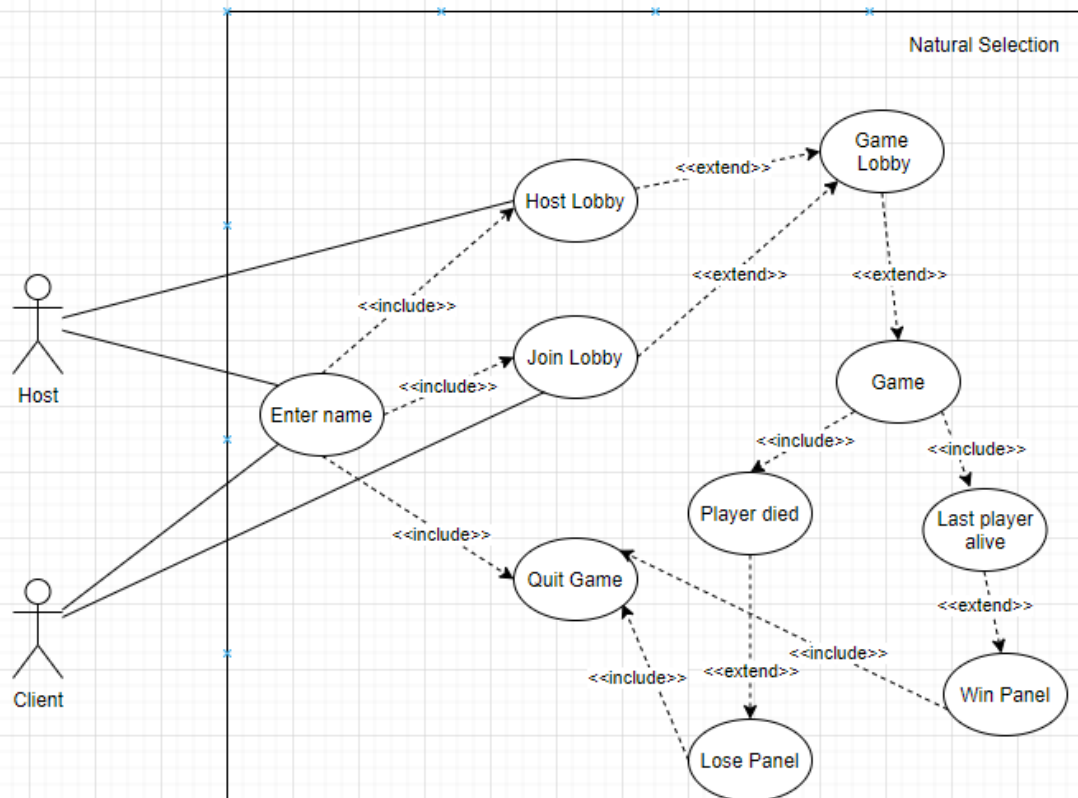
        public GameObject weaponBullet;
        public Transform weaponFirePosition;

        public AudioSource bulletshot;

        public void PlayAudio()
        {
            bulletshot.Play();
        }
    }
}

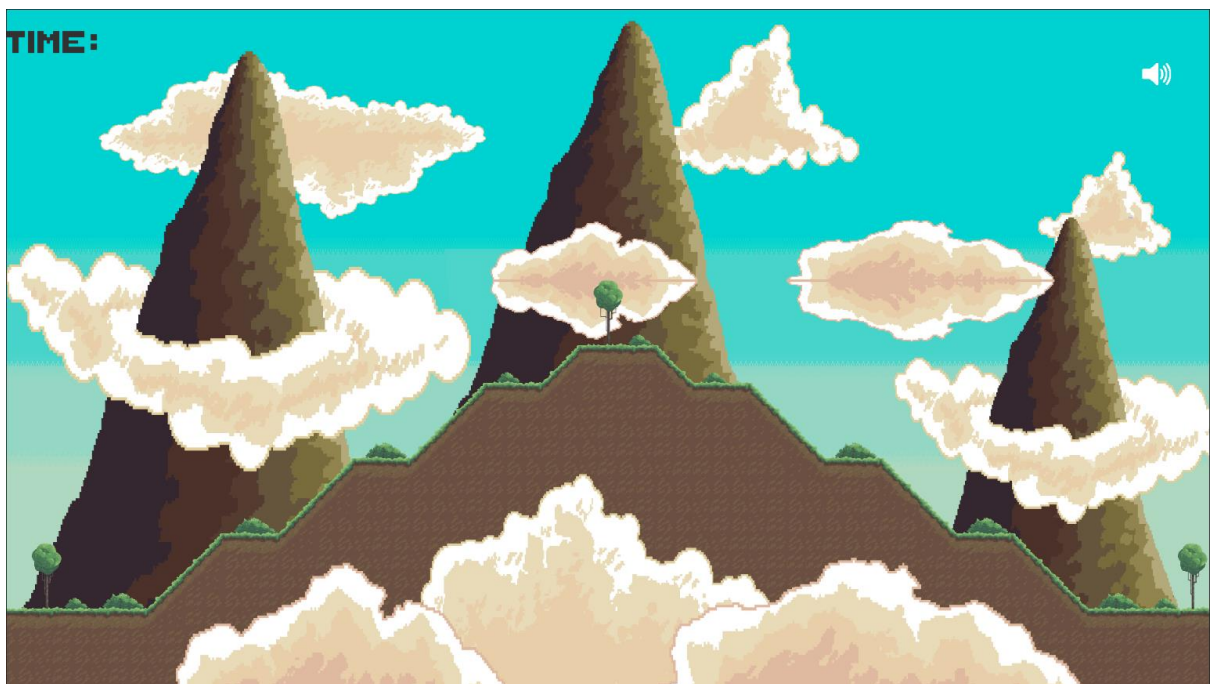
```

## Appendix 17:

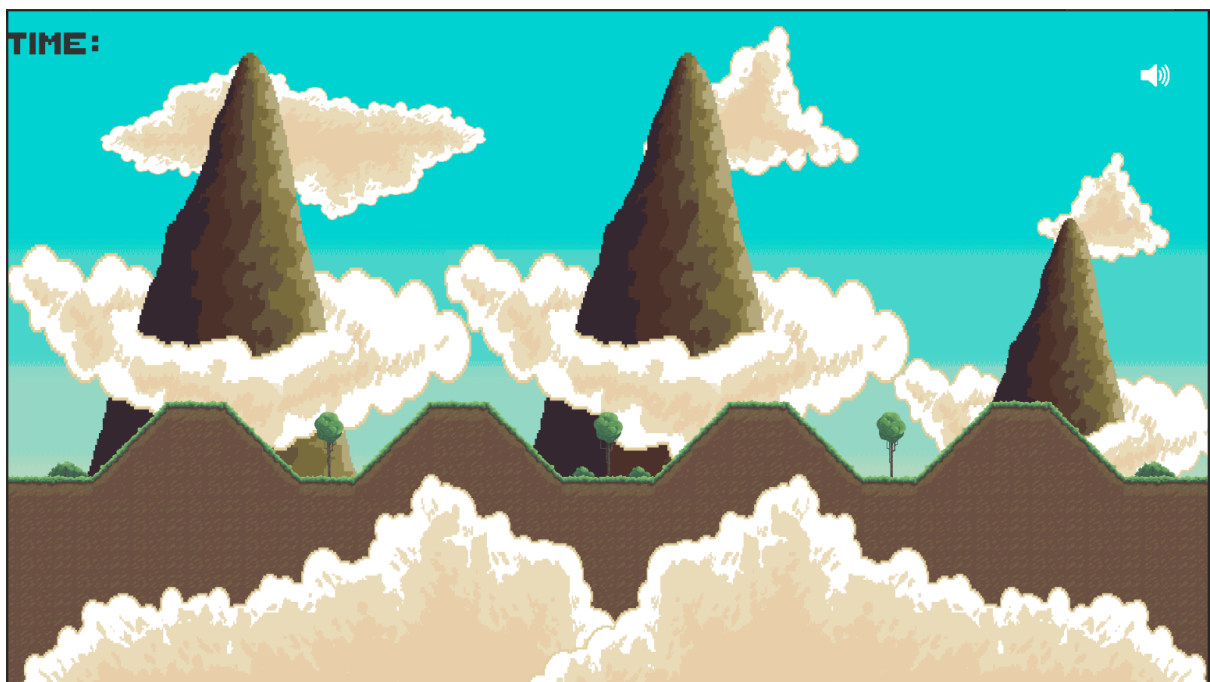




Appendix 18:



Appendix 19:

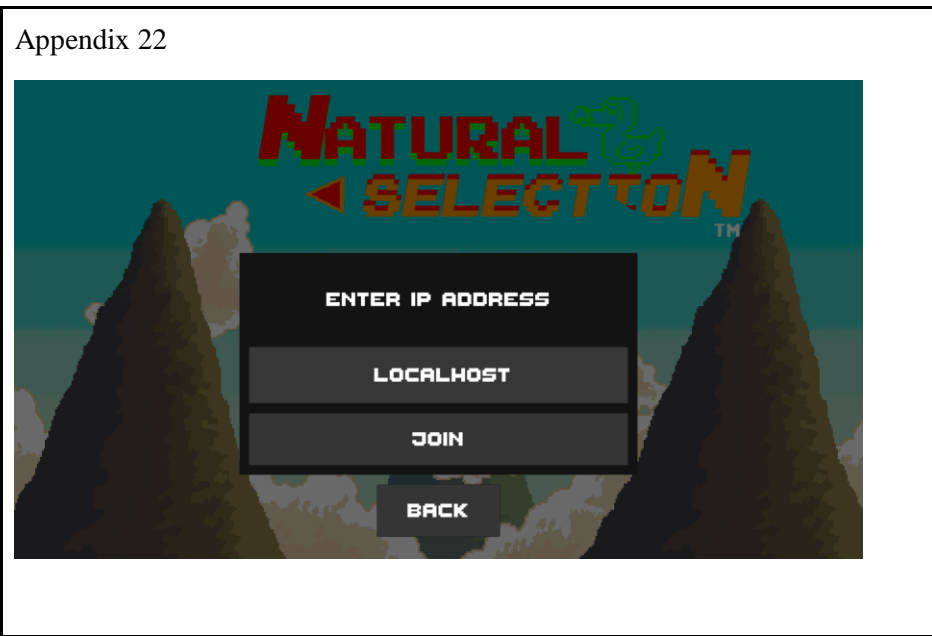
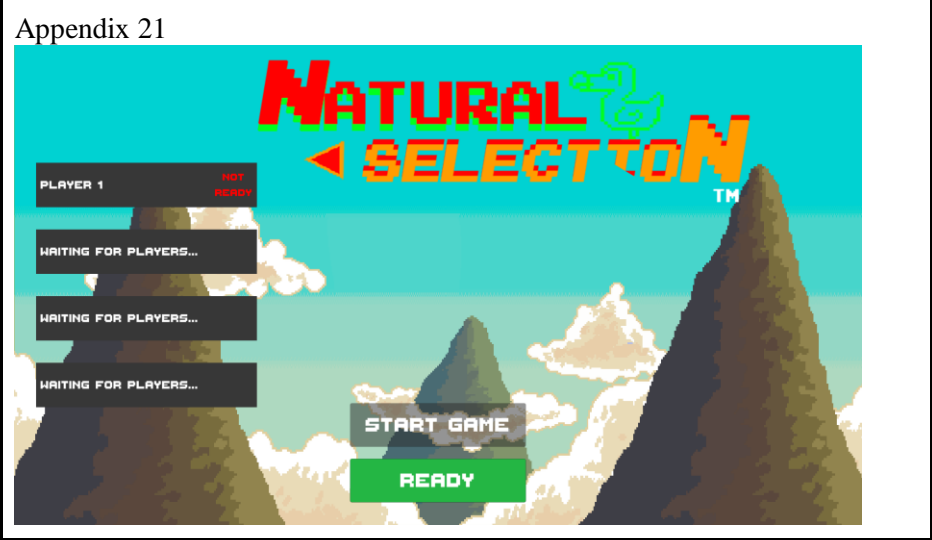


## Appendix 20:

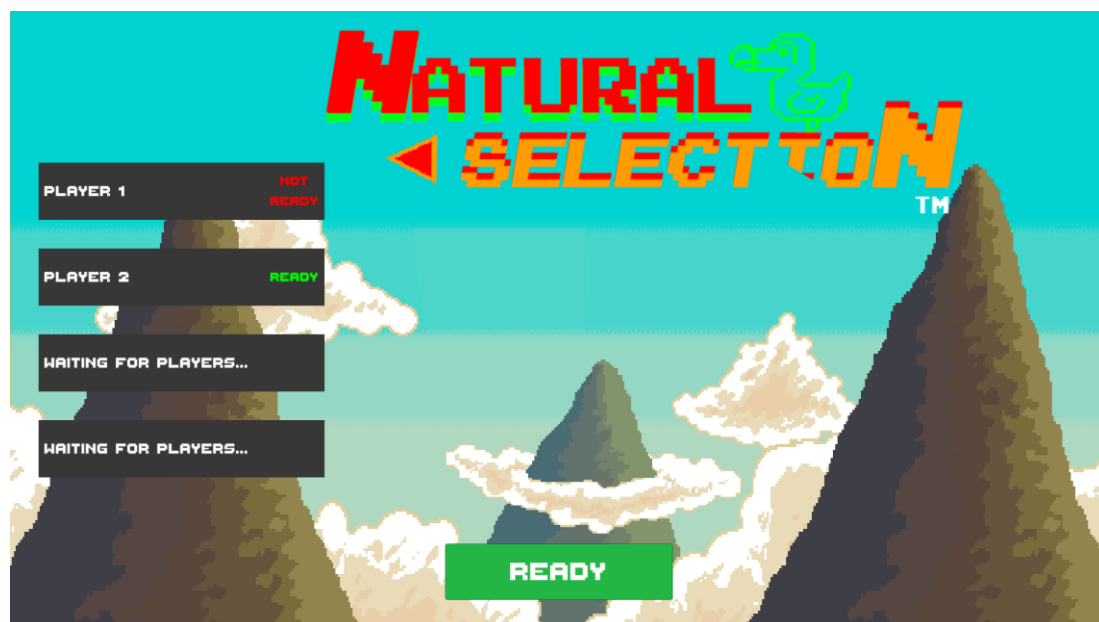
## Miscellaneous Files

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using EZCameraShake;
5 using UnityEngine.UI;
6
7
8 public class ShootScriptNewGun : MonoBehaviour
9 {
10
11     public Transform Gun;
12     public Animator gunAnimator;
13     Vector2 direction;
14     public GameObject Bullet;
15     public float BulletSpeed;
16     public Transform ShootPoint;
17     public float fireRate;
18     private float ReadyForNextShot;
19     public float magnitude;
20     public float roughness;
21     public float fadeInTime;
22     public float fadeOutTime;
23
24     //ammo display variables
25     public int ammo;
26     public bool isFiring;
27     public Text ammoDisplay;
28
29     //ammo and reloading variables
30     private int currentAmmo;
31     public float reloadTime = 1f;
32     public int maxAmmo;
33     private bool isReloading = false;
34     public Animator animator;
35
36
37
38     // Start is called before the first frame update
39     void Start()
40     {
41         currentAmmo = maxAmmo;
42     }
43
44     //stops weapons not working when weapons are switched during a reload
45     void OnEnable()
46     {
47         isReloading = false;
48         animator.SetBool("Reloading", false);
49     }
50
51     // Update is called once per frame
52     void Update()
53     {
54         if (isReloading)
55             return;
56
57         //ammo and reloading
58         if (currentAmmo <= 0)
59         {
60             StartCoroutine(Reload());
61
62             return;
63         }
```

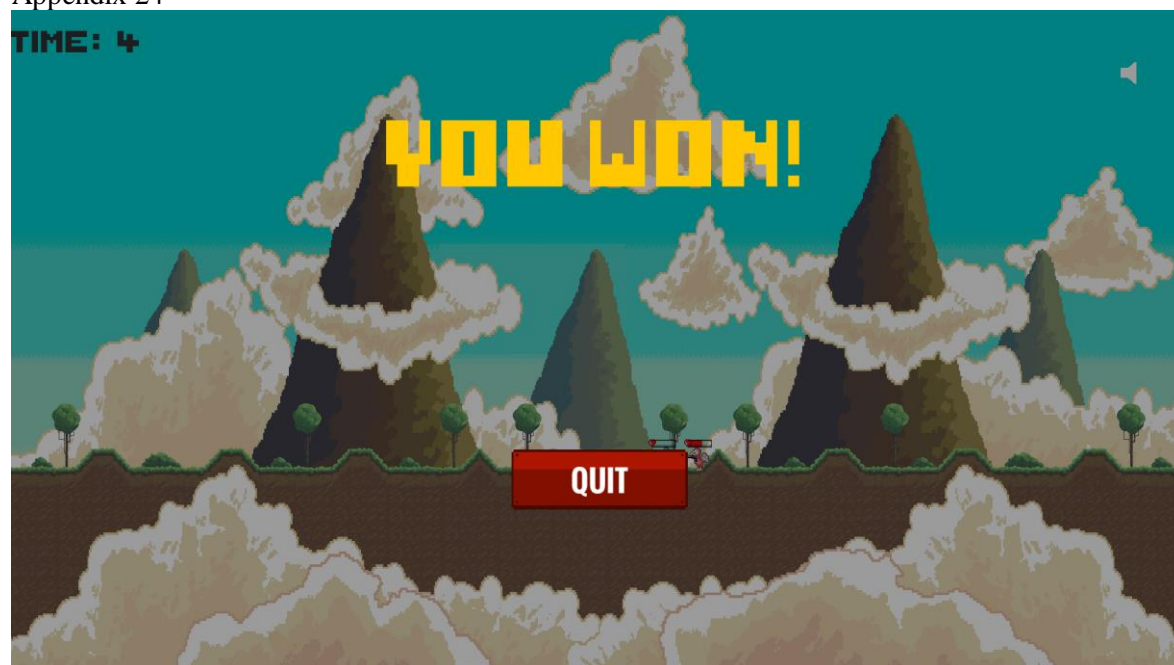
% No issues found



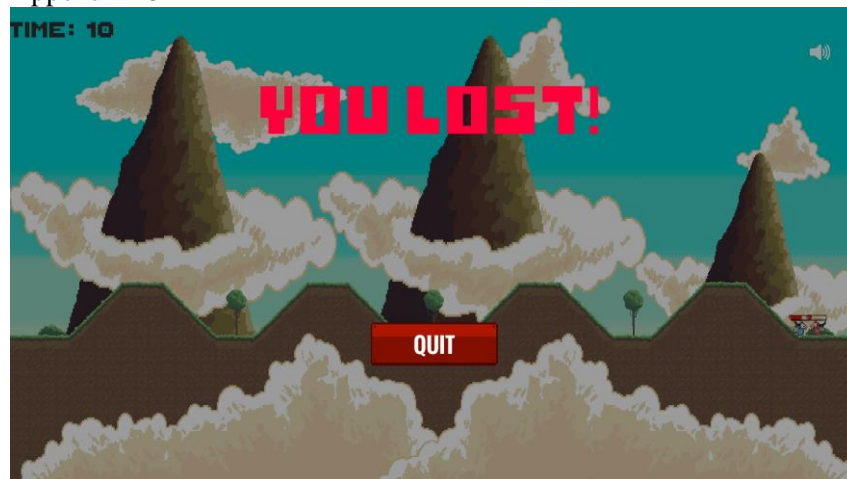
Appendix 23



Appendix 24



Appendix 25



Appendix 26

Appendix 27

