Contents

Contents

Analysis

Idea:

2D fighting game with different characters, each having unique moves and the user will need skill to win the game. This will be an arcade style game, so hardware requirements will be very low since most computers can run a pixel style game quite easily. To develop this idea, I will use C# in unity but I could have second thoughts on what language I can use in the future.

Problem identification

My initial idea for a game that I might develop will allow gamers who have been playing fighting games to experience something that isn't common in the usual fighting games. In some fighting games, a beginner can sometimes beat a higher-level player by just button smashing. This makes the time and effort the other player put into the game to learn the combos and skills to be wasted.

Even though there are game that need actual experience to win the game, at the early stages of playing against others online they will most likely spam and can be very annoying. Its only when you have played the game for a very long time and know how to use attacks at the right time and when to use them on that certain character since they will have to learn the pros and cons of every character in the game. This may take a long time especially since most fighting games have a variety of characters.

So for my project I will try to develop an idea for a game that will need skill and timing since it will make it more challenging for the user to learn then just button smash and hope for the best. I will also try to aim to make the users become a better player by making them learn combos and or just play smart even if they can't learn them. This is will also help them playing other fighting games since it makes them think of the basics and get better from there.

Essential features

Game modes

There will be 4 different modes, story mode where they can go through different chapters with each one being harder. 2nd will be an endless mode where the user has to survive as much as they can with one health bar, no regeneration. 3rd will be training mode where they can try out their purchased or earned skills and attacks from playing the game and using it on a dummy and the tutorial for the game will be an option to choose when they're in training mode. 4th will be a local multi-player game where it will be limited to 2 players fighting each other, they can pick any of the unlockable skills and attacks for free since it's just fighting with another user, but they will be limited

to how many they can choose(story mode will only be done depending on time limitations). These are just my initial ideas so not all of them may be added to the game since it may be need more time to implement or need more experience to do so.

The fighting mechanic

Throughout the duration of the fight the map shrinks so that the players cant wait it out for each other to come in the corners, it forces them to fight each other near the middle. Each attack has a max of 3 uses and will have a 1 second cool down to use again, this will make them to use combo attacks since it will allow them to continue the damage being dealt to the opponent and not just spam singular attacks. While the countdown to use the attacks again after their usage starts, the player can block but it by defending but it can be broken by a guard break that the other opponent can perform. The guard break has to be tested first so that I can decide whether it should have a cooldown.

Buying combos

To make the game feel more enjoyable and rewarding the player will be able to buy new combo skills or super attacks with points that can be accumulated in the game through the different levels. With each level being harder and having to learn new ways to beat the opponent. I will also try to make different costumes they can buy with the points with it being completing a certain mission when fighting or accumulating points.

skills

To buy the skills there will be a store in the menu separately. I will also allow them to get a demo so that they can try out how each skill looks like so they can make a satisfying choice of what they want to purchase.

Computational methods

Why is the problem solvable through computational methods?

My problem will require inputs and outputs that will be needed to play the game. This

Decomposition

Decomposition will be used to break the game into separate subprograms, enabling me to reuse the components throughout the program. This will also enable me to link the modules together to solve complex problems such as physics that the characters will have e.g. movement speed and jump force. This will also give me more time to work on other parts of the game and reference the sub programs to similar problems.

Abstraction

Abstraction will be used on how much detail there will be in the character design since it’s an arcade style game which will have a retro style to it, this type of style will be similar to old street fighter. I can also get rid of 3d physics that may have been used if I was trying to recreate modern fighting games like Tekken 7 or xenoverse. This will save time and memory in creating 3d models for these characters which is more complex than 2d sprites which I will be using. Making code to adjust the

camera to follow what position you want it to be in for certain situations such as movement and attacks is also more time consuming in 3d physics.

Thinking ahead

Thinking ahead will be useful when deciding if the user will like or dislike some aspects of the game. This will give me an idea of what to implement and change throughout the period of making the game. From making some parts of the game I can also be rational and decide if I can add a certain feature to the game depending on how easy or hard it is from the experience I have already been through form making the game. This will be very important since making something that is beyond my capabilities could result in a waste of time.

Back tracking

This method can be used when the user wants to go back to a certain level to play for fun or achieve a certain mission to unlock a new equip from the store or receive a new costume. This can also be used when the user is going back and forth between different modes and options that will in the game. It can also be used when developing the game because I can go backwards in a certain module and find what is the origin of the error when I face them making the game.

Heuristics

This method can be used to make new algorithms that can be implemented into the game that won't be the best solution but it is satisfactory enough to work. Then it can be developed further in the future till it performs at an optimum level. This will help to make a the base layer of the game and work on top of it to make amendments that my users will ask for from playing the game.

Stakeholders:

Client and users

Possible stakeholders that would be interested in playing this game can be teenagers who want to get into a skill based fighter games. Gamers who have experience on fighting games and want to learn new skills that is required to be good at the game. Even children who want to just have fun playing the game with their friends.

The users that will try out my game will be 6th formers who will be some of my classmates who are also working on making a programming project and people from my school who play games often. The users won't have a lot of time to play the game since they will be doing school work or other activities so they will have a limited amount of time to play my game. This means they would not be able to reach the max level or buy all the additional features for all the characters which could be useful for the feedback, unless they really enjoy the game and want to play it in their free time.

Competitors

Competitors that I will have to go against are companies such as Bandai Namco, Konami, Sega that are know for making fighting games that gave been in the industry for a very long time and have millions of users. Another competitor who isn't well known for fighting games but has been recently making quality content is arc systems who are mainly know for guilty gear and dragon ball fighter z. They may be also a potential threat to these companies but for me they are all way more successful to compete with because of their experience. If I wanted to compete, I would have to make my

game stand out from cliché fighting games and make a team to further improve game quality and get regular feedback to make a game that meets customer needs.

Interview

Before making the game I would have interviews with several of my clients and then ask them questions that will be appropriate in developing the idea for this game.

I would recommend them to play by themselves to give me feedback on how good the game is and to make suggestions to improve the game. Then I would ask them to play with a friend or family member to then tell me if its more enjoyable and was there any problems when playing 2 players.

Then with this feedback I will try to fix the issues that have been identified by the users and improve other aspects of the program that they like. This process will be repeated until it is good enough to play. This idea can't be improved to perfection since I have a limited amount of time to make the game and test, If I was to perfect it would take much longer and would need constant maintenance and feedback.

Interview questions

1. Have you played fighting games before?
   If the answer is yes, I will ask them what aspects do they like the most and not like.

   If they say no, I will give them some examples, how fighting games are like to play and other background information on this style of games.

2. What kind of themes do you like in a game?
   This question will help me decide what style the game should include to satisfy the client as they will want to play something that they will be familiar with.

3. How important is character customization to you and why?
   This question will also help me with how much design I have to do to meet the client's standard if they care about customization and if they don't then I could make simple skins that will save time when developing the game.

4. What type of music would you like while playing the game?
   This will inform me what they will want instead of me just adding something I like which the client might not be satisfied with. This question also helps with getting the client to know bit better as it also shows what type of theme they like in a game as music represents their personality.

5. Do you care about story mode in a game and why ?
   This will give me an idea how much effort I need to put into the story mode to satisfy my client if they care about the story, if not I will ask about what they care about the most in a game if they haven't played a fighting game or ask the same question regarding a fighting game.

Brandon : interview 1

1. **Have you played fighting games before?**
   'I like the variety of combos and the characters you can choose to play as, and the themes in general. I don't like how you can spam and win games since its boring and requires no skill. '

2. **What kind of themes do you like in a game?**
   'I like colourful themes as it lets you escape from the real world and lets you experience things that isn't possible in the real world. It's also enjoyable as bright themes are a good environmental to play in. '

3. **How important is character customization to you and why?**
   'It's quite important to me as you can get a variety of options to play as what you like and adds diversity to the game. it also adds another purpose then just completing missions since you can play to earn something you want. '

4. **What type of music would you like while playing the game and do you want all of the music already unlocked or earnable through points?**
   'Music that gives you good vibes and is upbeat because it makes the game exciting especially in fighting games as it helps you pace yourself when playing against an opponent. I would want some of the music already unlocked so that I will be able to get a feel of how it is then I might be willing to purchase them by points.'

5. **Do you care about story mode in a game and why ?**
   'Yes, as it gives you another option then just playing multiplayer and people sometimes prefer single player as they like playing by themselves. it also gives you a break from just playing competitively against others online if you play a game that has online multiplayer and it just fun as its entertaining to play through especially if it has good animations.'

Analysis of interview 1:

1) From the first question Brandon I can see infer that he has played fighting games before and is experienced in playing them since he states that he likes the “variety of combos”, gamers who practice landing combos in the game will talk about this. He also states that that he likes the characters you can choose to play as, this means that I should try to add a variety of characters but it since this will be difficult I will only be able to a few, it mainly depends on time limitations. The main point is that he doesn’t like aspect of spamming, also says that it boring and requires no skill. This is perfect since I will be trying to make a game that will require skill and experience.

2) The second question does give an insight to what he likes which are, “colourful themes”, this means that I would have to make the game fun and not violent. This would have to apply to the design of the game mixed with the fighting style that I want to add. information after that doesn’t give me too much information other than they would like to play in a positive environment.

3)The response to this question shows that character customization is important and likes it because of how it adds “diversity to the game” and “adds another purpose”. This means that I will have to make the ability to customize their character quite versatile and meaningful.

4) From this question I would have to add non-copyrighted music that will be “upbeat” and gives “good vibes” to my client. I would also have to enable the client to choose what music they want from the menu since they stated they wanted some of the music already “unlocked”.

5)The information from this question shows how important story mode is to the client since he clearly gives their opinion on how it give him a break from ,“online multiplayer”, as it is mostly , “competitive”. The means I would have to work on a story mode that would be ,”entertaining “, since they want it to be however they also want ,”good animations”, this will be time consuming since I would have to make the animations by myself or take them from online. Either it will take a quite a while to implement.

Questionnaire

These questions will be asked in trying to develop my initial idea:

Tick option that you prefer

1. **Would you want to play a game that has simple mechanics or complex?**

   This question will give me an idea of what type of mechanics that I will need to implement into the game to satisfy the client.

   Simple

   Complex

2. Would you rather win a match using actual skills or just button smash and hope for the best?

Button smash

skills

From this question I will be able to see how many of my audience care about learning than just button smashing, It will also decide what type of game it will be.

3. Would you prefer hand to hand combat and special attacks or just hand to hand?

Special attacks

Hand to hand

This question will save time to implement special attacks, giving me more time to work on other parts of the game.

4. Do you like the idea of a retro fighting game?

Yes

No

This will inform me if my audience like retro games, if not what other themes should I include to meet their needs.

5. Do you like playing games by yourself or with others the most?

Yourself

others

From this I will have to either make an AI based game or 2 players, if they like both ways then I would need implement those options.

6. Would you like to buy skills and costumes by accumulating points or completing missions?

Accumulate points

Complete missions

This will decide if the additional equips or to be bought by points or unlocked through missions. Depends on question 3.

7. **how often do you play games?**

1 hour or less in a week

1 hour every day

More than 1 hour everyday

The answer from this question will let me see how many people play games regularly and how much user feedback I can get from them to help improve my game.

8. **Do you have any visual impairments?**

Yes

No

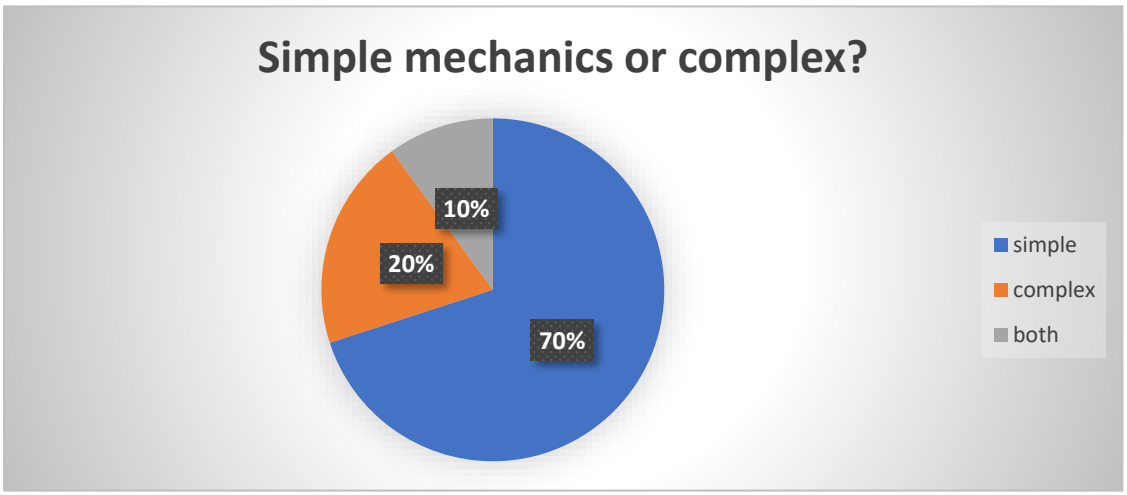
This will let me know if I need to adjust display settings for some users after I make the main base of the game since its not that common among people.

9. **Is personal performance important to you in a game?**

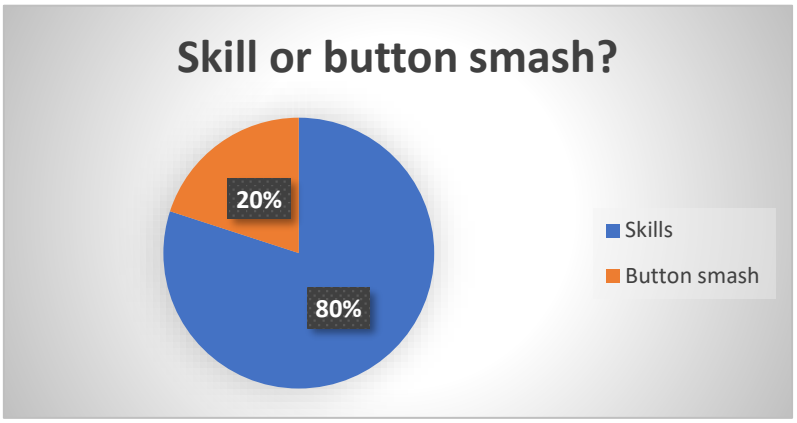
Yes

No

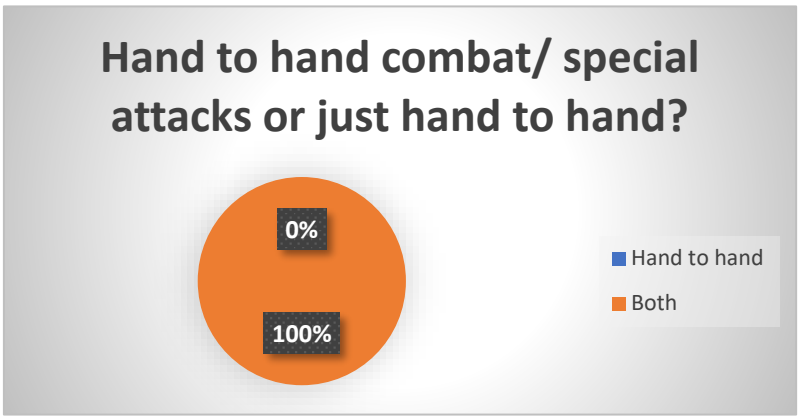
From this question I will have to implement a record of the player history from the fights in the game. Example wins, losses, points earned and achievements.



From this question I can see that 70% of the of the users like to play with complex mechanics so that means I would need to implement some type of way that requires them to think and react actively instead of having a simple fighting game style that you can just wait out for the other player to come to you. It also means I have to prioritise my time making a complex fighting mechanic since it is complicated to implement in a fighting game.



From the response of this question I can see that 80% of the users said skills, even though there are times where all players tend to button smash when they first play so I would have to make a tutorial that would explain to them how to play the game. This data may also be inaccurate as they might have just ticked skills to not be judged however, I did give these to a group of experienced gamers. This mean I would have to make a game that will require the players to think.

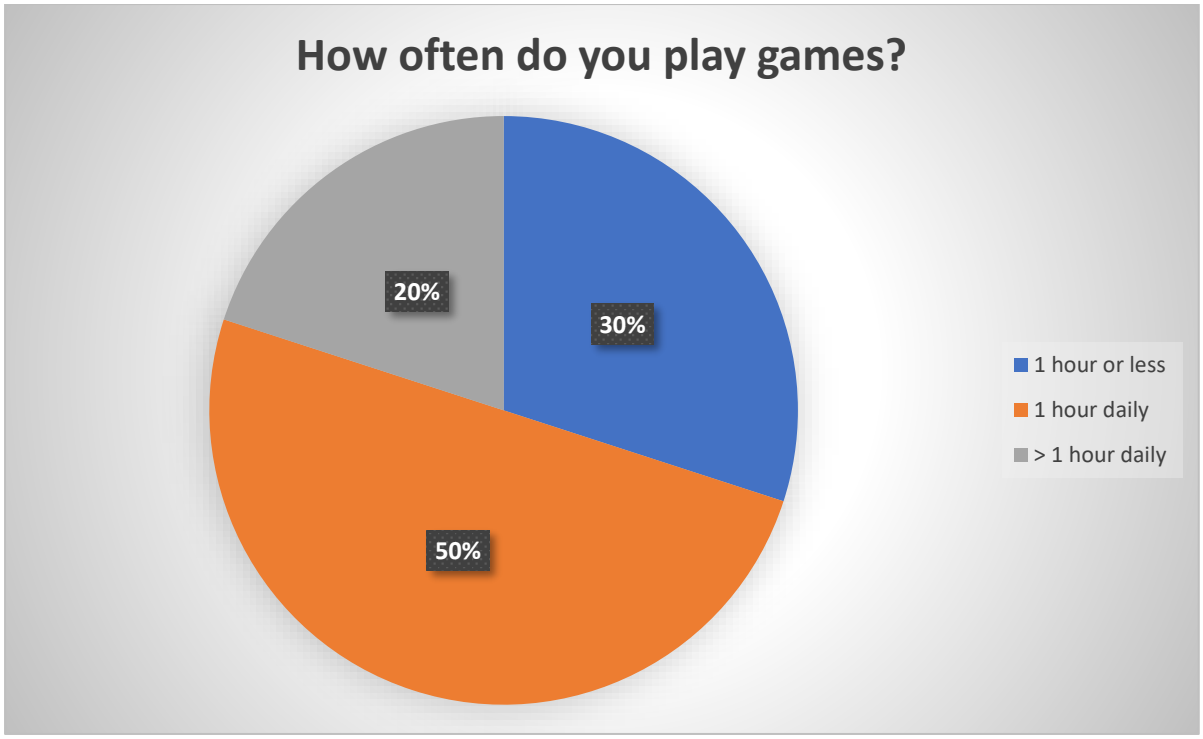


This question was surprising since some gamers like a real life fighting style to a fantasy one regardless it also means that I will be able to make the game more exciting and enjoyable to play. However it will mean I will have to spend time animating the super attacks which will be time consuming. I will try to make the super attacks simple but fun to use at the same time so that I can save time and satisfy the users.

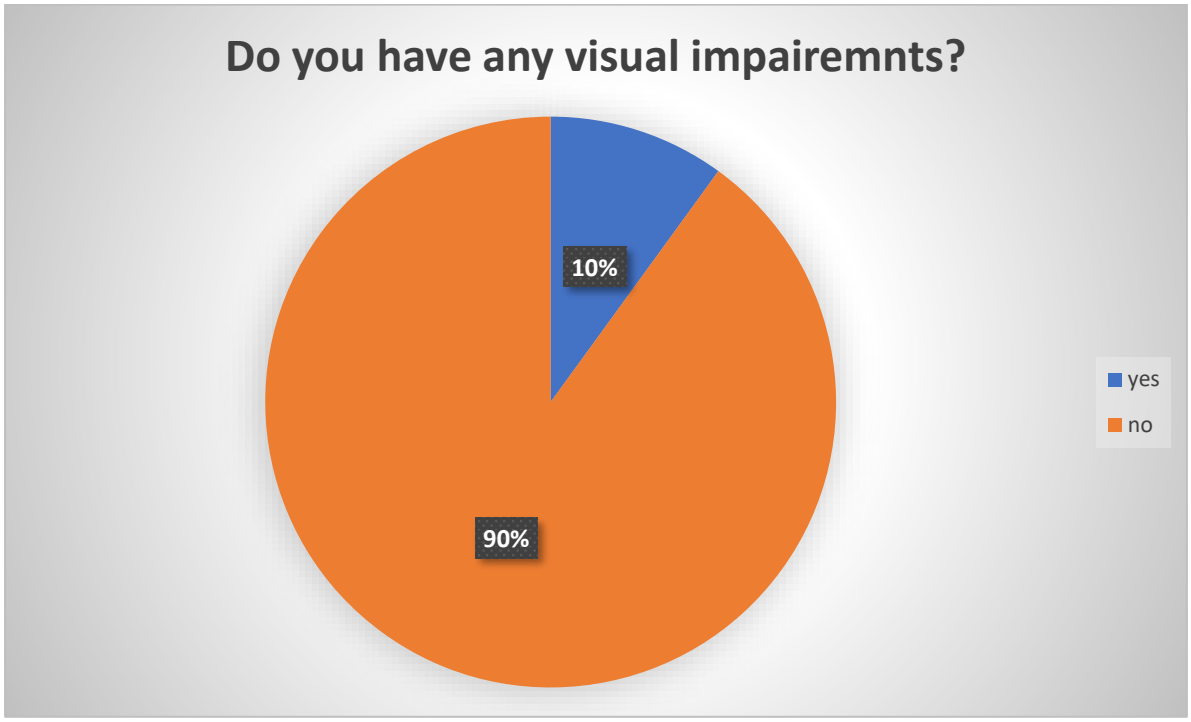
From this question 80% of users said yes to an arcade style game, that mean I will have to make the background and use sprites an arcade style theme. This will also mean that the game will be easier to play on their computers since a arcade style is usually 2d and has a pixel art style to it. This will help with the game running smoothly and their feedback will be more positive after they play the game.

50% of the users said by themselves and 10% said both, meaning I will prioritise my time focusing the game to be single player, however it was a close call so depending on time limitations I will try to also add a 2 player feature for the 40%. This means for now I don't have to worry about multiplayer, and I can save time to focus on the main parts of the game.

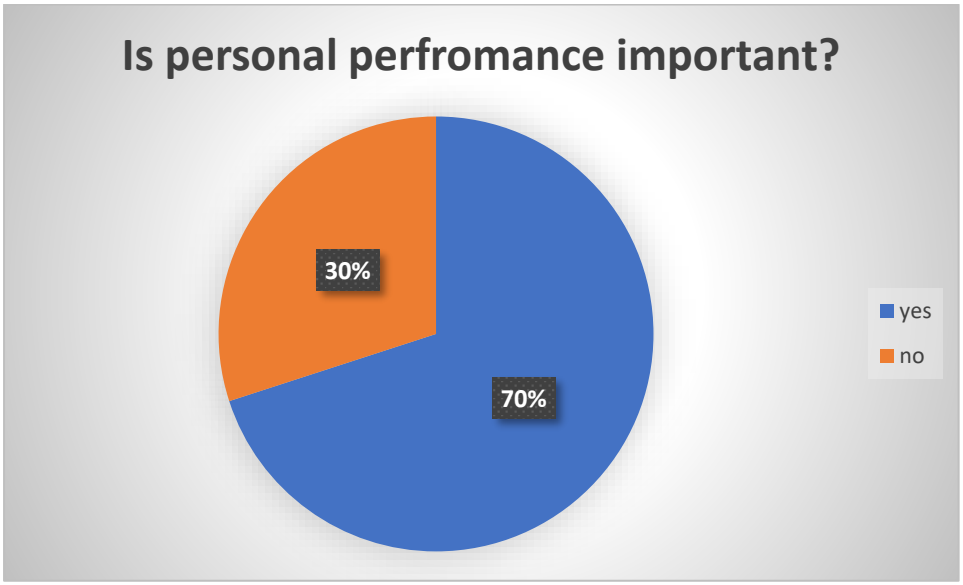
This surprised me since I thought that there will be an even tie between the 2 options, however 70% wanted the option of earning points to acquire the equips that I will include in the game. I will have to implement a score tracker for the user to keep track of their points and mention how much points they will earn for each level.



The 30% who don't have much time to play games will be busy with their studies and other activities. 20% who play the game daily for an hour or less day are most likely casually gamers who balance their time with their studies with their free time. The 50% are the people I will be trying to get to play the games the most since they have a lot of time to test my game in their free time and I will be getting the user feedback I need rom them.



The 10% was expected since not many people will have visual impairments the 90% will have no problem playing my game. If I get the opportunity to develop extra features into the game, I will try to add the choice to adjust the colours in settings to match the users' visual impairment. This is will also add a complexity to the game and present other opportunities to improve the game for certain users.



With the 70% caring about how they perform in the game, it tells me that I might have to add an option in my game to record their performance such as the number of wins and losses they have in the game, and how many points they have accumulated. This will satisfy user needs to keep track of their time in the game and how they are improving.

Existing solutions

Dragon Ball Fighter z:

Description:

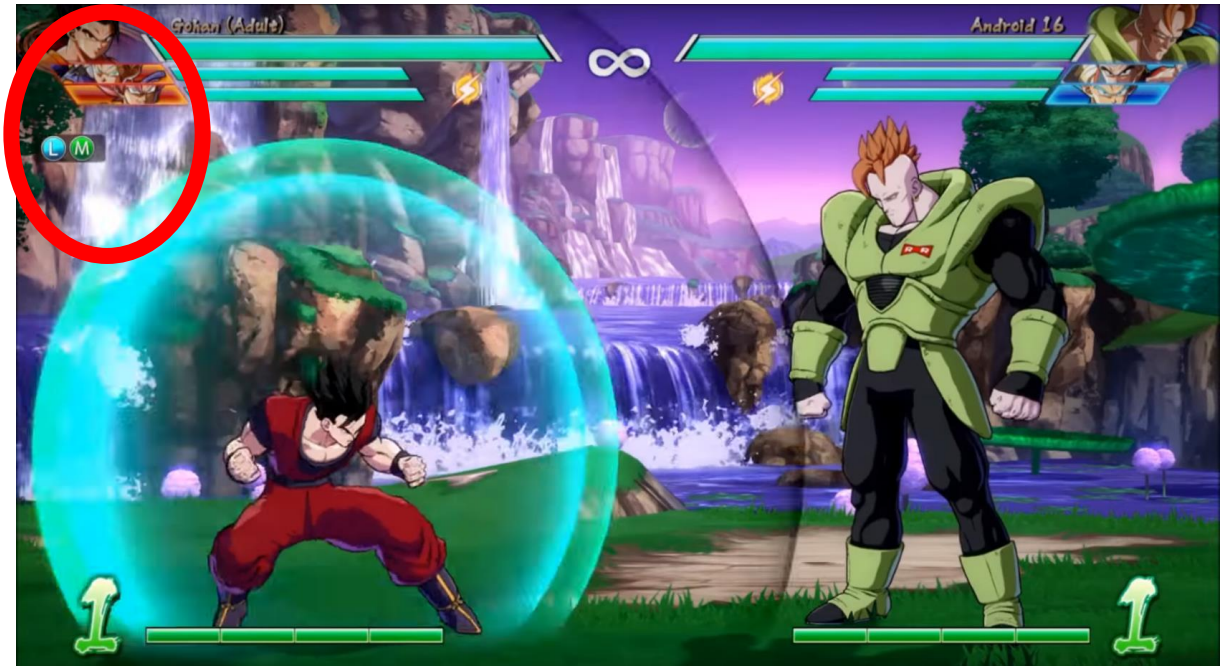
Dragon Ball Fighter Z is a 2.5D fighting game that has a story mode and various multiplayer modes such as ranked, casual and local and clearing waves of enemies. It has multiple characters each with their own unique combat movement and special attacks. This makes the player really feel like their playing the original character from the show. When playing the game, you can pick 3 characters to play with in the fight which you can swap out of after their cool down has finished and even combo with their super attacks and assists which helps pressures the opponent when trying to break their guard. To win the game you have to bring the HP of all the three opposing characters to 0 by using all of your 3 characters.

Notable features:

A feature in the game I really like is the ability to change between characters while fighting with the opponent, this helps with the pace of the game as it can make the experience really tense and exciting. I might also borrow the ability to break through the opponent's guard with 'dragon rush' which does damage on impact, depending on time limitations I might also implement the feature that allows the player to swap to another one of the opponent's character after the guard break. The purpose of this is to either fight another character that is less of a threat to the player or fight

against a character that has the least HP to have a higher chance of beating the opponent by decreasing the character count to 0.

The problem that I think this game has is that having 3 characters makes the fight too long and that causes people in servers to wait for a long periods of time to get into a game with someone online. I will reduce the character count to 2 to make the game finish a bit quicker which will also save time implementing an extra character. Another problem is that it not very user friendly in my opinion when going through all the skins in the game since there is so many. I will try to improve on this my making mine simple to use and to navigate.



The green sphere around the character indicates the guard break/'dragon rush' that allows the player to engage into the opponent and break their guard.
https://www.youtube.com/watch?v=4wjy-vezNbM

In the red circle is where the 3 characters are where you can choose to swap with while fighting and each have their own health bar.

JoJo's Bizarre Adventure Future Heritage

Description:

This is a 2D fighting game that has a variety of characters from its show, it has story mode and challenge mode that lets you fight with the same 1 bar of HP against different characters each round seeing how you can survive with your skills. Its originally an arcade game that has allows you to set your highest score in the challenge mode to compete with other players. Each character has a special power called a 'Stand' that appears in front of them that each have a unique ability respectively to the user's personality. The player can make them appear and reappear whenever they want but it will be out of use for a few seconds after the opponent's stand break yours after a certain number of hits, this makes you vulnerable to increased damage.

Notable features:

The features that I can implement into my game is the 2D sprite style that the game displays, even though its only 30 fps, the fighting movement is very smooth. The backgrounds are all related to the different scenes from the show, this can be used to implement to my game by making backgrounds that are related to the different characters that will be in my game. After making the backgrounds I will enable the player to choose what background they want before after they choose what character they want to play as.

A problem that I faced when playing this game is that it doesn't allow you to choose what map you want to play in, this is a shame since there are lots of memorable maps from the series that players would've liked to choose. I will try to solve this by adding a map screen that enables users to choose what map they want to play in.

These are the stands just to be clear, and the ideas that I will be borrowing from this game is the 2d style and the background is a prison which the character on the left wanted to stay in since he couldn't protect others from his 'stand'. I'm mentioning this since I will also be making backgrounds in my game that will relate to the characters.

Super Smash Bros:

This is a 2.5D knockout game that has multiple characters from various franchises that you can choose from to fight with. There is story mode, online and local multiplayer, training mode and some other mini games like home run contest and target test, etc. The damage is based on percentage, the higher the number the further you will knock your opponent into the air when you hit them. Landing a smash attack on low percentage players doesn't push them very far.

Notable features:

The game allows the player to choose what music they want when playing the game, I am also thinking of implementing this idea to my game since playing with no music or attacks sounds will be boring and not as exciting. Another feature which I might use from this game is the option to use items that fall from the sky to power up your character or give you an edge in the game. The reason why I'm not sure is I don't know how complex it might be to add items to the game.

Just like this bat there are many items that will be useable to fight other opponents with, this is the idea that I will be trying to add to my game.

Limitations

Possible limitations that I might encounter is the time to make all the essential features such as story mode since it will need to have dialogue and a story script which in my opinion will take a considerable amount of time to implement, this also applies to all the other modes that I discussed in the essential features. This will potentially impact how the game feels, especially the fighting mechanics since I might not have enough time to perfect this due to me working on the story, in return not being able to solve the problem that I set out to do in the beginning of the project.

Another limitation is the complexity of making the fighting game since the are various moves that will have to be animated and made for each frame, this will be difficult to make and therefore time consuming. So I would have to make a game that is simple and make animations that are less complex. This would impact the completion of the fighting mechanics and there will be only a prototype left over.

Success criteria

Criteria	Detail	Evidence
For the player to be able to use the menu with ease	**The menu will be a simple layout where they will be able to easily switch one to another**	**Screenshot of the menu layout**
For the player to pause the game whenever they want	**While fighting if they want to pause there will be a pause button in the game, when paused there will be a list of missions that they will be able to complete in that certain level. The menu will also be there for them to quit the game.**	**Screenshot of the pause menu**
For the player to quit the game whenever they want		**Screenshot of quit being displayed in console**
Customize characters	**There will be a separate area in the game where they are able to customize their characters through completing missions and earning points.**	**A screenshot of the area in the game that will allow customization**
To be able to play the game with simple inputs	**The game will include simple keyboard inputs that the player will easily learn**	**Gameplay of the tutorial of how to play the game**

Accumulate points from playing the game	**The game will have a point tracker and will inform them how many points they will get from that level**	**Screenshot of point tracker**
Music in game	**The game will have an option where they can choose what music they want to play while the game is open**	**Screenshot of the music list the player is able to choose from**
Selecting backgrounds/ maps	**There will be an option for the player to choose what background they want to play in**	**Screenshot of the backgrounds that the player chose from before playing the game in certain modes**
Key binds	**This will enable them to assign what key each attack should be to their liking**	**Screenshot of the Key binds screen**
Selecting characters	**Players should be able to choose what character they want to play as before entering a match**	**Screen shots of player 1 and 2 screens for selecting characters**
2 player support	**The game should be able to allow 2 users to play the game and supporting the appropriate controls when in a match**	**Showing player 1 and 2 moving/attacking in a match**
Health system	**Players should be able to apply the correct damage value corresponding to what they input**	**Screenshot of health system decreasing**
Timer for certain attacks	**Attacks such as super attacks should be set a timer for it to be used again instead of allowing the player to spam it**	**Screen shot of inspector of the timer for super attacks**
Input limiter depending on timer	**Since my game is aimed at countering spam, I will be setting a limit of how many attacks the player can input so that they will have to think about how they will attack instead repeatedly using them**	**Screen shot of the code that controls player inputs**

Solution requirements

Final solution will be required to:

Take user input and display animations for the character their using, apply damage to the opponent and be restricted from button spamming. This is essential since my problem is to solve button smashing and make the user play with skill.

The final solution requires the player to be able to fight against the their friend in the game. They should also be able to train by themselves with a dummy, this is important because it will help them become a better player and use the basics element of fighting in other games.

The user should be able to customize whichever character they choose from the game from earning points, and unlocking super attacks using the points. They should also be able to have a trial run on how the super attacks looks like before they have purchased it.

Hardware requirements:

The user will need a PC with a monitor and keyboard (mouse is recommended) or a laptop. The main input will be done with the keyboard since that's where they will be playing the game. Some kind of display is essential for the game to be displayed. The pc or laptop should have a dual core CPU, RAM would have to be around 2gb. This is just an estimation since it's a simple 2d pixel art game that doesn't need a lot of power to run. After the game is finished is able to run I will state the exact requirements.

Software requirements

The game will need to run on windows or macOs, this is will be the common operating systems that most people will have and its used for gaming. The graphics that is already installed in pre-built pc's should be enough. In custom built pc's the drivers should be up to date but they will most likely be able to run the game with no problem since they have really good hardware and software since its custom built.

Minimum software requirements to play game

Desktop:

OS: Windows 7 SP1+, macOS 10.12+, Ubuntu 16.04+

Graphics card with DX10 (shader model 4.0) capabilities.

CPU: SSE2 instruction set support.

https://unity3d.com/unity/system-requirements

Design

Decompose problem

I'm decomposing the problem to work out which part of the game I should develop first. This way I can prioritise my time working on the most complex parts and then go on to work on the least time consuming. Separating the program into sub problems also allows me to track through what has been completed this helps me to manage the tasks I have to finish the game.

The structure of decomposing a problem also helps with how the game menu and option will help me to make the displays that should be seen by user when clicking the buttons.

1)**Menu system:** The menu is the first thing the user will see when they load up the game it will have the single player, customize, options, training mode, story mode and 2 player (story and 2 player depends on time limitations).

1) **Training mode:** This will be where the user will be able to train against a dummy.

    i) **Choose Character:** this is where the player will be able to pick which characters they want to play as.

    ii) **Skills:** This allows the player to choose the skills and special attacks they want to equip to their characters, all skills will be free in training mode.

    iii) **Maps:** This option allows the player to look through the maps that I will add into the game and choose whichever they want to play.

2) **In game menu:** There will be a button in one of the corners of the game that will allow the player to pause and use the in game button, change settings such as sound volume or music, and quit to return to the main menu.

    a) **Quit:** This button will allow the user to quit and return to the main menu

    b) **Resume:** This button will allow them to return playing the game after whatever they wanted to do after pressing the pause button.

3) **Single player button:** This is where the player will be entered into single player where they will be able to go through different levels.

    a) **Character selection:** This is where they will choose which character they want to play , all characters will be free to use but the skills and special attacks will have to be unlocked for each of them through earning points by playing the game.

4) **Character movement:** The chosen character will be controlled by the inputs form the keyboard. Which are ‘A’ and ‘D’ for left and right, ‘W’ and ‘S’ for up and down. I will also try to add a double dash animation when the user double clicks the ‘A’ or ‘D’ inputs as it will make the fight more intense since any of the players can close in on you at any moment.

   a) **Attacks:** The user will be able to attack with ‘I’ for punches, ‘O’ for kicks, and ‘P’ for super attacks. Punch damage = 5, Kick damage =5 , super attack damage = 10 to 20 depending on how it will affect the game while player are fighting since it can end the fight too quick so I will put a limit to how many super they can do. Total HP will be 100, the numbers for the attacks are just an example of how the attacks will work.

   b) **Defend and breaking guard:** The user can block their opponent’s attacks using the ‘K’ input and they can break someone’s guard/ defence with the input ‘L’.

   c) **Character swap:** The player will be able to swap between the 3 characters they chose to fight as while fighting their opponent, the input ‘J’ will enable them to do so. This button has a 1 to 2-minute cool down depending on the development of the game.

5) **Enemy movement:** The user will face against characters that will have a set of moves that will be repeated with different timings, this will make the game more challenging and fun then just attacking the enemy.

6) **Options:** The users will be able to change some settings in game.

   a) **Music:** the user will be able to change the music to what they want, I will try to give them a few options to choose what music they want from the preferences they gave in the interviews
   b) **Volume:** The user will be able to control the volume of their game by using a scrollable slider that will be horizontal.

7) **Customization:** this is where they player will be able to customize their character.

   a) **Choose character:** The player will be displayed with the characters that will be in the game, they will able to choose which character they want to customize.

   b) **Skins:** The player will be presented with different skins for their character they will be able to purchase through points

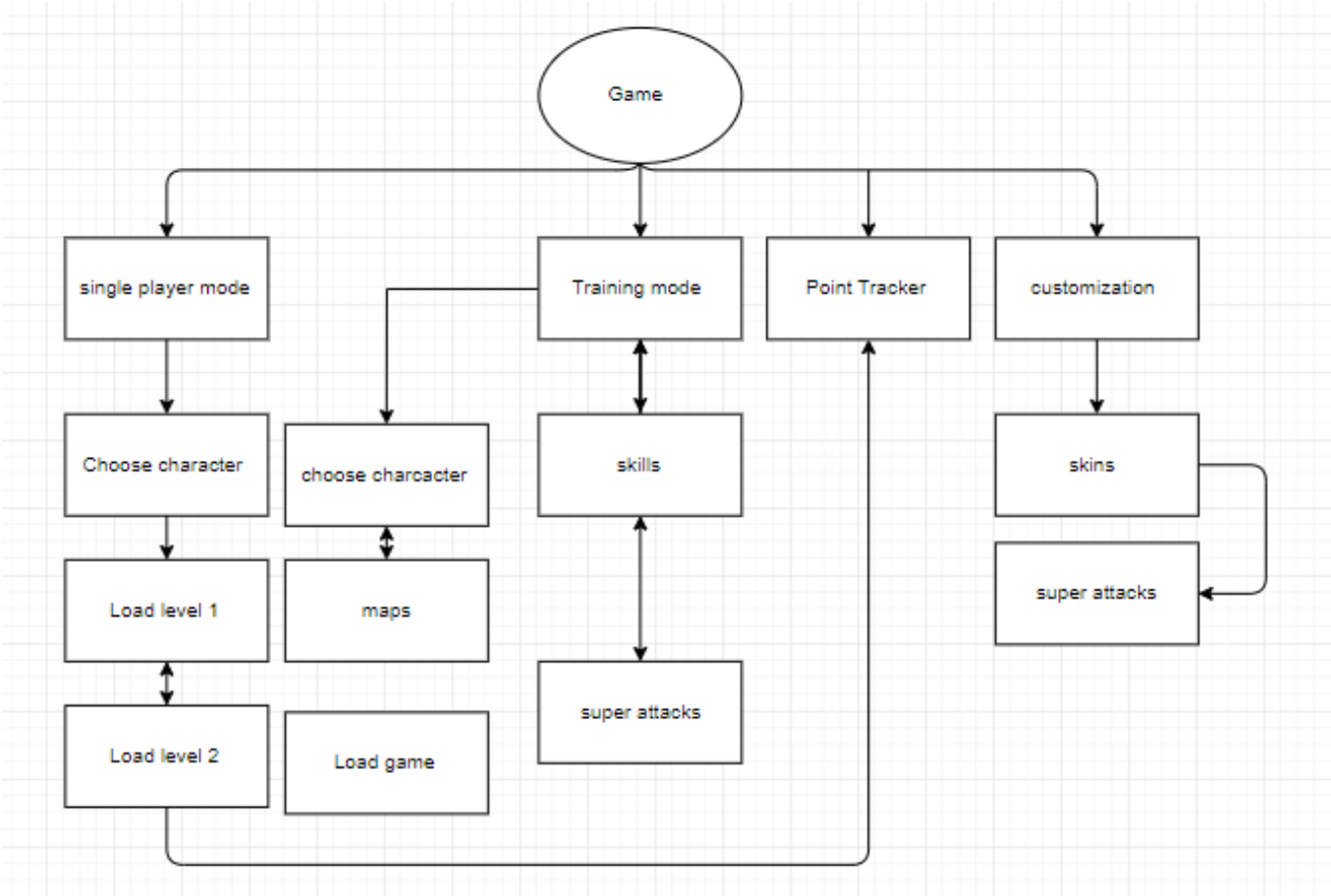
c) **Super attacks:** This button will give the player the options to see all the super attacks

i) **Demo:** This button will make them switch to a fighting stage where they will have a trial run to test and see if they want to purchase the super attack by playing with it
ii) **Return:** This button will return them to the super attack section of customization

8) **Point tracker:** This is where the user will be notified of how many points they have from playing the levels in **Single player mode**, +this will be displayed in the load screen and customization.

Overall layout of game

This diagram shows the layout of how the game will look when the user loads the game. The first row that you see after "Game" you have 4 options to choose from. Single player mode where the player can choose what character they want to play as and will be taken to the 1st level and to the next if they win. Training mode will have to 2 buttons to choose from which is choose character and skills which lead to super attacks. This is where they can use choose whatever equip they want in training mode but will eb limited to how many they can choose from. Then there is the point tracker which takes data from single player mode and adds the points according to the tracker by the amount of times the player has cleared the levels. finally, the customization option where players can but the skins and super attacks by the point they earned.



Game pseudocode

Variables:

Global variables:

- **Areas:** This is an array that the user will be able to navigate through the game using the these buttons**: [Single player,Training mode, Options,Customiztion]**

- **Chosen:** This will decide which character the user will play as

- **Characters:** This is an array that will contain all the chaarcters that will playable in the game

- **Player1:** This variable will be used to refer the first character chosen to play as to assign the all the attacks that the user would want to the character to do when playing the game

- **Player2:** This variable will be used to refer the second character chosen to play as to assign the all the attacks that the user would want to the character to do when playing the game

  The function of the 2 varaibles above will also help with swapping the characters in and out

- **Horizontal:** This is will have the 2 buttons which are 'A' and 'D' which will enable the player to move backward and forward in the 2d view

- **Vert:** Like above but its for jumping('W') and crouching ('S')

- **Cooldown:** This stores the amount of time the player will have to wait before they use an attack. After 3 times using the punch button the player has to wait 2 seconds, this is the same for kicks. super attacks will be reset after 10 seconds after being used once. This is also used for swapping between characters in the game since the player shouldn't be able to swap continoulsy when playing or it will go against the purpose of the game which is not to spam(After swapping out once they player has to wait 20 to 30 seconds, the final time will depend on the development of the game)

- **Punch_count:** This is will control how many times the player can punch In the game, currently its limited to 3

- **I:** This variable is directly linked to the 'I' button which is used for punching, its set to True at the beginning and but goes to False if the punch limit is reached and goes back to True if cooldwon is over

- **Kick_count:** This is will control how many times the player can kick In the game, currently its limited to 3

- **O:** This variable is directly linked to the 'O' button which is used for kicking, its set to True at the beginning and but goes to False if the kick limit is reached and goes back to True if cooldown is over

- **SP_count:** This is will control how many times the player can punch In the game, currently its limited to 3

- **P:** This variable is directly linked to the 'P' button which is used for super attacks, its set to True at the beginning and but goes to False if the super attack limit is reached and goes back to True if cooldown is over

- **Swap:** Thsi is used to allow the user to swap chracters when playing, it stays false so that the user doesn't swap forever and turns True when they want to and goes back to false until the cooldown is reached and the user wants to swap again.

- **Deg:** This is used to decide either the user wants to defedn or block/guard

- **Enemy_hp:** This stores the total health poinst of the enemy the player will face

- **Total_points:** This stores the amount of points the user has accumualted by playing through the levels

- **Points:** This is how many poinst the player can get by playing the level which will be added on top of **Total_points** if the win the level

- **Win:** This is a boolean which stays False until the user wins the level whicb then turns True and allows **Total_points** to be increased by 100 each time the user wins

- **Player_hp:** This is the total hp the user will have when enetering the fight which stays at 100

- **Remain_hp:** This is the health points the player will have when taking damage from the opponent, example: 100 -2 if they are kicked. This variable will be deciding how much hp **Player_hp** will have throughout the fight.

- **Enemy_hp:** This has the same pupose as **Player_hp** but its for the enemy and also the value may be increased to make it more difficult for the user

- **Eremain_hp:** This has the same purpose as **Remain_hp** but its for the enemy

- **Match:** This will store an array of moves that the player has to match in order to perform a combo attack.

Data structures

Combat: The comabt system will need a data structure since there will many moves that will need to be stored and and recalled whenever the user is playing the game since they will be attacking continously. Using a data structure will save time developing the game since an array of moves can be called in any game the user plays and it can be used for any of the charcters that will be present in the game to use the combat moves. This method of reusing code will make the game more efficient.

Music: The music that will be included In this game will need to be stroed into a list. This will help the user to acess the music from various areas of the game such as the options button in the main menu and pause menu while playing the game.

Maps: For the user to choose what map they want to play in, the maps will need to be stored in an array that will allow the player to access in the different modes that the game will have. Such as single player , training mode and other modes that could be implemented in the future.

Super attacks and skills: When the user accumalates enough points from playing single player mode, they will be able to purchase super attacks and skills. There will be several of these so they will need to be in a array for the user to buy form and it can also be easier to orgnaise them and call when making the game.

Validation

What the program will check for when the player uses the keyboard or mouse for inputs:

In game:

When the player enetrs W A S or D the character on screen should move accordingly, this espcially goes for the double dash that can be performed by pressing A or D twice.

If the user enters an attack button more than 3 times they will get a cooldown for that attack, this is done to stop them spamming and using the attacks carefully.

If the user has dropped to 0 hp for both the character they chose to fight with then the level will end and bring them back to the single player option.

If the player clicks the button to pause the game then the user should be displayed with the in gmae pasue menu which should also stop the game in the background.

If the game carries on more than the it was supposed to last for the timer will end the

If the users deal an attack to the opponent they should deal damage to them and the hp bar should decrease.

While the defence button is being held all dmg should be cancelled and thst state should only last for 2 to 3 seconds.

Id the users enter the specifc buttons for a combo then the additional animations for that combo should play and deal the damage that its supposed to. It should also start a cooldown for the nestx time iot can be used and be limited to how many times it can be used I the game.

In the menu and options

If the user clicks a button such as single player they should be displayed with the right screen such as character selection screen after clicking the sinlge player mode, this is shown in my game layout (page 22).

The point tracker should add the right amount of points each earned by the player after winning the levels and should not add any is they lost, this should be displayed correctly in the main menu.

Useabilty features

Navigation:

For the user to navigate through the game, for each display they go to after clicking the main 4 options(single player, trainging mode, customization, point tracker) there will be a back button to press to go to the previous screen. This way they can go through all the feautres however they want.

Consistency

For the player to have a consistent experience when playing the game all the equips they have bought should be saved to how they last left the game. This way it will be relevent for the user to not waste time re equiping the purchased items. Their points should also be saved from the last time they played the game unless they want to not save anythin.

User feedback

I will ask user feedback from my clients on the how their experince was with the gameplay and the menu useabilty. With their feedback I can aim to solve their issues and improve the game. This will then help me to get a better understabding of what type of addidtional feature they would like to see in the game.

Visual clarity

They font style and size will stay the same for all the text in the gamesince this way it will be easier to keep the game clear to read when needed, the style of the game will be linked to the retro style of the game. I will be using bright colours to make the game attracive and fun to play In, this will give the client the frienfly useer exeperience they want.

Effeciency

To make the game development efficient I will be asking If certain certain aspects of the game is important to the client. Such as is multiple chracters are neccessarry or are many levels needed to have playing the game. This will save time tand allow me to effeciently develop the game without taking any uneccessary turns while trying to match the standards of the client.

Flexibilty

I will try to make the useabilty of the game features flexible by allowing the user to easily access the options their given when displayed with the equips they can add to their characters. I will do so by giving them the choice of playing the animations of each skill and super attack as a demo that will

displayed if they use them in the game after purchase. Another flexibilty I will introduce is replaying the level they have already completed this way ot doesn't waste time to go back to the begiinning.

Error prevention

To prevent errors while playing the game I will make the inputs of the game simple tis way the user is able to perform all the moves without having problems registering on display. I will also make implement leeways for how long the player has to wait before they press the next button to perform a super attack shich is possible through combos.

Pseudocode

Menu pseudocode

```
Start menu
        Areas = [Single player,Training mode, Options,Customiztion]
        If user input =  single player:
                display single player to screen
        else If user input =  Training mode:
                display training mode to screen
        else If user input = options:
                display options to screen
        else If user input = customization:
                display customization to screen
```

This just an idea for how the menu will work. But for now I have created an abstract version of how the menu works.

This code will check for user input for what option they want to choose in the main menu. Then It will display the next screen that will be within that option.

Single player display pseudocode

```
Load screen

Chosen = 0
main menu = false
back = true
start = false
Characters = [char1,char2,char3,char4]
if back == true:
        main menu = true

If user input == char4:
        For x in range(len(Characters -1 )):
                If user input ==  char4:
                        Chosen = char4
                Else if user input == char3:
                        Chosen = char3
                Else if user input == char2:
                        Chosen = char2
                Else if user input == char1:
                        Chosen = char3
                Else if user input == char1:
                        Chosen = char3
```

This if statement will check if the user has clicked back and if so they will be taken to the main menu

This will check for what character the player has chosen, this will also work for if they decide to change their mind and select a different character.

```
If user input == start:
                start = true
                Load level 1
```

checks if user wants to start the game which will then load the 1st level

Moving pseudocosde

```
While player1 horizontal == “A”:
        Display A animation

While player1 horizontal == “D”:
        Display A animation

While player1 vert == “W”:
        Display W animation

While player1 vert == “S”:
        Display S animation
```

This is the input that the player has to use to move around In the game

Fighting pseudocosde

Punches:

```
Import time

#I don’t know hoe time works as of yet In c# but I will be using a timer to craet   cooldowns for the attacks

Cooldown = 0

Punch_count = 0

I = True

If player1 horizontal == “A” and player1 atk == I :
        Player1 = Left_punch
        Display Left_punch animation
        Dmg_given = -10
        Punch_count =+ 1

If player1 horizontal == “D” and player1 atk == I :
        Player1 = Right_punch
        Display Right_punch animation
        Dmg_given = -10
        Punch_count =+ 1

If player1 vert == “W” and player1 atk == I :
```

#This will check if the player has pressed an attack and will apply damage to the player and play the attack animation for punch. It will also add 1 to the counter for punch

```
        Player1 = Jump_punch
        Display Jump_punch animation
        Dmg_given = -10
        Punch_count =+ 1


If player1 vert == "S" and player1 atk == I :
        Player1 = Low_punch
        Display Low_punch animation
        Dmg_given = -10
        Punch_count =+ 1

While Punch_count == 0 and Cooldown == 0:
        If  Punch_count == 3 :
                I = False
                Cooldown =+2
        If Cooldown == 0:
                Punch_count = 0
                I = True
```

#This will check if the player has pressed the punch button 3 times, if so it will disable the button which is 'I' on the keyboard for 2 seconds. When thecooldown has finished the button will be enabled again.

Kicks:

```
Import time

Cooldown2 = 0

Kick_count = 0

O = True

If player1 horizontal == "A" and player1 atk == O :
        Player1 = Left_kick
        Display Left_kick animation
        Kick_count = +1
        Dmg_given = -10


If player1 horizontal == "D" and player1 atk == O :
        Player1 = Right_kick
        Display Right_kick animation
        Kick_count = +1
        Dmg_given = -10


If player1 vert == "W" and player1 atk == O :
```

#This will check if the player has pressed an attack and will apply damage to the player and play the attack animation for kicks. It will also add 1 to the counter for kicks

```
        Player1 = Jump_kick
        Display Jump_kick animation
        Kick_count = +1
        Dmg_given = -10

If player1 vert == "S" and player1 atk == O :
        Player1 = Low_kick
        Display Low_kick animation
        Kick_count = +1
        Dmg_given = -10


While Kick_count == 0 and Cooldown2 == 0:
        If  Kick_count == 3 :
                O = False
                Cooldown2 =+2
        If Cooldown2 == 0:
                Kick_count = 0
                O = True
```

#This will check if the player has pressed the kick button 3 times, if so it will disable the button which is 'O' on the keyboard for 2 seconds. When the cooldown has finished the button will be enabled again.

Super attacks

```
Import time

Cooldown3 = 0

SP_count = 0

P = True

If player1 horizontal == "A" and player1 atk == P :
        Player1 = Left_super
        Display Left_super animation
        Super_count = +1
        Dmg_given = -20

If player1 horizontal == "D" and player1 atk == P :
        Player1 = Right_super
        Display  Right_super animation
        Super_count = +1
        Dmg_given = -20

If player1 vert == "W" and player1 atk == P :
        Player1 = Jump_super
        Display Jump_super animation
        Super_count = +1
        Dmg_given = -20
```

#This will check if the player has pressed an attack and will apply damage to the player and play the attack animation for super attacks. It will also add 1 to the counter for supers.

```
While Cooldown3 == 0:
        If  SP_count == 1 :
                P = False
                Cooldown3 =+10
        If Cooldown3 == 0:
                SP_count = 0
                P = True
```

#This will check if the player has pressed the punch button 1 time, if so it will disable the button which is 'P' on the keyboard for 10 seconds. When the cooldown has finished the button will be enabled again.

Character swap

```
Import Time

Cooldown.Time = 0

def timer(Cooldown):

        Swap = False

        While Cooldown.Time ==0:

                If Cooldown.Time == 0 and Char1 == J :

                        Swap = True

                        Cooldown.Time =+ 20

                        Swap = False

                        Display Char2_swap animation

        While Cooldown.Time ==0:

                If Cooldown.Time == 0 and Char2 == J :

                        Swap = True

                        Cooldown.Time =+ 20

                        Swap = False

                        Display Char1_swap animation
```

This line checks if character 1 had pressed J, if they have they will swap out with character 2. After this there will be a 60 seconds cool down for the next time the characters can swap again. The duration might change in the development of the game

This line's purpose is the same as the one above.

Defending

```
If player1 horizontal == "A" and player1 Deg == K :
        Player1 = Right_def
        Display Right_def animation

If player1 horizontal == "D" and player1 Deg == K :
        Player1 = Left_def
        Display Left_def animation

If player1 horizontal == "A"  and player1 horizontal == "S" and player1 Deg == K :
        Player1 = RightLow_def
        Display RightLow_def animation
```

Deg stands for *De*fending and *g*uarding which I abbreviated to make it easier to keep track of.

```
If player1 horizontal == “D” and player1 horizontal == ”S” and player1 Deg == K :
        Player1 = LeftLow_def
        Display LeftLow_def animation
```

Guard break

```
If player1 horizontal == “A” and player1 Deg == L :
        Player1 =  Left_break
        Display Left_break animation

If player1 horizontal == “D” and player1 Deg == L :
        Player1 = Right_break
        Display Right_break animation
```

#This will check which direction the player breaks the guard of the other player disable it

Point tracker

```
Enemy_hp = 100

Total_points = 0

Points = 100

Win =  False

If Enemy_hp == 0:

        Win = True

        Total_points = + Points
```

#This will check if the enemy hp has dropped to 0 in a game and if it has the point tracker will increase by 100 points

Health system for player

```
Player_hp = 100

Remain_hp = 100

While Player_hp =< 100  :

        If player is kicked:

                Player_hp = Remain_hp -2

        Else if player is punched:

                Player_hp = Remain_hp -2

        Else if player is super attacked:

                Player_hp = Remain_hp -10

If Player_hp ==  0:

                End game

                Return to previous screen
```

#This will keep track of the damage the player is taking and will calculate it as long as hp is equal to 0 or below. If the player hp reaches 0 the game ends user will be taken to the previous screen such as single player mode or training mode

Health system for enemy

```
Enemy_hp = 100

Remain_hp = 100

While Enemy_hp is kicked:
        Enemy_hp = Remain_hp -2

While Enemy_hp is punched:
        Enemy_hp = Remian_hp -2

While Enemy_hp is super attacked:
        Enemy_hp = Remain_hp -10

If Enemy_hp = 0:
        End game
```

#This will keep track of the damage the enemy is taking and will calculate it as long as hp is equal to 0 or below. If the player hp reaches 0 the game ends user will be taken to the previous screen such as single player mode

Combo attack animation

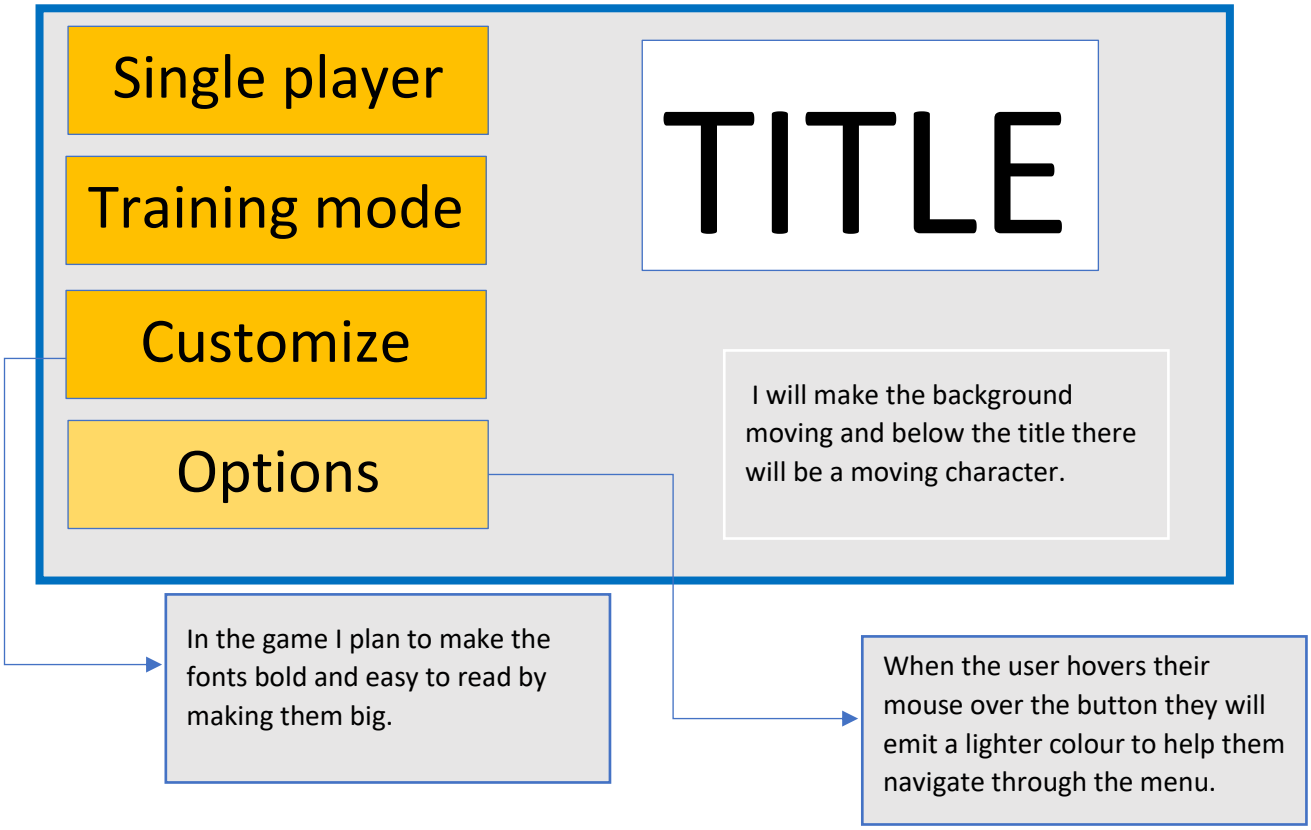
```
Match = [I,O,S ,O, D,I,O]

Def Match ( ):
        For x in len(range(Match):
                If user input == Match:
                        Dmg_given = 7
                        Display_Comboatk
```

#This is just an idea of how the combo system will work in th game but it will be much more complex then this in c#. Match will just be the required inputs the player has to press in the that order to play an additional “Comboatk” to deal extra damage.

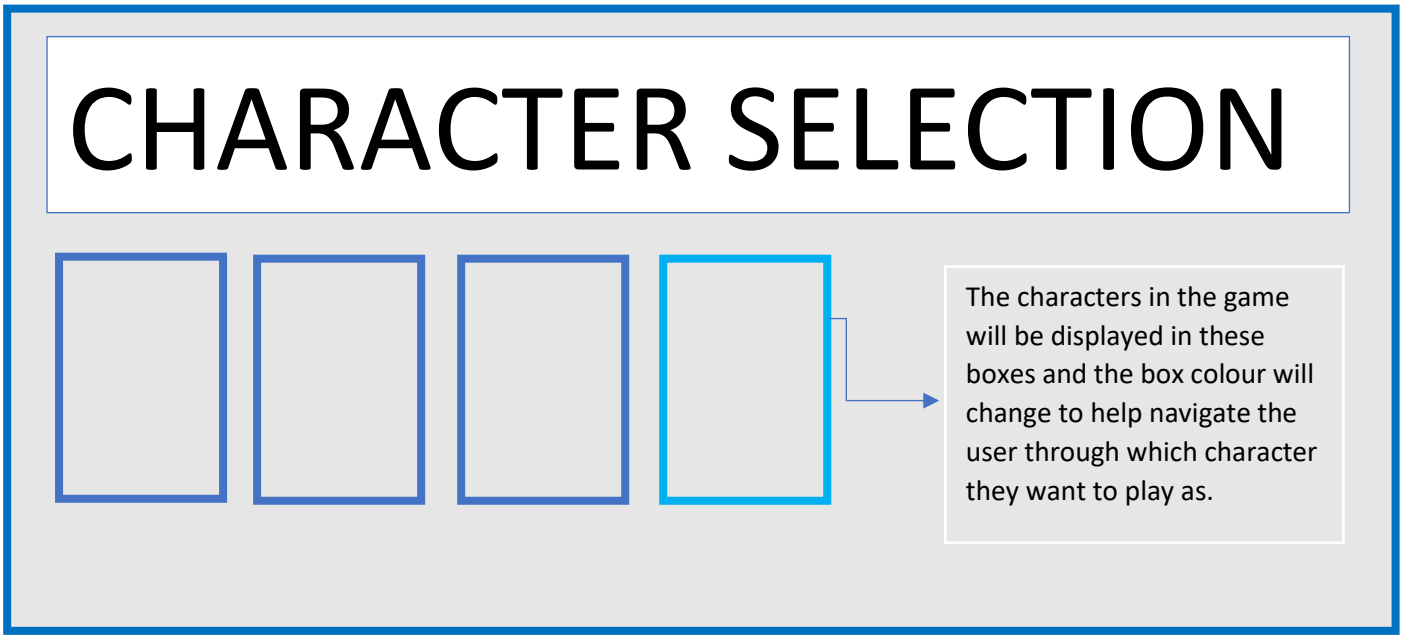
Game design

Menu screen: **The menu that will load up when the game is opened by the user.**

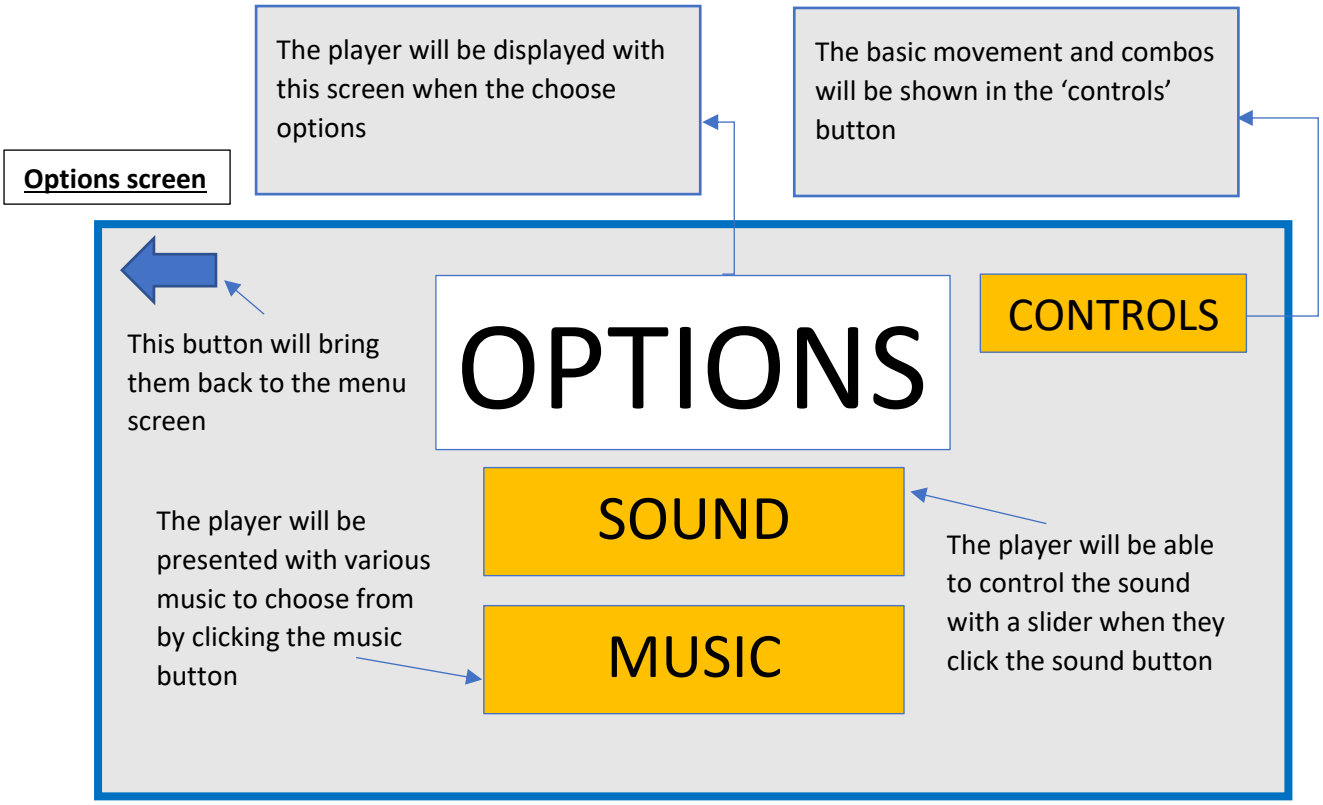
The colours and fonts presented in this design is just an example of how the game will be displayed in the game, its not the final design.



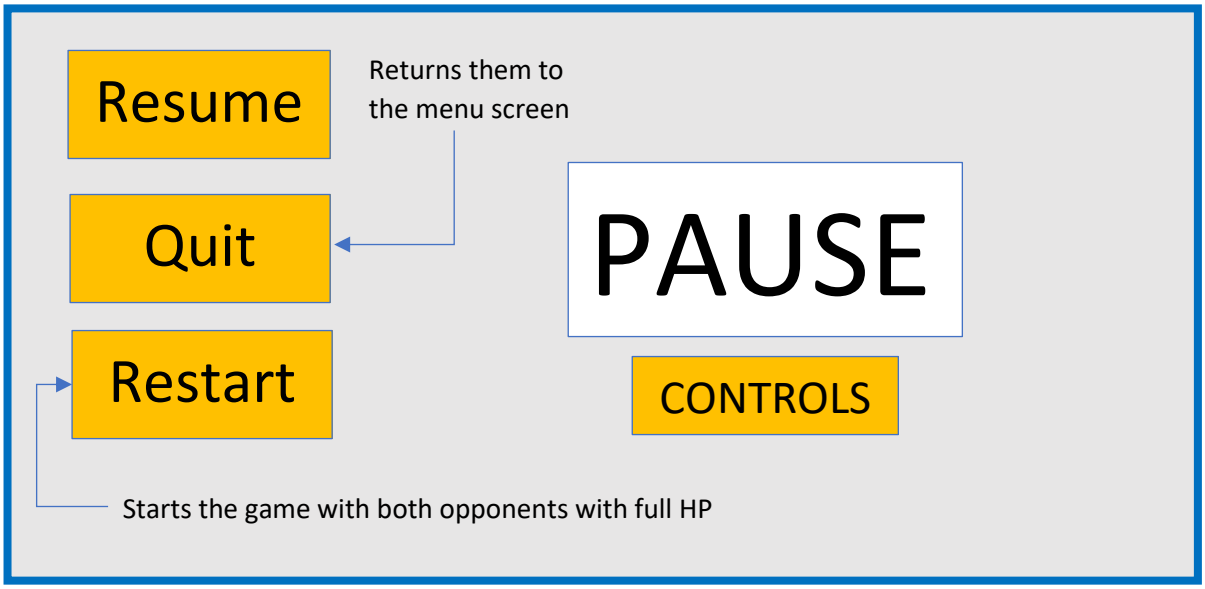
Single player/training mode: this is what will be displayed when the user presses either button but with slight differences.

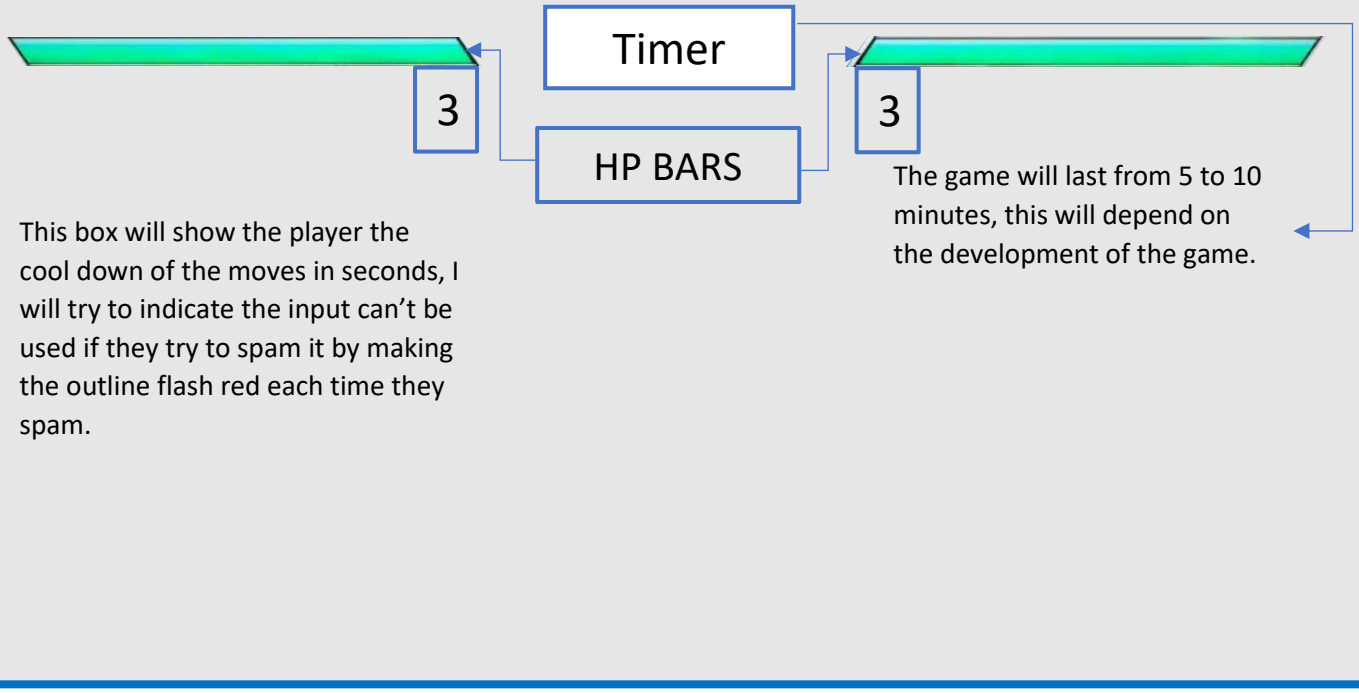


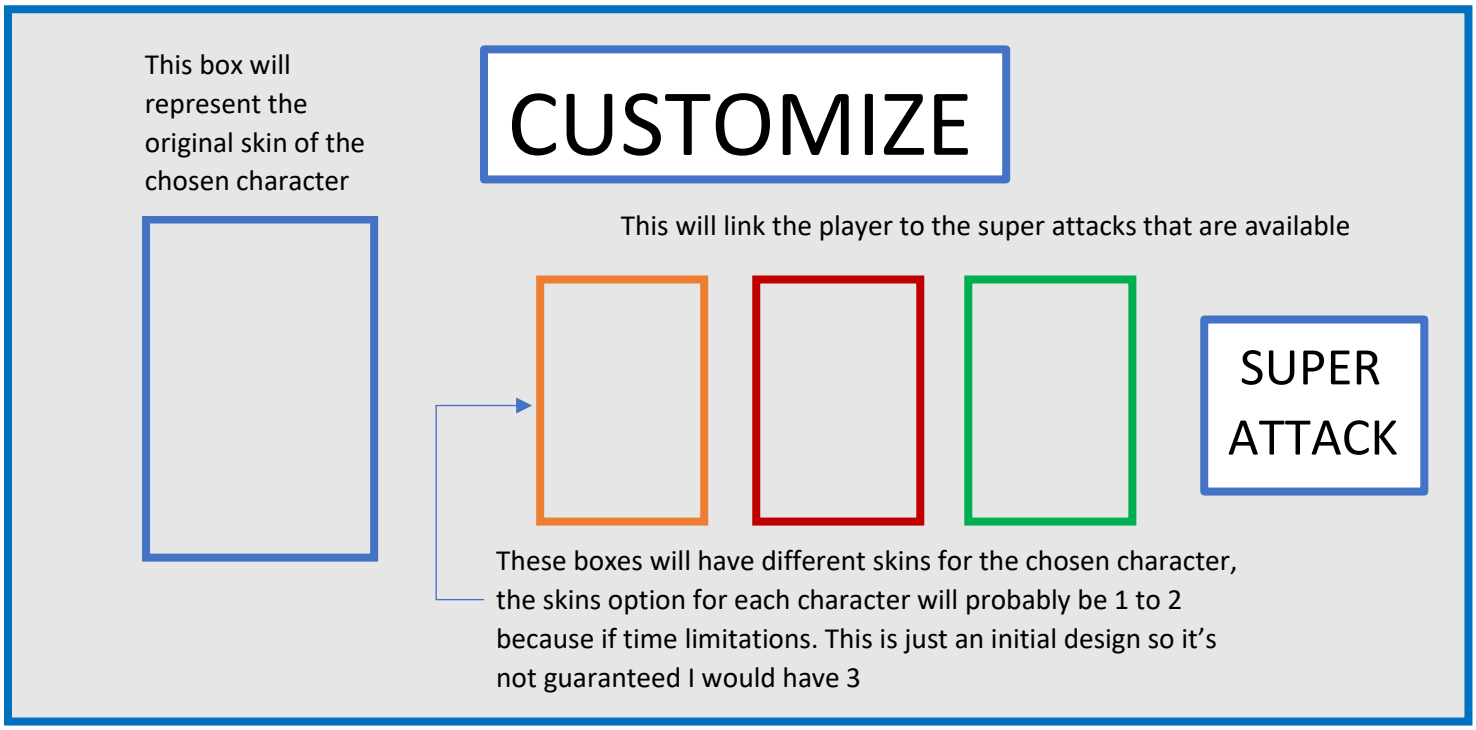
Options screen



IN-GAME MENU




Timer
3
HP BARS
3
This box will show the player the cool down of the moves in seconds, I will try to indicate the input can't be used if they try to spam it by making the outline flash red each time they spam.
The game will last from 5 to 10 minutes, this will depend on the development of the game.


This box will represent the original skin of the chosen character
CUSTOMIZE
This will link the player to the super attacks that are available
SUPER ATTACK
These boxes will have different skins for the chosen character, the skins option for each character will probably be 1 to 2 because if time limitations. This is just an initial design so it's not guaranteed I would have 3

Iterative development test data

Test No.	What testing	How testing	Test data	Good/Bad/Boundary	Expected result	Actual result	Actions needed
1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has		
2	Fight system/ box colliders	Use inputs assigned to each character move and check if box colliders are being activated	I,O,P,J,K,L	Good: “I” Bad: “v” Boundary: n/a	See if the attacks are activating /colliding with other character’s box colliders		
3	Animations	Test if animations play if conditions are met	I,O,P	Good: “I” Bad: “G” Boundary: N/A	See if the animations for specific conditions are displayed		
4	Movement	Use the horizontal and vertical inputs to see if character moves	W,S,A,D	Good: “W” Bad: “B” Boundary: N/A	The user will be moving in response to each of the user inputs so will the enemy		
5	Spam counter code	See if attacks are disabled after using them after a certain number of times	I,O,P,	Good: “I” Bad: “C” Boundary: pressing “I” 3 times	The attack or action will be disabled for a specific amount of time		
6	Character selection	Check if the character chosen is loaded into the fight and their move set also exist when using them	Using mouse and buttons	Good: use mouse to click character box Bad: click outside the box Boundary: the edge of the box	The character should be loaded into the game with their assigned attacks and abilities		

7	Customization	Check if skins, super attacks and skills are able to be used after unlocking them and using them in the game	Use mouse to select equips or use buttons	Good: use mouse to click what to purchase Bad: using mouse to click outside the boxes to purchase them Boundary: edge of box	The user should have access to the equips purchased in customization		
8	Point tracker	Check if the points accumulated from playing is added to the point tracker and allows the user to purchase from the customization area	Beat level to see if point tracker value goes up	Good: outputs 100 after win Bad: outputs 0 after win Boundary: N/A	The value should increases when the levels from single player mode is won		
9	Maps	The maps options will allow the user to select what background they want to play in	Use mouse and buttons	**Good:** Wanted map is displayed in game **Bad:** map not appearing in game **Boundary:** N/A	User should be able to click on what map they want to play in and be transitioned to the game in the map they selected		
10	Swap characters	Check if the user is able to swap characters and the	Input J	Good: character is swapped Bad: not swapped Boundary: not swapped If cool down is active	The character1 should swap with character2		
11	Defend	Use input to defend against enemy	Input K	Good: if the damage dealt is cancelled Bad: damage not cancelled Boundary: damage dealt after defence is disabled	The user should be able to defend and negate the damage that is being dealt to them		

12	Guard break	Use input to break enemy guard/ defence	Input L	Good: the other character guard is broken Bad: not broken Boundary: N/A	The guard of the opponent should break and let the attacker continue to attack the opponent		
13	Health system	Check if the user and enemy health is decreased after taking damage	Use attacks to see if enemy is taking damage and check user is taking damage by letting enemy attack	Good: player1 health decreases after attack Bad: no damage dealt Boundary: health staying the same after being attacked in defending state	The health of both the opponents should decrease by the amount the attacker is applying to them. Example kick should do 5 damage and super should do 10		

Test strategies: Post development

White box

I will be using white box testing on the game after developing the first prototype and while developing the code since I can test how the player moves and attacks while making the game since unity allows me to do so. This can make it easier for expecting how the game will feel like after the basic functionalities have been implemented into the game. This will also help fix errors that might appear in the game after development and also make adjustments to the client needs.

Black box

To use the iterative development testing I will use the method black box testing. This will help me track of what is working in the game and what I need to fix, adjust and add. This way I can also try to test some of the initial ideas from my analysis and see if I was able to make what I wanted in the beginning. In this testing since there will be no needed information on the internal structure and code of the game I will also look at the game from a user perspective and add what my clients would want more in the game.

Post development testing

Most of my post development testing will be user input and display output. I will also be using code testing that may be needed for new features client want me to add or adjust.

1) See if the attacks register and are displayed right after the button has been pressed
2) Test if the animations for each attack is running smoothly without lag
3) Check if player and enemy are taking damage from each other form attacks
4) Press pause button while playing to see if the game stops and displays the option to quit game
5) Check to see if the point tracker is being updated after winning and losing, this way I can see the point system is working properly
6) Use the mouse to see if the buttons in the menu and are working the display is transitioning between the options in the layout that has been created (page 22)
7) Using the mouse to select different songs or music to see if they change when set to default
8) Check if the map that was selected before match starts is the background when fighting
9) Using the mouse to purchase and skins and skills to use in game
10) Check if the point tracker decreases after purchasing skins and skills
11) Check if data is saved after closing game

These were some of the main test ideas that came to mind since this is part of my success criteria that I have to meet and also these are essential for the game to work properly and to make improvements on form user feedback

Development

Reference:

Broly sprites:

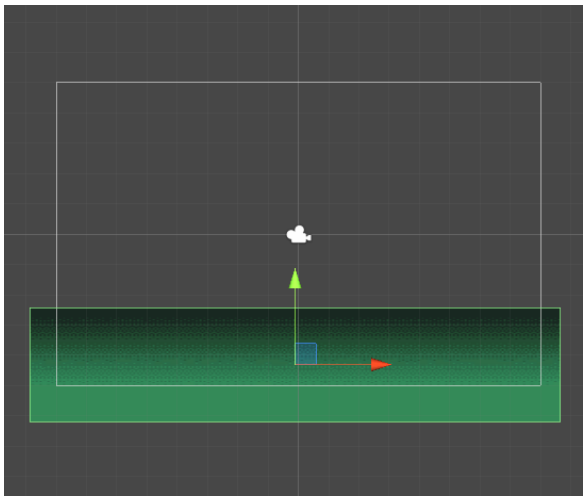
https://www.deviantart.com/supernico92/art/LSW-LSSJ-Broly-Sprite-sheet-v-2009-267412338

Goku sprites:

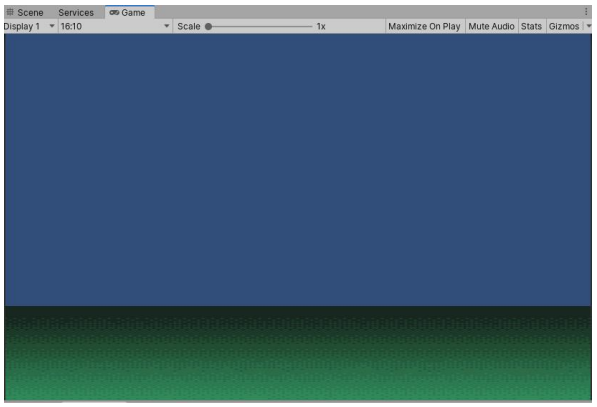
https://www.deviantart.com/thekrillmaster/art/Goku-SSJB-Go-gi-sprite-sheet-708558297

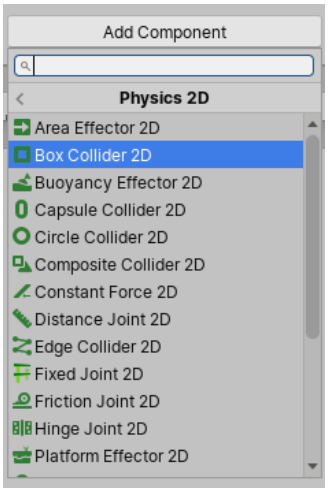
I used a software that takes sprite sheets and cuts them out into individual sprites. It also allowed me select whichever I want and import them into my unity game file as assets.

Foreground and background

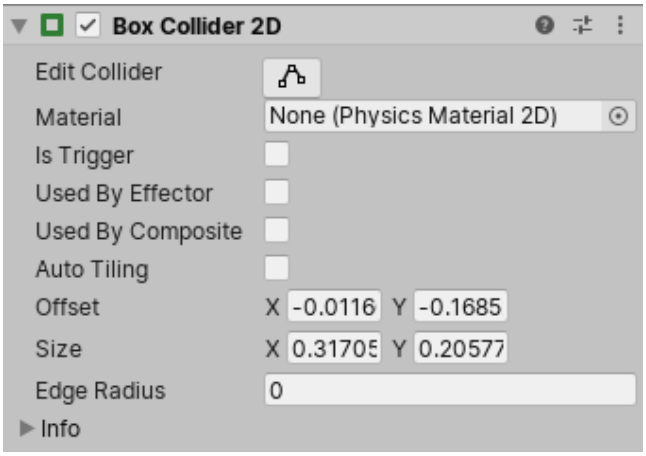


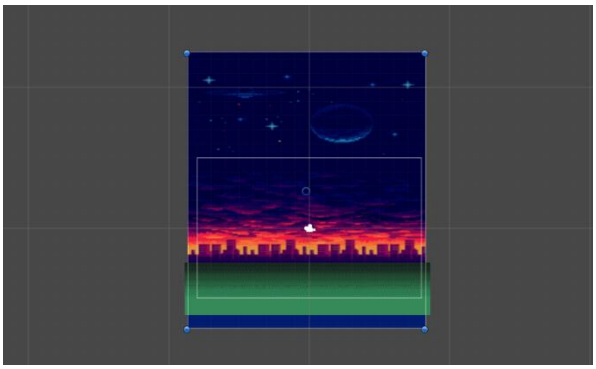
For the foreground I stretched the game object to the outside of the camera to make sure that when the game is played there isn't any gaps. In the game window you can see that the surface is fully covered.

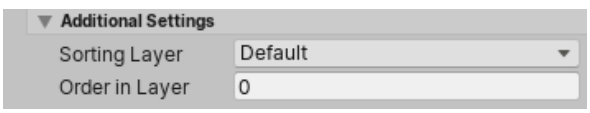


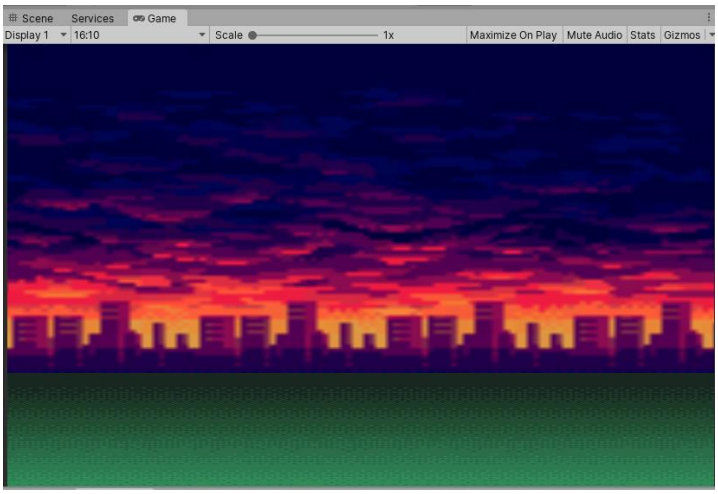


To add physics to the ground I used add component to create a box collider around the object to so that the character sprites don't fall through the ground.



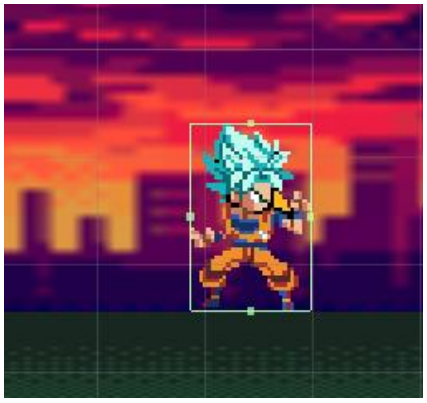


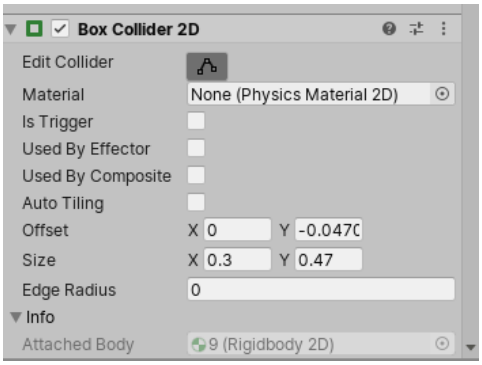


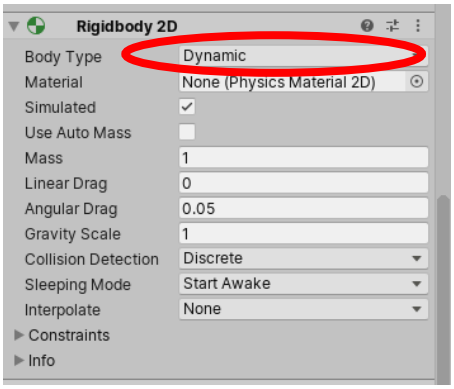


After I just dragged in the background I wanted to use into the camera and stretched it to fit the camera. I made the order in layer to 0 since I want all the other sprites in the future to all be in front of the background.

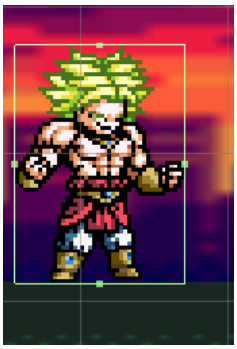
Importing characters into the scene







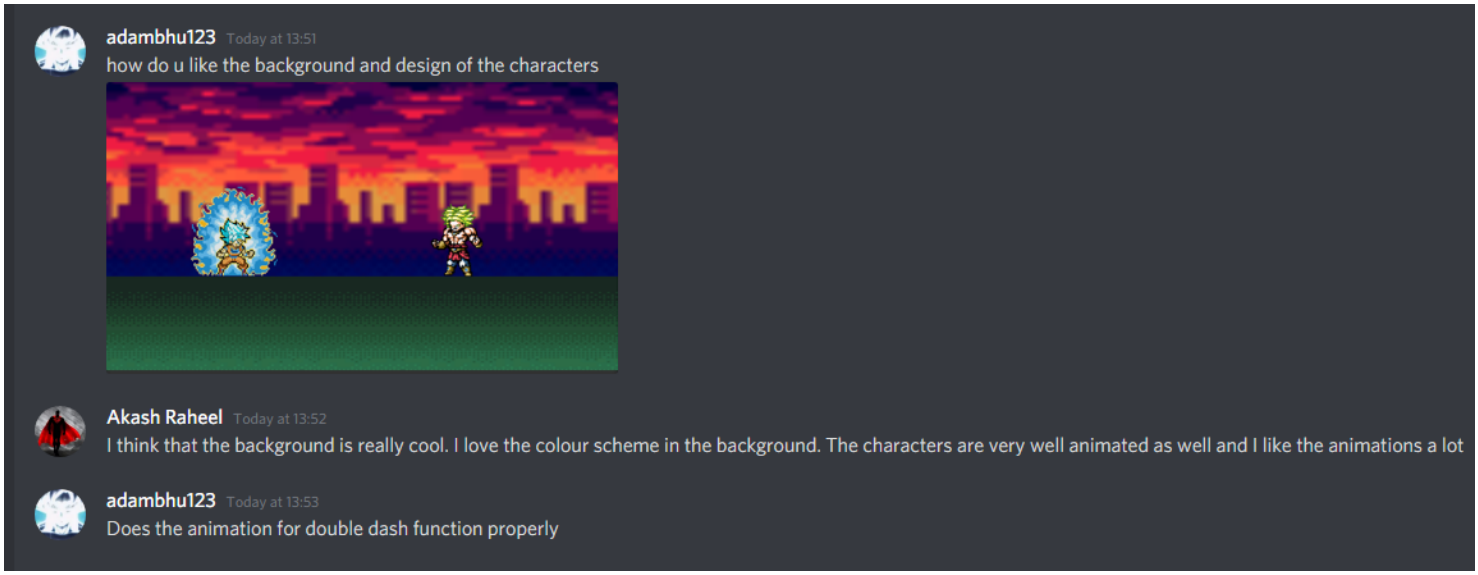
The character has a box collider 2d that will act as a box whenever it comes into contact with another object in this case opponents. I also added a rigid body 2D so that he can move when I attach a movement script to him.



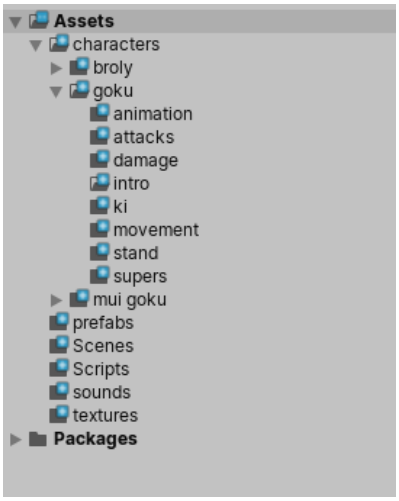
I did the same thing to this sprite by adding a box collider and a rigidbody 2D. This character will be used to test in the early stages to check if Goku can apply damage to him, if Broly will play animations if he is hurt, and if he will cancel damage if being attacked by Goku. While I make the essential scripts for player such as moving and attacking, I will also make a second version to attach to player2.

Currently this is how I have set my game and the sprites I have used are from itch.io. These will be the characters I will be using to test the code on and try to make the combat system with. I got these for free and the comments have people asking if they can use them to makes games ,the creator has said yes but to mention their names in the credentials which I will.

User feedback



The user likes the current setup of the game which will be good use as a reference when developing further maps and character that will have a similar to the one above



These are the assets I will be using to animate the attacks the characters will be performing in the game. They also include the scripts to make the characters move as the user wants to and the animations are there is well that I will create. As I document my development I will show all the sprites for each animation.

```
using UnityEngine;
using UnityEngine.Events;

2 references
public class CharacterController2D : MonoBehaviour
{
    [SerializeField] private float m_JumpForce = 400f;                          // Amount of force added when the player jumps.
    [Range(0, 1)] [SerializeField] private float m_CrouchSpeed = .36f;          // Amount of maxSpeed applied to crouching movement. 1 = 100%
    [Range(0, .3f)] [SerializeField] private float m_MovementSmoothing = .05f;  // How much to smooth out the movement
    [SerializeField] private bool m_AirControl = false;                         // Whether or not a player can steer while jumping;
    [SerializeField] private LayerMask m_WhatIsGround;                          // A mask determining what is ground to the character
    [SerializeField] private Transform m_GroundCheck;                           // A position marking where to check if the player is grounded.
    [SerializeField] private Transform m_CeilingCheck;                          // A position marking where to check for ceilings
    [SerializeField] private Collider2D m_CrouchDisableCollider;                // A collider that will be disabled when crouching


    const float k_GroundedRadius = .2f; // Radius of the overlap circle to determine if grounded
    private bool m_Grounded;            // Whether or not the player is grounded.
    const float k_CeilingRadius = .2f; // Radius of the overlap circle to determine if the player can stand up
    private Rigidbody2D m_Rigidbody2D;
    private bool m_FacingRight = true;  // For determining which way the player is currently facing.
    private Vector3 m_Velocity = Vector3.zero;

    [Header("Events")]
    [Space]

    public UnityEvent OnLandEvent;

    [System.Serializable]
    2 references
    public class BoolEvent : UnityEvent<bool> { }

    public BoolEvent OnCrouchEvent;
    private bool m_wasCrouching = false;

    0 references
    private void Awake()
    {
        m_Rigidbody2D = GetComponent<Rigidbody2D>();

        if (OnLandEvent == null)
            OnLandEvent = new UnityEvent();

        if (OnCrouchEvent == null)
            OnCrouchEvent = new BoolEvent();
    }
```

```
0 references
private void FixedUpdate()
{
    bool wasGrounded = m_Grounded;
    m_Grounded = false;

    // The player is grounded if a circlecast to the groundcheck position hits anything designated as ground
    // This can be done using layers instead but Sample Assets will not overwrite your project settings.
    Collider2D[] colliders = Physics2D.OverlapCircleAll(m_GroundCheck.position, k_GroundedRadius, m_WhatIsGround);
    for (int i = 0; i < colliders.Length; i++)
    {
        if (colliders[i].gameObject != gameObject)
        {
            m_Grounded = true;
            if (!wasGrounded)
                OnLandEvent.Invoke();
        }
    }
}
```

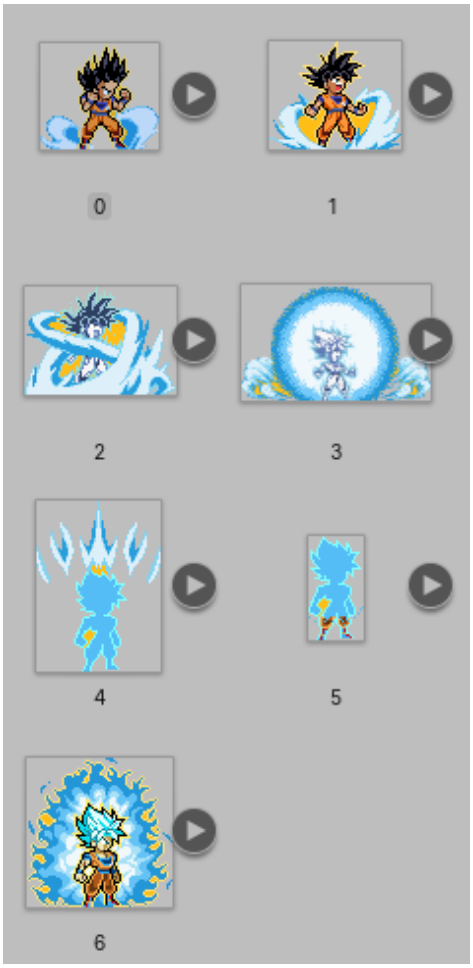
```
2 references
public void Move(float move, bool crouch, bool jump)
{
    // If crouching, check to see if the character can stand up
    if (!crouch)
    {
        // If the character has a ceiling preventing them from standing up, keep them crouching
        if (Physics2D.OverlapCircle(m_CeilingCheck.position, k_CeilingRadius, m_WhatIsGround))
        {
            crouch = true;
        }
    }

    //only control the player if grounded or airControl is turned on
    if (m_Grounded || m_AirControl)
    {

        // If crouching
        if (crouch)
        {
            if (!m_wasCrouching)
            {
                m_wasCrouching = true;
                OnCrouchEvent.Invoke(true);
            }

            // Reduce the speed by the crouchSpeed multiplier
            move *= m_CrouchSpeed;

            // Disable one of the colliders when crouching
            if (m_CrouchDisableCollider != null)
                m_CrouchDisableCollider.enabled = false;
```

```
        else
        {
            // Enable the collider when not crouching
            if (m_CrouchDisableCollider != null)
                m_CrouchDisableCollider.enabled = true;

            if (m_wasCrouching)
            {
                m_wasCrouching = false;
                OnCrouchEvent.Invoke(false);
            }
        }

        // Move the character by finding the target velocity
        Vector3 targetVelocity = new Vector2(move * 10f, m_Rigidbody2D.velocity.y);
        // And then smoothing it out and applying it to the character
        m_Rigidbody2D.velocity = Vector3.SmoothDamp(m_Rigidbody2D.velocity, targetVelocity, ref m_Velocity, m_MovementSmoothing);

        // If the input is moving the player right and the player is facing left...
        if (move > 0 && !m_FacingRight)
        {
            // ... flip the player.
            Flip();
        }
        // Otherwise if the input is moving the player left and the player is facing right...
        else if (move < 0 && m_FacingRight)
        {
            // ... flip the player.
            Flip();
        }
    }
    // If the player should jump...
    if (m_Grounded && jump)
    {
        // Add a vertical force to the player.
        m_Grounded = false;
        m_Rigidbody2D.AddForce(new Vector2(0f, m_JumpForce));
    }
}


2 references
private void Flip()
{
    // Switch the way the player is labelled as facing.
    m_FacingRight = !m_FacingRight;

    transform.Rotate(0f, 180f, 0f);
}
```

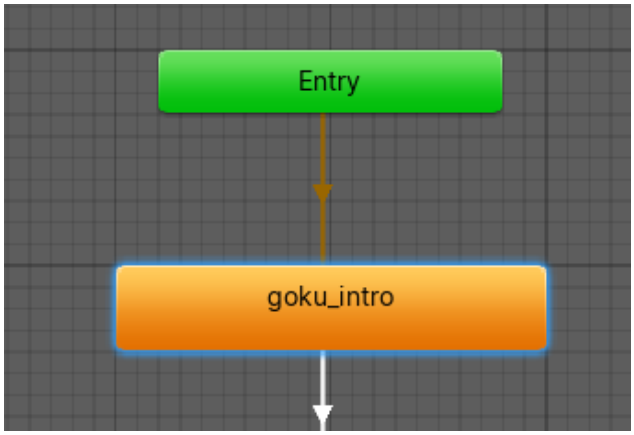
The code above is from a youtuber called Brackeys, I used his code for the physics that will be involved in the game for the characters when they move and horizontally and vertically. If I tried to make this myself, it would make the character feel stiff to move around in the game and it won't be natural for the user to play like that in the game. The comments made in the code are done by him so I will also be learning from this code and will try to apply it to other parts of the game myself.

https://github.com/Brackeys/2D-Character-Controller/blob/master/CharacterController2D.cs

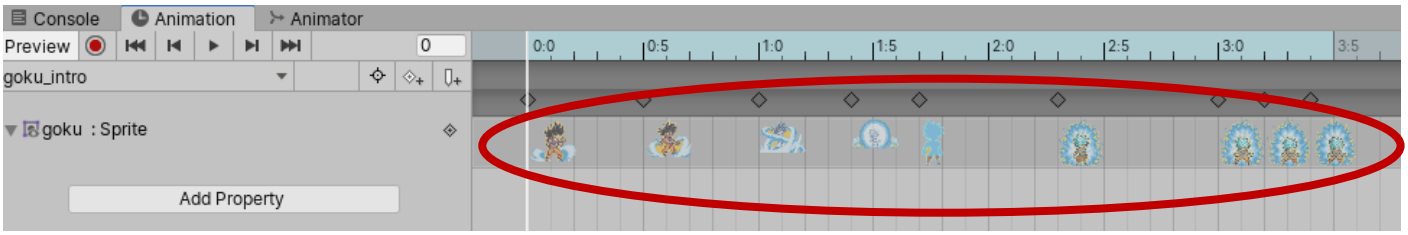
Intro animation for Goku



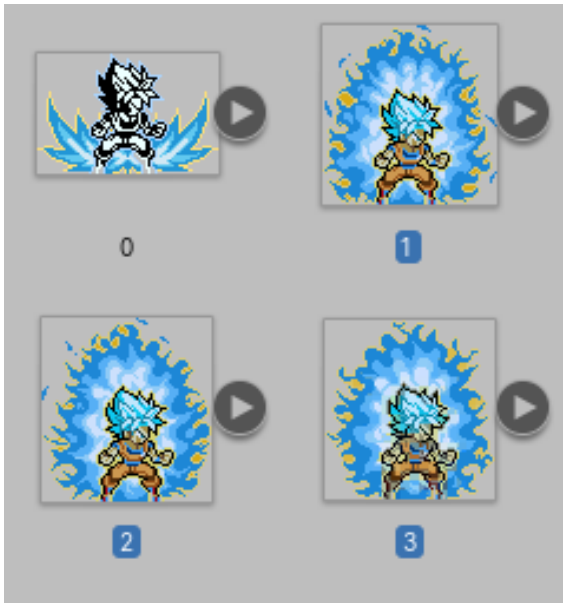
These sprites are used for the intro scene when the game starts for the character Goku.



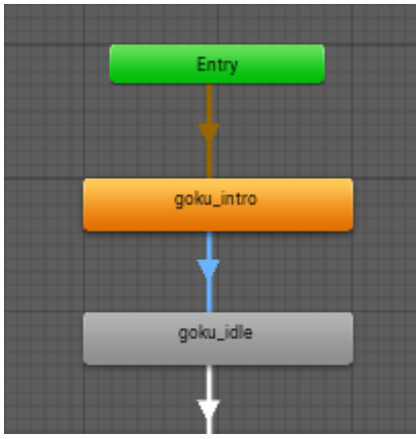
I have set the intro animation as the entry when the game starts in the animator.

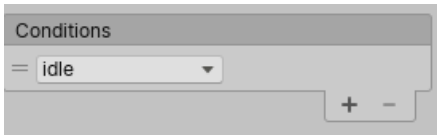


I used the animation tab to enter the sprites in and space the into the right key frames so that animation plays naturally.

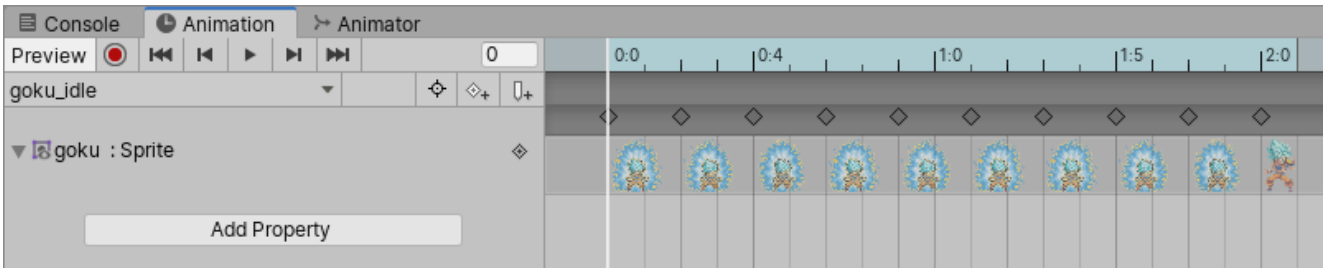


The sprites with number 1,2,3 are used when the intro animation stops then goes into the idle position which I will show after.



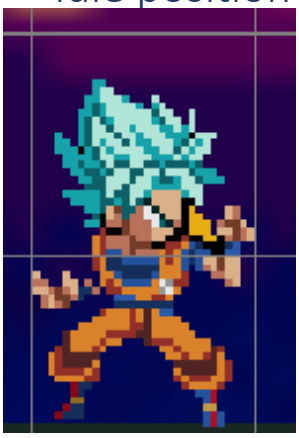


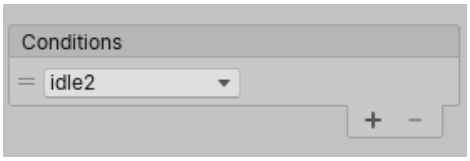
After goku_intro comes goku_idle which is triggered by the idle parameter which I added in as trigger that plays after the intro stops.

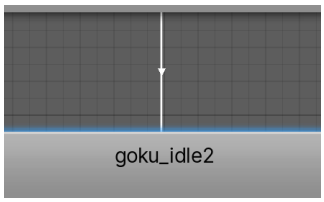


I have reused the same sprites again to have a loop effect when the animation plays. Then the last key frame is how the character will be when they are not moving in the game.

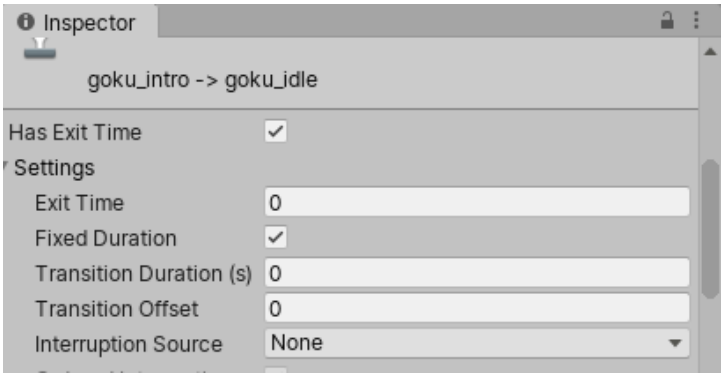
Idle position and movement animation



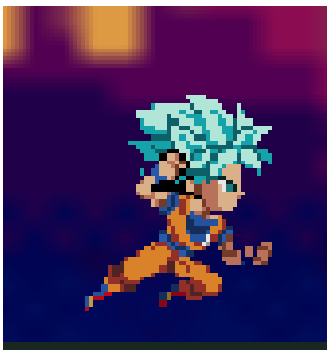


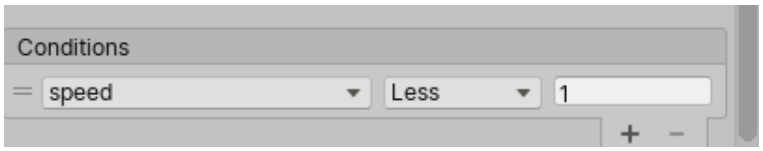


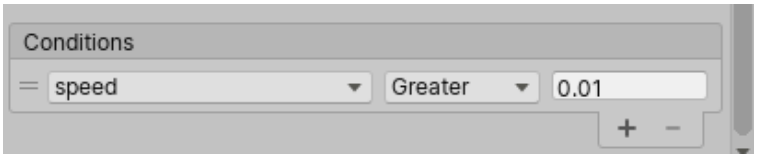
After the goku_idle plays then it transitions to goku_idle 2 by using the condition **idle2** which triggers after the previous state stops playing. This is will be the state of the player when he is idle and all the animations that are played by him such as the attacks will go back to this state when he stops performing them.

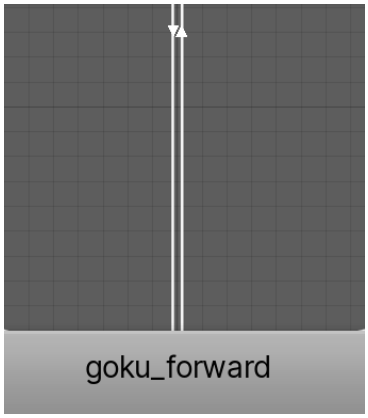


The transition between goku_idle and goku_idle2 have the same exit time and transition time which is 0 because I want them to go to the next state instantly since this will make the gameplay more responsive.





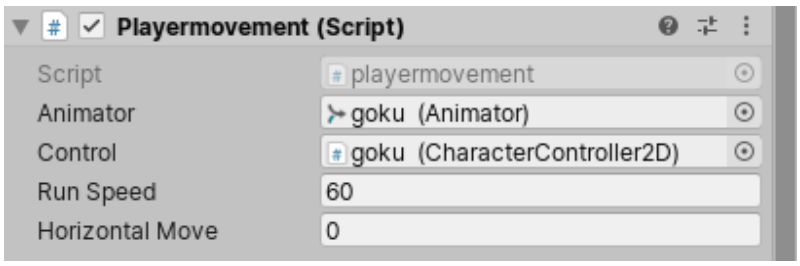




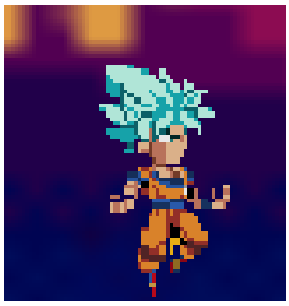
Goku_forward will be the state that will be used to play the moving animation when the player moves their character left and right on the ground or air. I have used a float parameter as my condition to decide when the player movement animation should play . It works whenever the player moves horizontally and if so the code below will make the speed equal to the horizontalMove variable which changes depending on whether the player is moving. When moving the speed is 40 and that's more than 0.1, so it plays the goku_forward animation when moving. When the speed is below 1 the horizontal Move variable goes back to 0 when the player isn't moving this causes the player transitioning back to goku_idle2.

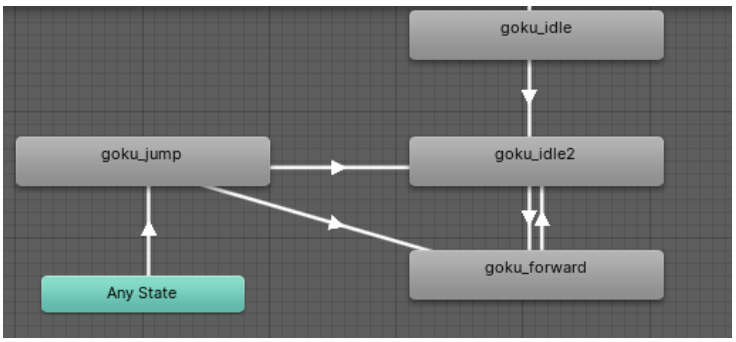
```
public Animator animator; // this references the animator which controls all the animations that the character does in the game
public CharacterController2D control;// this is the script i will be referncing to make the charcter move smoothly
public float runSpeed = 40f;
public float horizontalMove = 0f;
// the variables above will control movement speed
bool up = false;
0 references
void Update()
{
    {
        horizontalMove = Input.GetAxisRaw("Horizontal") * runSpeed;
        //GetAxisRaw determines if the player is going left or right by the value -1 moving left and 1 when moving right then runSpeed multplies the value by 40 to give the character speed
        animator.SetFloat("speed", Mathf.Abs(horizontalMove));
        // this is used to get the input from the keyboard for horizantal movement and runSpeed will control how fast they can move
        // the animator refernced will play the animation for moving left anf right
        //Mathf.Abs is used to keep the speed value always poistive even when it is going left which is -1
    }

    if (Input.GetButton("UP"))
    {
        up = true;
        animator.SetBool("jump", true);
        // this is to figure out if the player is jumping or not, if os the animtion for jump will be played
    }

}

0 references
public void Landing(){
    animator.SetBool("jump", false);
    // if they are not jumping the animation wont play
}

0 references
void FixedUpdate(){
    // moves character
    control.Move (horizontalMove * Time.fixedDeltaTime, false, up);
    up = false;
}
```

This code is used for the input that the user will be using to move the character. This will enable them to move horizontally and jump up and down, I will also explain how I made the jump animation work below.



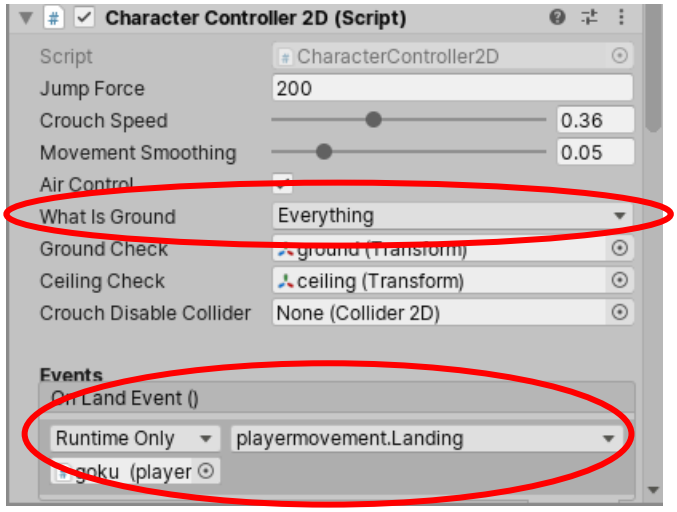
I can control and adjust the run speed to what I want since I made it a public variable, and I also referenced both animator and control to allow the script to enter the animations and play them for player movement, and the game object the script is working for.





This is the sprite for when the player jumps. The code for controlling the input for jumping is above in the same script for moving the character. I used “Any State” for transitioning to jump because when playing the game, the player should be allowed to jump when they want and shouldn’t wait for other state transitions unless they are taking damage or already in the air. But I might include double jump in the future. The animation plays when jump is true, and because I used Any state it saves times making transition between the other states, unless jump is false.

The two images on the left are the reference where the ceiling and the ground is for the character, this is then used in the charactercontroller2D script to detect when ever the player comes into contact with any object on top or bottom of them. It also allows the player to jump into the air by detecting what is ground and can only jump if there is ground. Since I used this code form Brackey I wont be able to explain everything in the code since it isn't in my level of knowledge but I will try to explain the parts I understand. In the 1st area I highlighted red that is where it checks what is ground, I set everything because there Is no other place to land other then the foreground sprite. The 2nd area is what keep the character from jumping forever, it does so by checking if the player is grounded and then stops jumping.




```
    // If the player should jump...
    if (m_Grounded && jump)
    {
        // Add a vertical force to the player.
        m_Grounded = false;
        m_Rigidbody2D.AddForce(new Vector2(0f, m_JumpForce));
    }
}
```

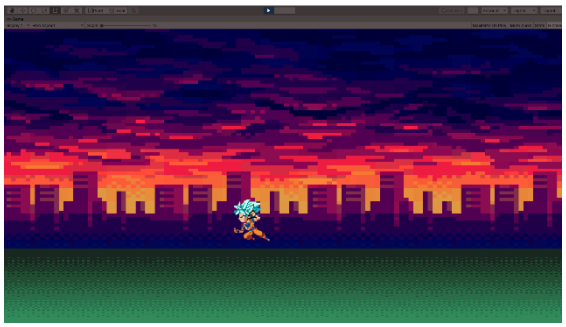
The code that I got from Brackey checks if the player is grounded and jump is true of false. If it's true then a "vertical force" is added. This lets the player jump.

```
0 references
private void FixedUpdate()
{
    bool wasGrounded = m_Grounded;
    m_Grounded = false;

    // The player is grounded if a circlecast to the groundcheck position hits anything designated as ground
    // This can be done using layers instead but Sample Assets will not overwrite your project settings.
    Collider2D[] colliders = Physics2D.OverlapCircleAll(m_GroundCheck.position, k_GroundedRadius, m_WhatIsGround);
    for (int i = 0; i < colliders.Length; i++)
    {
        if (colliders[i].gameObject != gameObject)
        {
            m_Grounded = true;
            if (!wasGrounded)
                OnLandEvent.Invoke();
        }
```

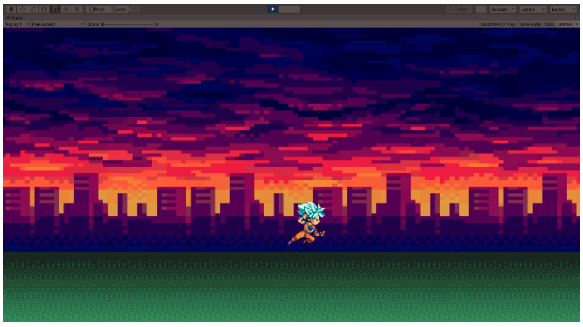
As you can see in the code above you can see that the ground and ceiling position I referenced early is used in Brackey's code to determine that the player is grounded.

Test for movement

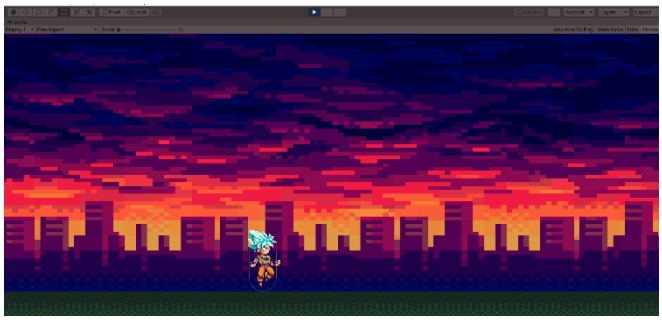
1	Movement	Use the horizontal and vertical inputs to see if character moves	W,S,A,D	Good: “W” Bad: “B” Boundary: N/A	The user will be moving in response to each of the user inputs so will the enemy	The character is moving left and right when keys A and D are used	however, the character doesn’t stop jumping animation



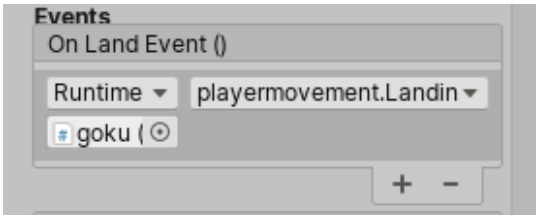
Left flying animation



Right flying animation



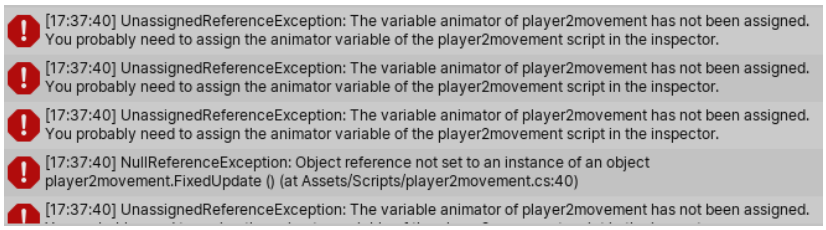
Character doesn’t stop jump



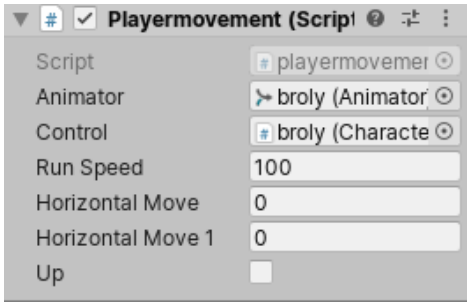
I forgot to reference the on land event function to the character controller 2d script, this should fix the issue of the character continuously playing the jump animation.

2	Movement	Use the horizontal and vertical inputs to see if character moves	W,S,A,D	Good: “W” Bad: “B” Boundary: N/A	The user will be moving in response to each of the user inputs so will the enemy	The character is moving left and right when keys A and D are used. The character now jumps once when jump button is entered

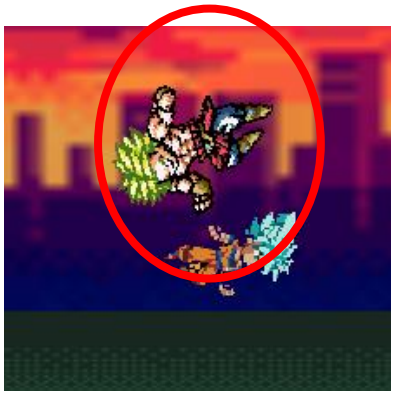
Error

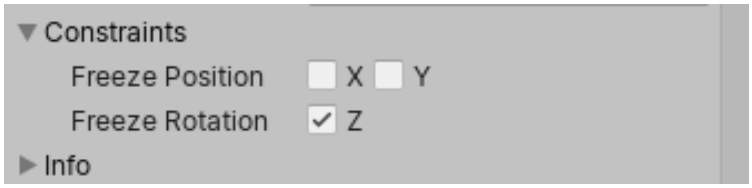


This problem appeared when I tried to make a character controller for player 2. I resolved this issue by dragging the reference which was the 2nd character into the slot in the code that is in the inspector. This simple issue was hard for me to solve just because I didn't fully understand the error and simply forgot to reference the animator. This is because I thought I just had to attach the code to the character. This has taught me to check the references that I should include and check the inspector if I'm missing any in the future.



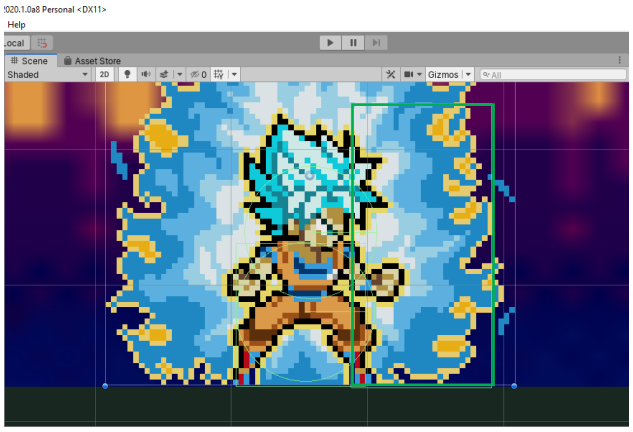
Character interaction issue



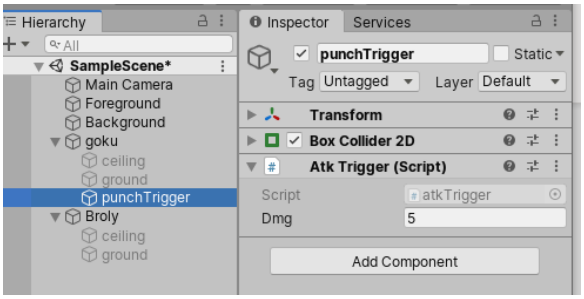


This screenshot shows the result of after one of the characters' trying to jump over the other. This has led them to tilt forward and unable to get back the upright position. This is because the box colliders of the sprites form corners near their heads. This causes each other sprites to tilt forward since its two rectangles colliding. To solve this issue, I froze the 'z' axis of each of the sprites so that even if their box colliders interact ,when they try to jump over each other it doesn't make them tilt forward.

Dealing damage

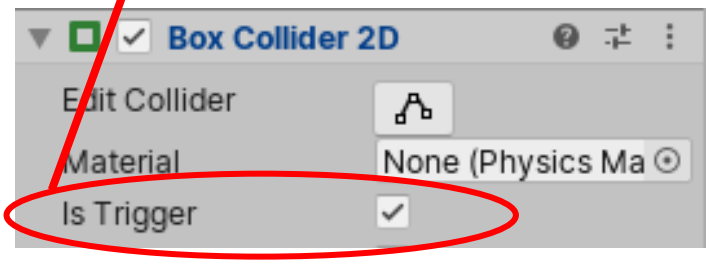


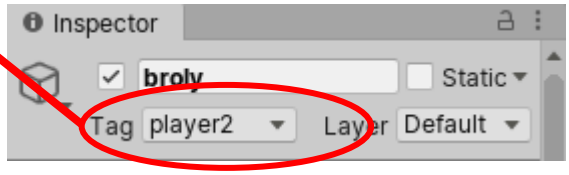
The green rectangle in front of the character is trigger check for when the user inputs an attack button. When an attack is clicked the box will check if an object is within the box if it is they will take damage by comparing the object tag and the tag that is in the code that is attached to the box collider. In this case it checks for the tag player2.



This is the script that checks for player2 and inflicts damage if the player is in contact with the attacks. The last line of the script below will send the dmg int variable equal to 5 to the "Damage" function which is in player2's script called dmg_taken(see page 53). This will then subtract the hp by 5.

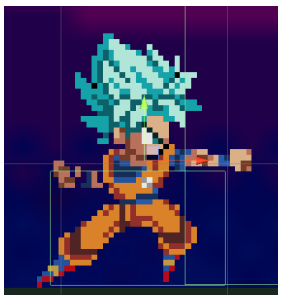
```
0 references
public class atkTrigger : MonoBehaviour
{
    public int dmg = 5;
    public int dmg_cancel = 0;

    0 references
    void OnTriggerEnter2D(Collider2D col)// this is a sunction that checks any 2d colliders that it come in contact with
    {
        if (col.isTrigger != true && col.CompareTag("player2"))
            //if the box collider this script is attached to is triggered through punch or kick and the contact object has a tag of player2
        {
            col.SendMessageUpwards("Damage", dmg);
```

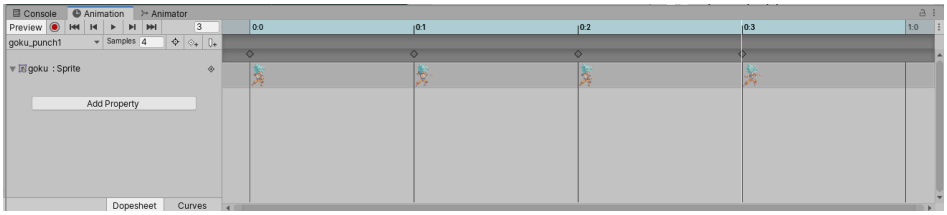





```
player1_trigger.enabled = true;
// the trigger then also becomes true
```

This code is from page 56, when the player enters an attack button which I explained further in the code below, their box collider trigger will be activated.



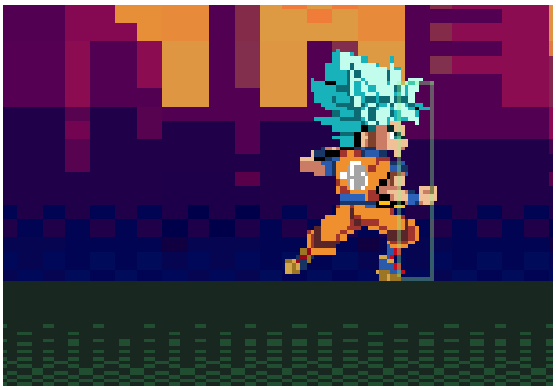



```
using UnityEngine;
using System.Collections;

0 references
public class Player1Atk : MonoBehaviour{

    bool punch = false;
    bool kick = false;
    // the above variables set punch and kick to false
    private float atkTimer = 0;
    private float atkCd = 0.3f;
    // the float variables are used for the delays between attacks
    public Collider2D player1_trigger; // this references the collider that is the child of the gameObject which is the character
    public Animator animator;
    0 references
    void Awake()
    {
        animator = gameObject.GetComponent<Animator>();
        player1_trigger.enabled = false;// this sets the trigger in the begging of the game to false

    }

    0 references
    void Update()
    {

        if (Input.GetKeyDown("i") && !punch)
        {
            punch = true;
            atkTimer = atkCd;
            // this checks if the player enters the button punch button which is i, if so then punch becomes true


            player1_trigger.enabled = true;
            // the trigger then also becomes true



        }

        if (punch)
        {
            if (atkTimer > 0 )
            {
                atkTimer -= Time.deltaTime;// if the atk timer is bigger then 0 then this line will decrease the time by delta time which works as a real timer
            }
            else
            {
                punch = false;
                player1_trigger.enabled = false;
                // if player is not usinf the punch key button then punching becomes false and the trigger will also be false
            }
        }

        animator.SetBool("punch",punch);
        // this will set the punch animation in the animator of player 1 to true or false depending on the two if statements above
```

This script checks if the player has pressed the button “I” and they aren’t already punching, if so then punch becomes true and the punch trigger is enabled. There will also be a timer that will countdown from 0.3 seconds to punch again. If the player isn’t punching, then punch animation becomes false. The animator will play the punch depending on the 2 if statements for punch. The attack timer will be adjusted in the future depending on how the game plays.

Testing activation of box collider with punch animation

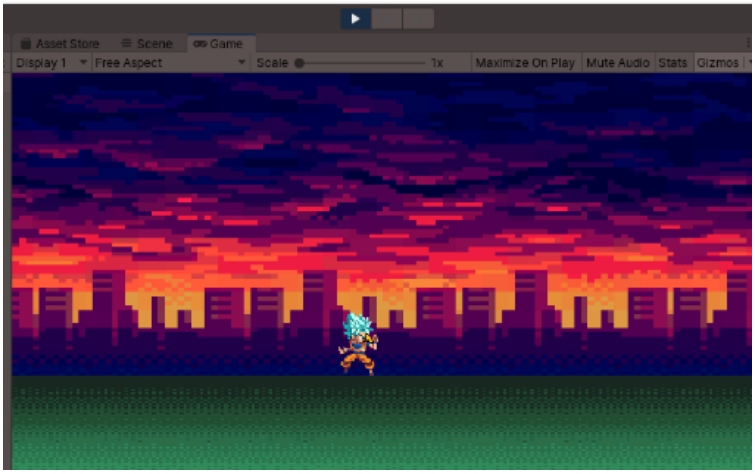
1	Fight system/ box colliders	Use inputs assigned to each character move and check if box colliders are being activated	I,O,P,J,K,L	Good: “I” Bad: “v” Boundary: n/a	See if the attacks are activating /colliding with other character’s box colliders	The box collider for detecting other characters activates when input for punch is entered	none

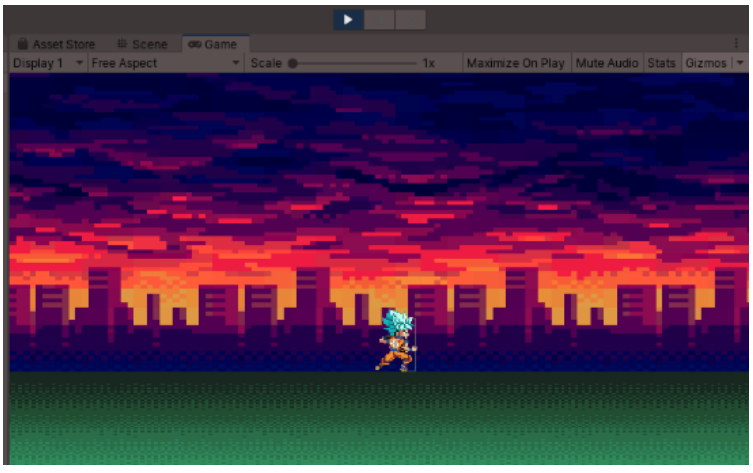


The collider in front of the character is also show on the Left image.

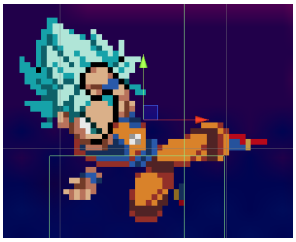
3	Attack animations	Test if animations play if conditions are met	I,O,P	Good: “I” Bad: “G” Boundary: N/A	See if the animations for specific attacks are displayed	When the punch button is entered the animation is played accordingly	none

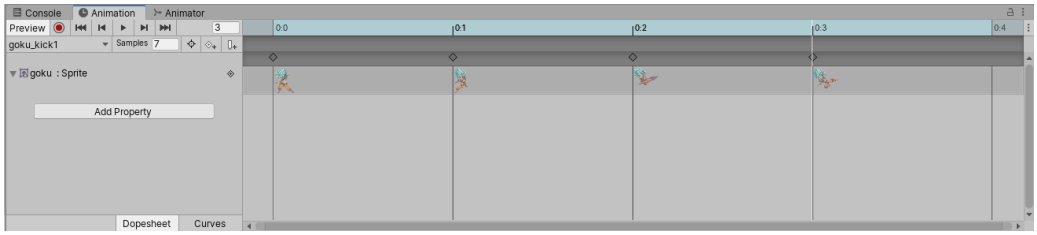
Idle position transitions to -> punch





Kick animation

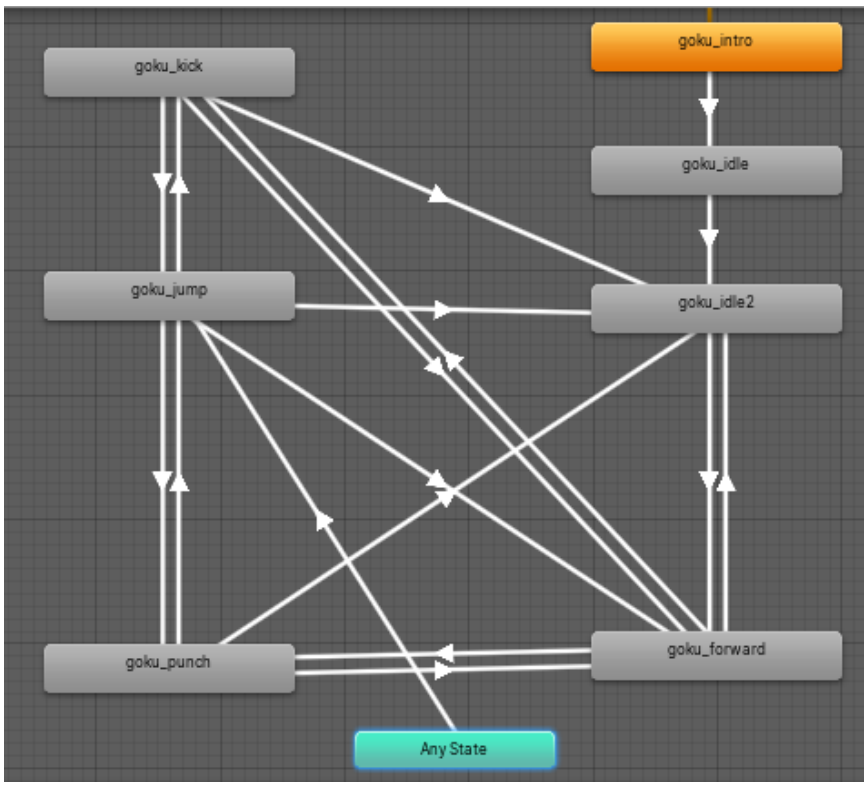





```
if (Input.GetKeyDown("o") && !kick)
{
    kick = true;
    atkTimer = atkCd;
    // this checks if the player enters the button punch button which is i, if so then punch becomes true

     player1_trigger.enabled = true;
     // the trigger then also becomes true
}

if (kick)
{
     if (atkTimer > 0)
     {
         atkTimer -= Time.deltaTime;// if the atk timer is bigger then 0 then this line will decrease the time by delta time which works as a real timer
     }
     else
     {
         kick = false;
         player1_trigger.enabled = false;
         // if player is not usinf the punch key button then punching becomes false and the trigger will also be false
     }
}

animator.SetBool("kick", kick);
// this will set the punch animation in the animator of player 1 to true or false depending on the two if statements above
```

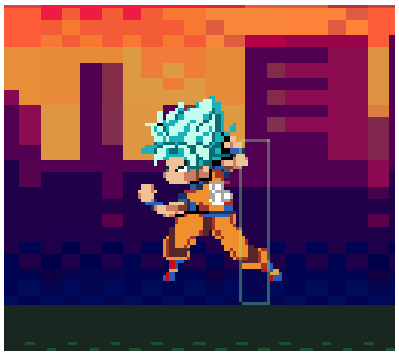
This script checks if the player has pressed the button “I” and they aren’t already punching, if so then punch becomes true and the punch trigger is enabled. There will also be a timer that will countdown from 0.3 seconds to punch again. If the player isn’t punching, then punch animation becomes false. The animator will play the punch depending on the 2 if statements for punch. The attack timer will be adjusted in the future depending on how the game plays.

This is how the animator currently looks with the kick and punch animations. I adjusted their exit time and duration by testing how the animation play when I enter them. The attacks are playing and they transition back to goku_idle2. They also become false in the air when jump is active since I made the conditions to attack only is jump is false.



Testing activation of box collider with kick animation

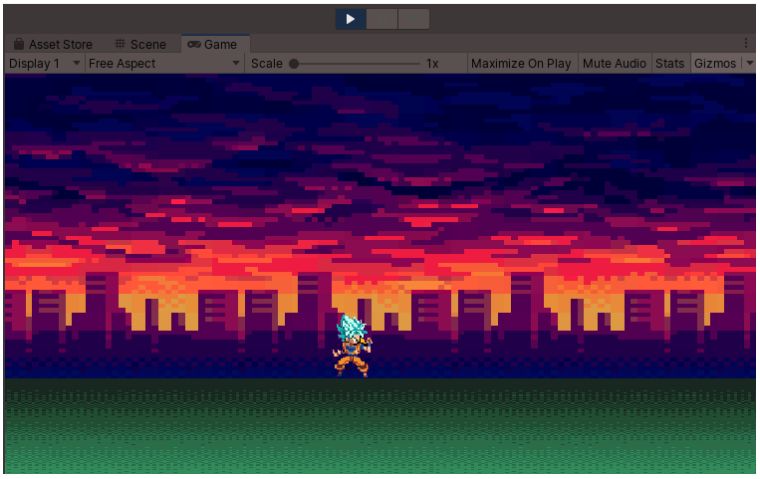
2	Fight system/ box colliders	Use inputs assigned to each character move and check if box colliders are being activated	I,O,P,J,K,L	Good: “I” Bad: “v” Boundary: n/a	See if the attacks are activating /colliding with other character’s box colliders	The box collider for detecting other characters activates when input for kick is entered	none

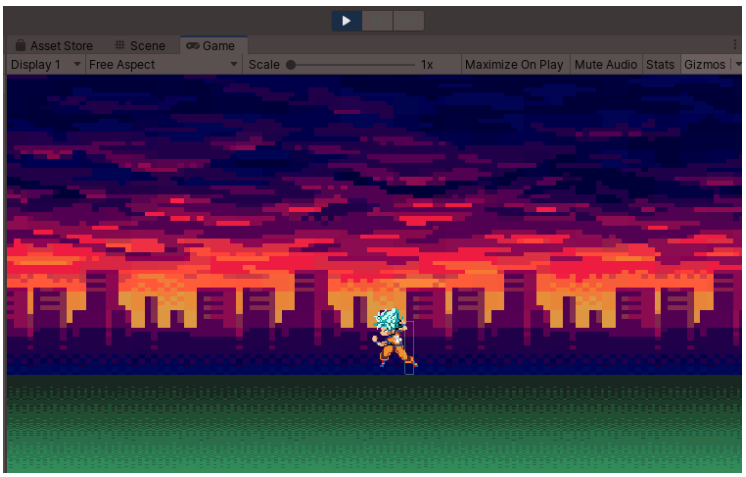


The box collider is shown In front of the character when button for kick is entered on the left image.

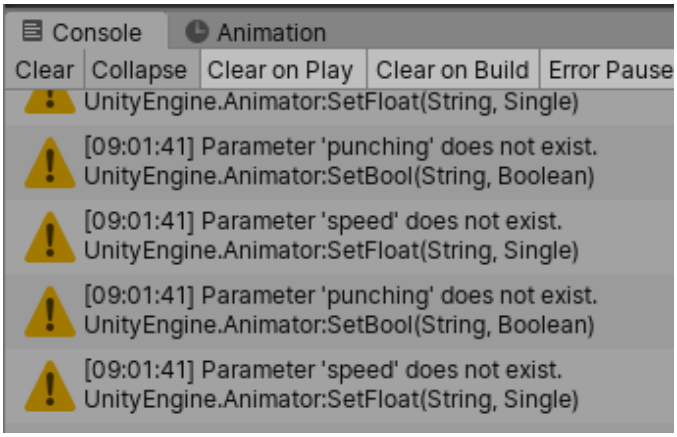
3	Attack animations	Test if animations play if conditions are met	I,O,P	Good: “I” Bad: “G” Boundary: N/A	See if the animations for specific attacks are displayed	When the kick button is entered the animation is played accordingly	none

idle position transitions to -> kick animation





Error



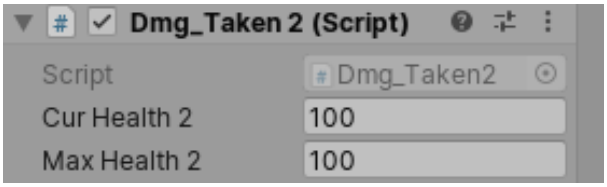
This warning appeared when I try to make the punching animation play whenever the punch button is pressed. Even though the name of the animation that I referenced in the code was exactly the same as the one in the animator it still didn't play the animation. After researching on how to fix the issue people had the same problem and called it a bug. Eventually I just changed the name of the parameter to punch and it worked! I don't know if there is a certain rule that comes to naming parameters but if this issue appears again I will just try renaming the parameter.

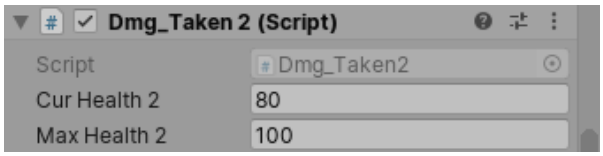
Health system

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

2 references
public class dmg_Taken : MonoBehaviour
{
    public int curHealth;
    public int maxHealth = 100;

    0 references
    void start()
    {
        curHealth = maxHealth;// this sets the player2 helath to 100

    }

    0 references
    void Update()
    {

        if (curHealth <= 0)
        {
            Destroy(gameObject);
        }
        // if the player health is below 0 then the player2 will be destroyed in the game
    }
    1 reference
    public void Damage(int damage)
    {
        curHealth -= damage;
        // this function will be refernced by the player1 trigger whenever player1 is attacking player2 hp will decrease the amount of damage it sends to this script

    }
```

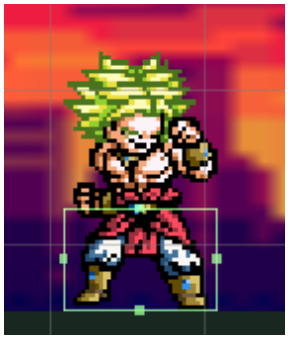
This script is made to give the characters a health system so that when they are attacked they will have a health points out of 100. Punch and kick are worth 5 points. If their health is below 0 then they will be destroyed in the game. This is just temporary to check if they are below 0 but in the future I will make it that they will have a knocked out animation. The "Damage" at the bottom will be used by the attack trigger scripts that I have for player 1 and 2. So that when the trigger is active they will subtract 5 from this health script until the character is dead. This was used on Broly first to check if player1 can damage him.

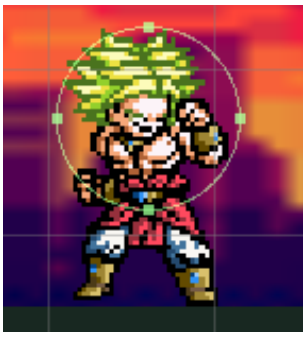
To check if Goku will deal damage to Broly I will check if "Cur Health 2" will decrease on the right. I mad it a public variable so that I can access it while making the game and it will also allow me if the character are taking damage.



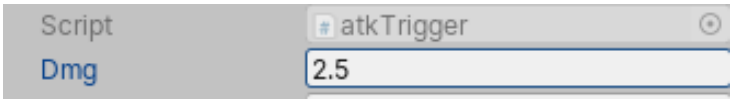


After 2 hits the health points went down by 20 even though each hit is worth 5.



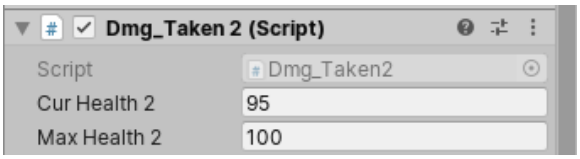


After trying to figure the problem and realised that there were 2 colliders which I added so that the other player won't stay on top of Broly when he jump on top. The circle collider will make opponents slide off. To solve this I changed the int variable "dmg" to a float and reduced the 5 to 2.5 so that even if it doubles the damage given it will still give 5 damage. The reason why the damage was 10 not 5 is because the 2 colliders were regarded as 2 objects in 1 so the damage was doubled.



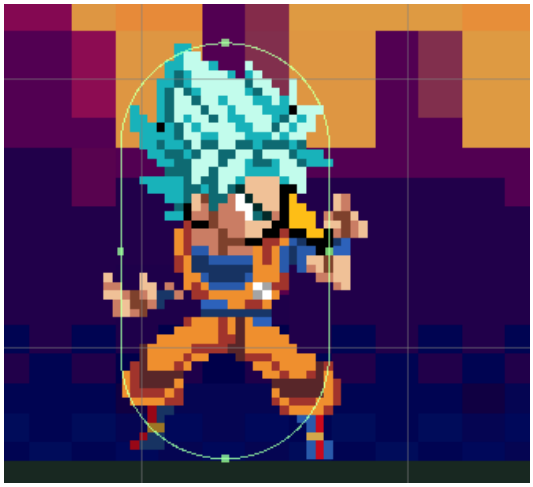
Even after changing the value it decreased by 4, to solve this I went into the dmg_taken script and changed the "Cur Health 2" and "Max Health 2" to float values so that it can detect the left over decimal 0.5. the screenshot below shows that it worked.

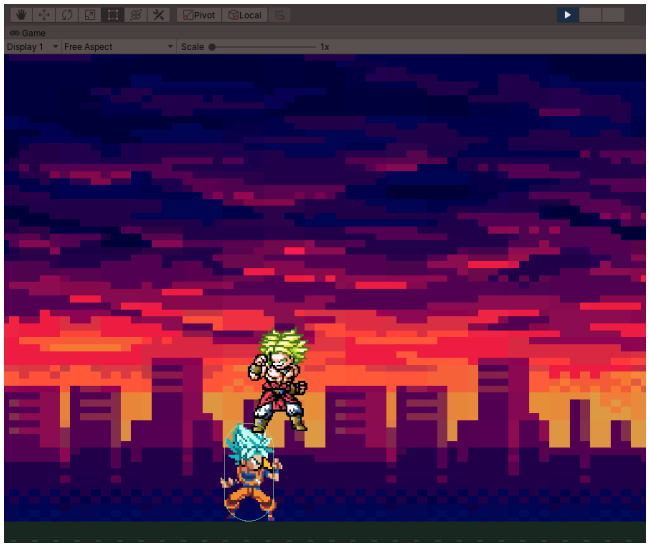
```
public float curHealth2 = 0;
public float maxHealth2 = 100;
public Animator animator;
bool dmg_anim2 = false;
```



Updated version of colliders

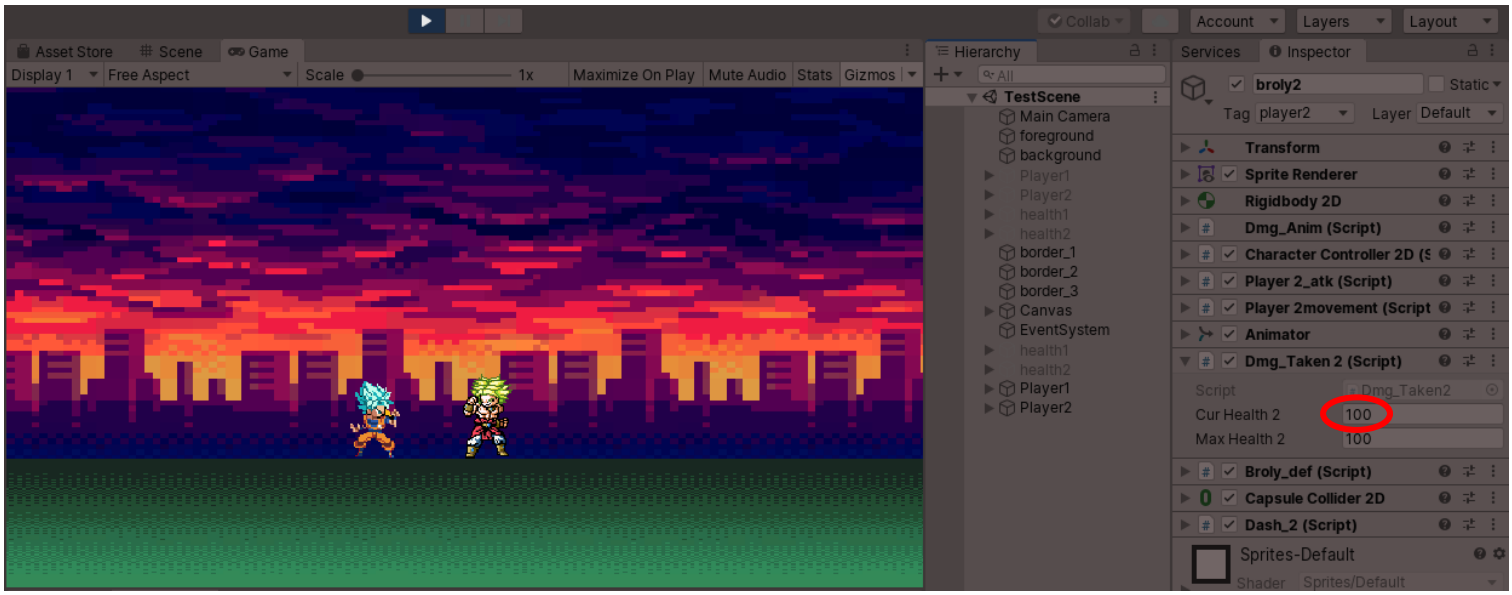
I found out that capsule colliders worked better since if I want to tweak the mass settings or speed of the characters the 2 different colliders I had before behaved in a weird way that didn't let me control how the characters should move. It also has the circular shape at the top so it will partially stop the issue of characters standing on top of each other. On the right image you can see that Broly is sliding of the edge of the collider instead of standing on top.

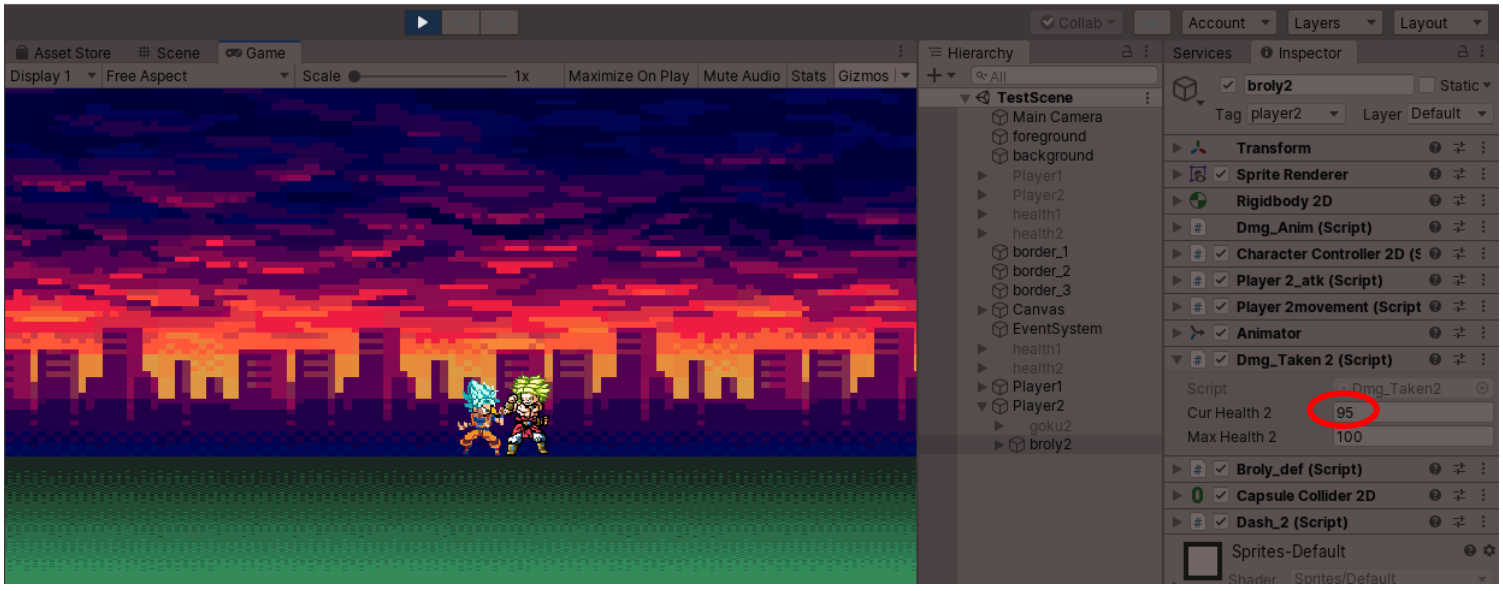




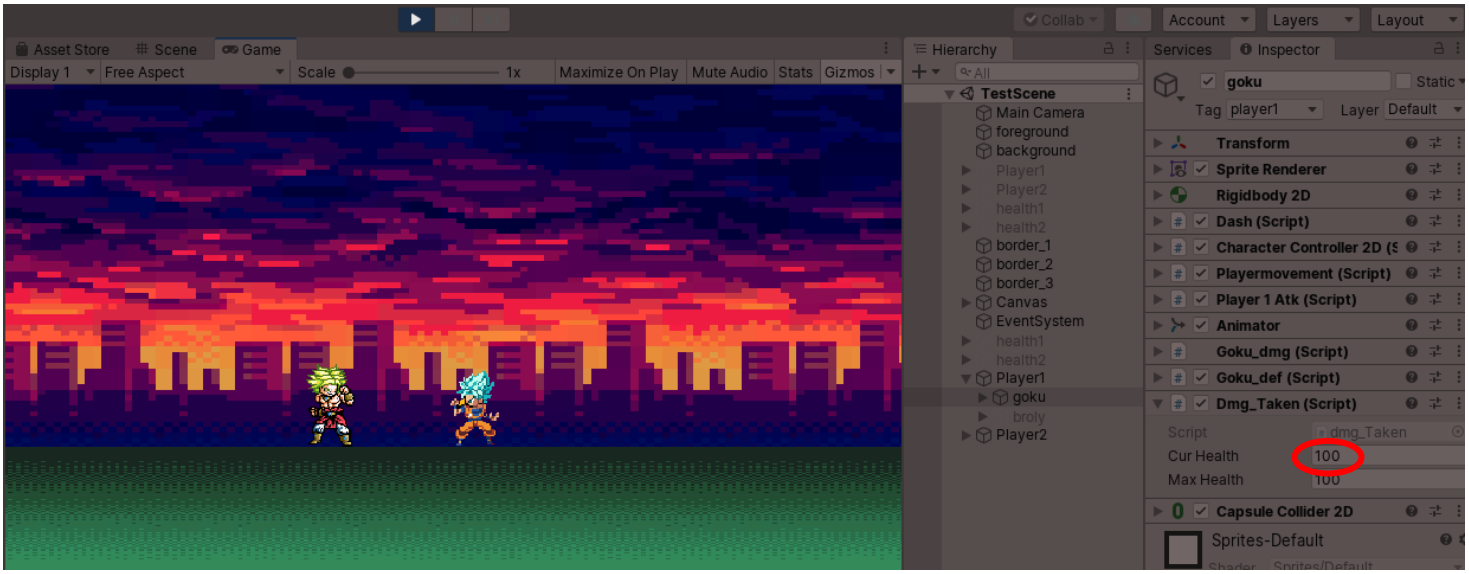
Testing health system

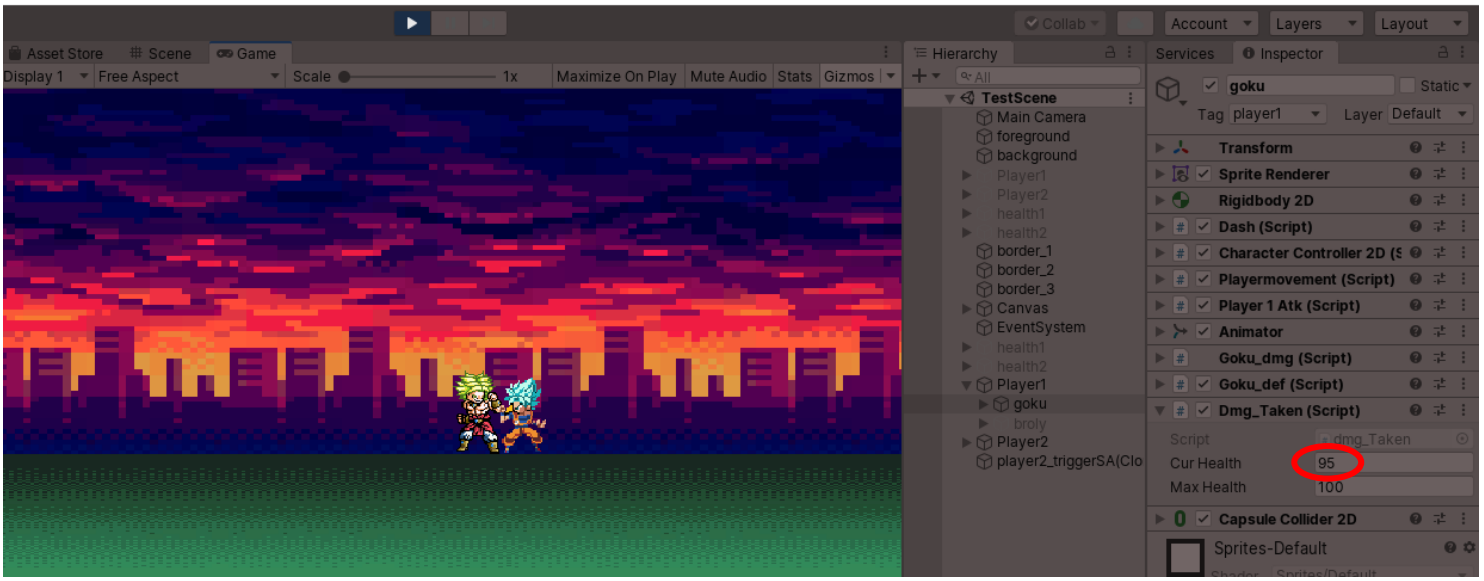
13	Health system	Check if the user and enemy health is decreased after taking damage	Use attacks to see if enemy is taking damage and check user is taking damage by letting enemy attack	**Good**: player1 health decreases after attack **Bad:** no damage dealt **Boundary**: health staying the same after being attacked in defending state	The health of both the opponents should decrease by the amount the attacker is applying to them. Example kick should do 5 damage and super should do 10	After Broly took 1 punch from Goku his health decreased by 5	none





From the first image the health of Broly is at 100, in the 2nd image after being hit the health goes down to 95. This is also repeated for the Goku taking damage below. The test for damage was successful for both characters.

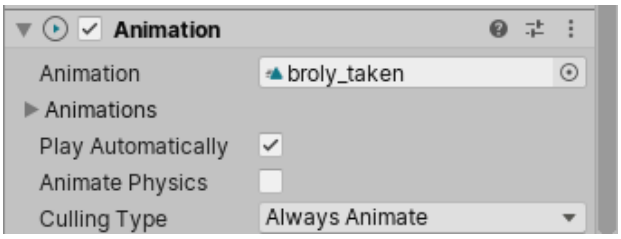




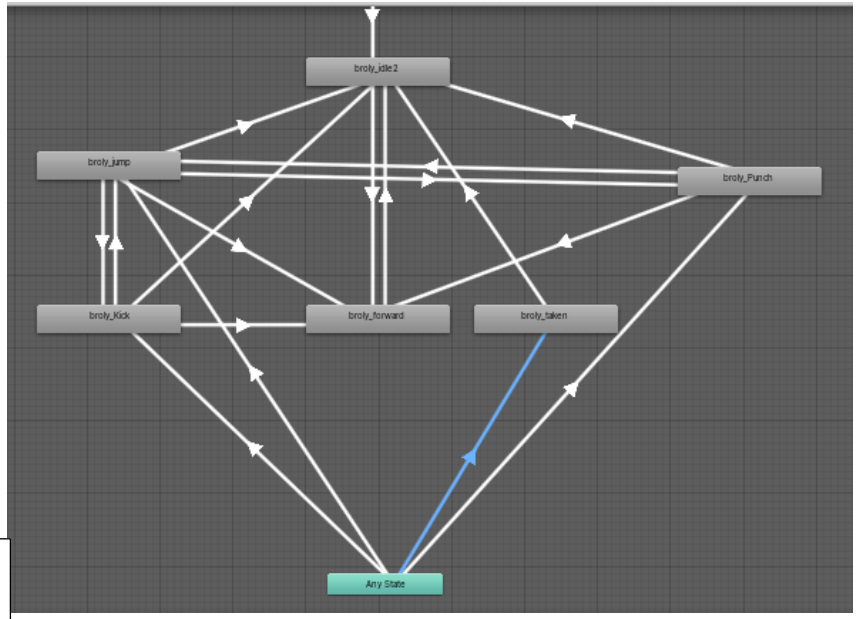
Damage Animation


```
0 references
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.tag == "player2")
    {
        animator.SetTrigger("Hit");
    }
    else
    {
        animator.SetTrigger("idle2");
```

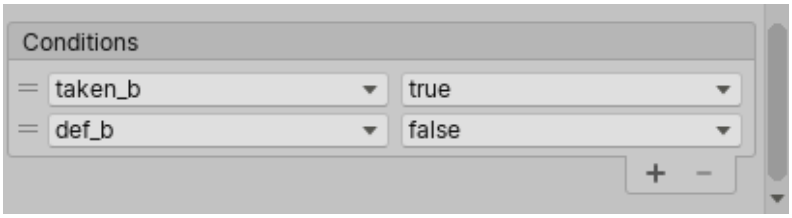
This function I tried the first time did not work and didn't play any animation when tested on. I was first testing If a damage taking animation will play if player1 came into contact with player2.




```
0 references
public class dmg_Anim : MonoBehaviour
{
    public Animator onHit;
    0 references
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("player1_trigger"))
        {
            onHit.SetBool("taken_b", true);
        }
    }

    0 references
    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.CompareTag("player1_trigger"))
        {
            onHit.SetBool("taken_b", false);
        }
    }
}
```

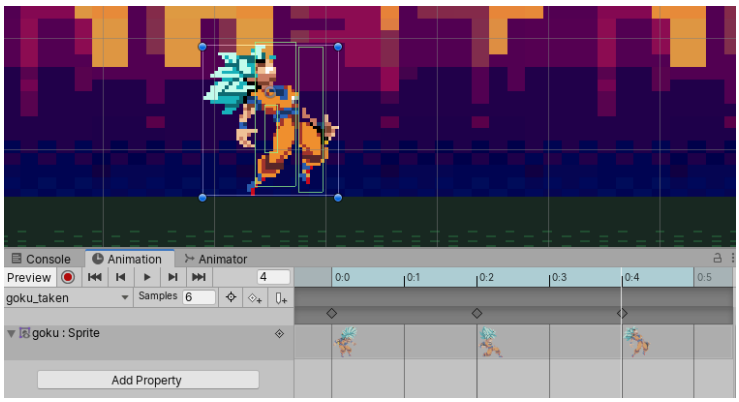


This is section is where I tried to make Broly play an animation whenever he is hit. The script above works similar to the attack trigger from page 50 but when the attack trigger from Goku is within Broly's sprite range then the animation for taking damage will be played. A new function I also used is "OnTriggerExit2D", this will play when the attack trigger is deactivated so that the character will stop playing the damage when not being hit.



Damage animation

I used an “animation” component that would play the damage taken animation however even just for testing if another collider comes in contact to trigger the animation it didn’t work. So, I used the animator on the player taking the damage, created a script that detects the attack trigger of the other player so that the character will play the hurt animation whenever the other player is attacking. I repeated this process for Goku as well.



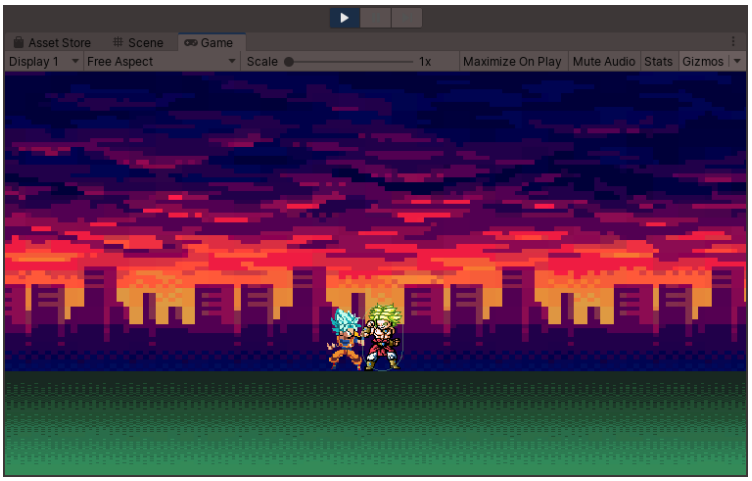

```
/* private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.tag == "player2")
   {
        animator.SetTrigger("Hit");
    }
    else
     {
         animator.SetTrigger("idle2");

     }
}*/
```

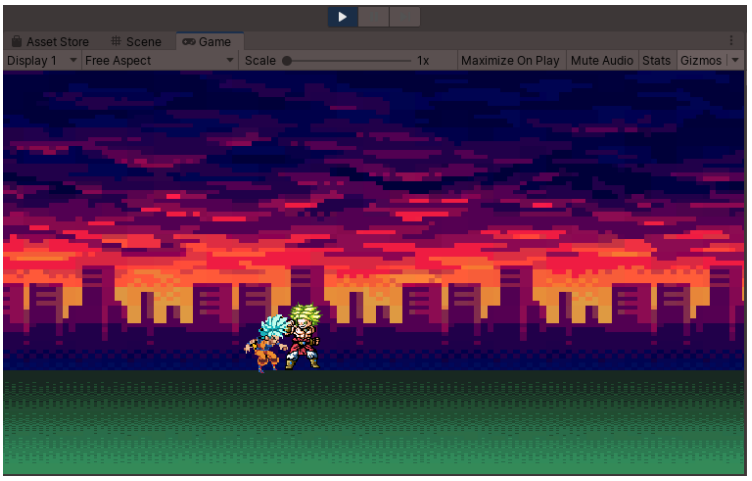

```
public class goku_dmg : MonoBehaviour
{
    public Animator onHit;
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("player2_trigger"))
        {
            onHit.SetBool("taken_g", true);
        }
    }

    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.CompareTag("player2_trigger"))
        {
            onHit.SetBool("taken_g", false);
        }
    }
}
```

Test for damage animation and health

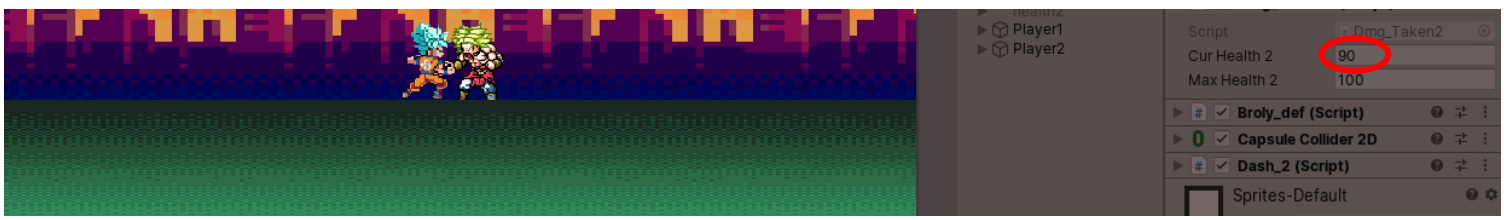
3	Animations	Test if animations play if conditions are met	I,O,P	Good: “I” Bad: “G” Boundary: N/A	See if the animations for specific conditions are displayed	When Goku is attacked by Broly then the animation for taking damage is played	None

idle position **transitions to ->** **damage animation**

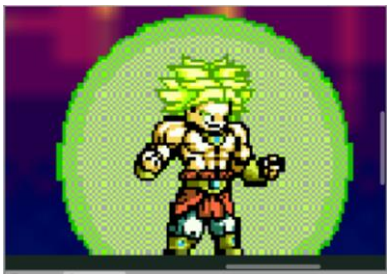


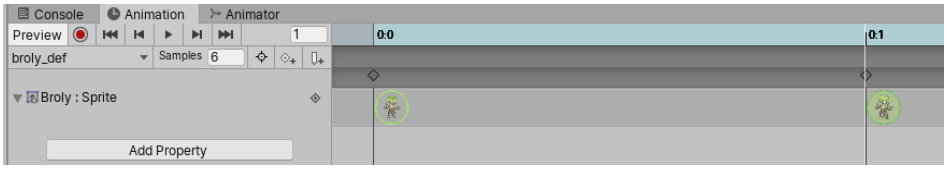


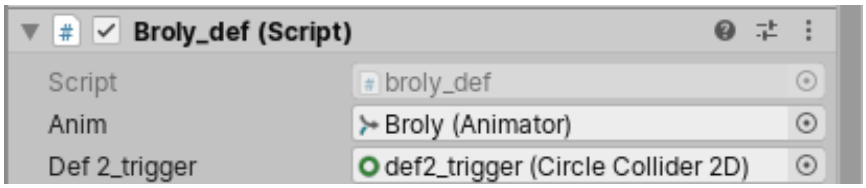
13	Health system	Check if the user and enemy health is decreased after taking damage	Use attacks to see if enemy is taking damage and check user is taking damage by letting enemy attack	**Good**: player1 health decreases after attack **Bad:** no damage dealt **Boundary**: health staying the same after being attacked in defending state	The health of both the opponents should decrease by the amount the attacker is applying to them. Example kick should do 5 damage and super should do 10	After Broly took 2 punches from Goku his health decreased by 10	none



Defence/guard








```
public Animator anim;
bool def_b = false;
public CircleCollider2D def2_trigger;
public Collider2D def_trigger;
// both of the above triggers will be used to reference player2's colliders that will be activated when the defend button is activated
private float defTimer = 0;
private float defCd = 3;
// the 2 float variables above will be used to set the countdown of how long the def barrier lasts for

 0 references
 void Awake()
{
    def2_trigger.enabled = false;
    anim = gameObject.GetComponent<Animator>();
}

0 references
void Update()
{

    if (Input.GetKeyDown("1") && !def_b)
    {
        def_b = true;
        def2_trigger.enabled = true;
        // if the user has entered the guard key then defend will be true and so will the trigger

        defTimer = defCd;// this will make the barrier last for 3 seconds
        transform.gameObject.tag = ("def_2");
        // this line of code will change the player2's tag from player2 to def_2 this way when player1 attacks it will detect the tag and cancel the damage being dealt
        Debug.Log(tag);
        print(tag);
        // the 2 lines above will print to the console if the tag has changed


    }

    if (def_b)
    {
        if (defTimer > 0)
        {
            defTimer -= Time.deltaTime;
            // this will act as a timer for the guard barrier to go down
        }
        else
        {
            def_b = false;
            def2_trigger.enabled = false;
            transform.gameObject.tag = ("player2");
            // if player 2 isnt defending anymore then teh tag will go back to player2 and player1 can return to doing damage
        }

        anim.SetBool("def_b", def_b);
        // this line will play the aniimtion of the guard barrier depending on whether player 2 has entered the guard button
```

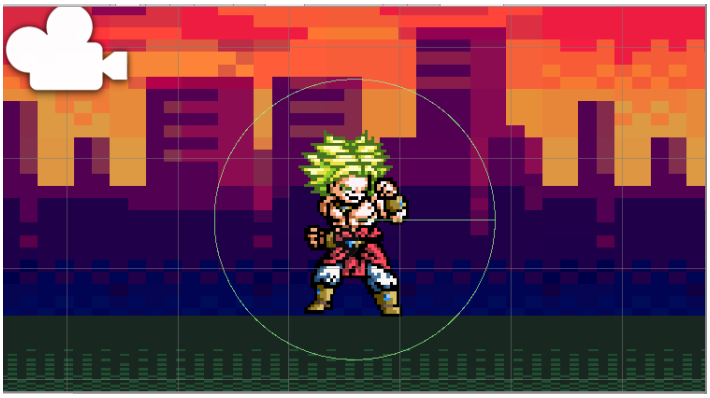
This is the script that checks if the player has pressed the defend button, if so the animation for defending will play and the damage taken will be cancelled. Since the animation forms a barrier around Broly I used a circle collider to make sure that any attack within that area will be negated. How this script cancels damage is that whenever the character is defending the object tag of the character will change from “player2” to “def_2”, this will make the “atkTrigger” script useless since it only checks for “player2” to apply damage.

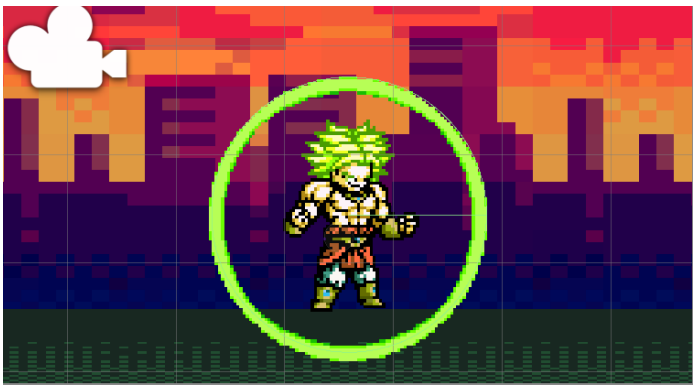
```
    void OnTriggerEnter2D(Collider2D col)// this is a sunction that checks any 2d colliders that it come in contact with
    {
        if (col.isTrigger != true && col.CompareTag("player2"))
            //if the box collider this script is attached to is triggered through punch or kick and the contact object has a tag of player2
```

Test for activation of collider and animation for defend

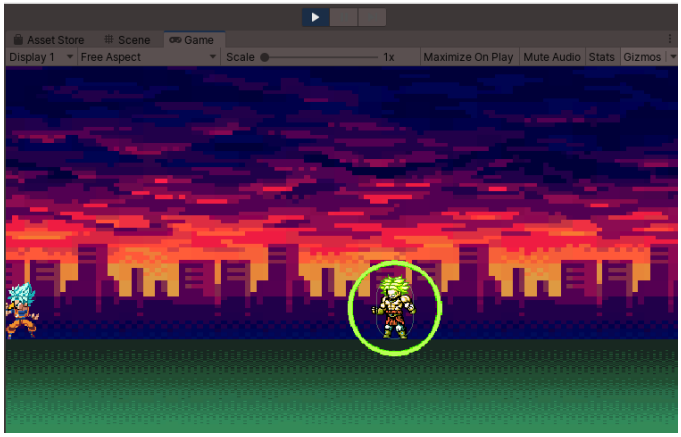
3	Animations	Test if animations play if conditions are met	I,O,P	Good: “I” Bad: “G” Boundary: N/A	See if the animations for specific conditions are displayed	When the defence button is pressed the animation for it plays accordingly	None

Idle position **transitions to ->** **defend animation**





11	Defend	Use input to defend against enemy	Input K	**Good:** if the damage dealt is cancelled **Bad**: damage not cancelled **Boundary:** damage dealt after defence is disabled	The user should be able to defend and negate the damage that is being dealt to them	The damage isn’t being nullifies even if the defend animation is playing	Need to fix trigger for guarding attacks



6	Defend	Use input to defend against enemy	Input K	Good: if the damage dealt is cancelled Bad: damage not cancelled Boundary: damage dealt after defence is disabled	The user should be able to defend and negate the damage that is being dealt to them	The damage is now being nullified for the amount of time I wanted in the script	none

Double Dash

This is the animation when the player double dashes. It will create a teleporting effect.

```
0 references
public class Dash : MonoBehaviour
{
    private Rigidbody2D rb;
    public float dashSpeed;
    private float dashTime;
    public float startTime;
    private int direction;
    public int rightTotal = 0;
    public float rightTime = 0;
    public int leftTotal = 0;
    public float leftTime = 0;
    Animator anim;

    0 references
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        dashTime = startTime;
        anim = GetComponent<Animator>();
    }
    0 references
    void Update()
    {

        if( direction == 0)
        {
            if (Input.GetKeyDown(KeyCode.D))
            {
                rightTotal += 1;
                // if the player enters the right key which is D then the count for how many times they pressed that will increase by 1
            }

             if((rightTotal ==1) && (rightTime < 0.2))
                rightTime += Time.deltaTime;
             //if the count for how many times they pressed they left key is 1 and the time for long it has been since they pressed it is less than 0.2 seconds
             // then the timer activates

                if((rightTotal == 1) && (rightTime >= 0.2))
                {
                    rightTime = 0;
                    rightTotal = 0;
                  // if the count for left key pressed is still 1 and has been more than or equal to 0.2 seconds then count goes back to 0 so will the timer

                    anim.SetBool("goku_dash", false);
                  // this sets the double dash to false if the the above if statement is true
```

The script above will enable the player to double dash. It works by checking the value of “direction”. If the value Is 0 then nothing happens. If the value of direction is 2 which is equal to moving right, then it runs the code above to meet the requirements before playing the animation. This works the same for the left double dash of the direction which is equal to 1.

After the first key is pressed a counter starts and if the same key is pressed again before that counter finishes, the character will perform double dash that increases their velocity and the script also triggers the animation that makes them look like they are leaving an after image.

```
if ((rightTotal == 2) && (rightTime < 0.2))
{
    direction = 2;
    rightTotal = 0;
    anim.SetBool("goku_dash", true);
    //if left key is pressed twice and the time that has elapsed after the first count is still less than 0.2 seconds then the double dash animtion plays for player1

}

else if (Input.GetKeyDown(KeyCode.A))
{
    leftTotal += 1;
    // if the player enters the right key which is D then the count for how many times they pressed that will increase by 1
}

if ((leftTotal == 1) && (leftTime < 0.2))
    leftTime += Time.deltaTime;
//if the count for how many times they pressed they left key is 1 and the time for long it has been since they pressed it is less than 0.2 seconds
// then the timer activates

if ((leftTotal == 1) && (leftTime >= 0.2))
{
    leftTime = 0;
    leftTotal = 0;
    // if the count for left key pressed is still 1 and has been more than or equal to 0.2 seconds then count goes back to 0 so will the timer

    anim.SetBool("goku_dash", false);
    // this sets the double dash to false if the the above if statement is true

}

if ((leftTotal == 2) && (leftTime < 0.2))
{
    direction = 1;
    leftTotal = 0;
    anim.SetBool("goku_dash",true);
    //if left key is pressed twice and the time that has elapsed after the first count is still less than 0.2 seconds then the double dash animtion plays for player1
}
```

```
// the code below will handle how fast the speed is of the double dash

}
else
{
    if(dashTime <= 0)
    {
        direction = 0;
        dashTime = startTime;
        rb.velocity = Vector2.zero;
    }
    else
    {
        dashTime -= Time.deltaTime;
        if(direction == 1)
        {
            rb.velocity = Vector2.left * dashSpeed;
        }else if( direction ==2)
        {
            rb.velocity = Vector2.right * dashSpeed;
        }

       /* else if (direction == 3)
        {
            rb.velocity = Vector2.up * dashSpeed;
        }
        else if (direction == 2)
        {
            rb.velocity = Vector2.down * dashSpeed;
        }*/
    }
}
```

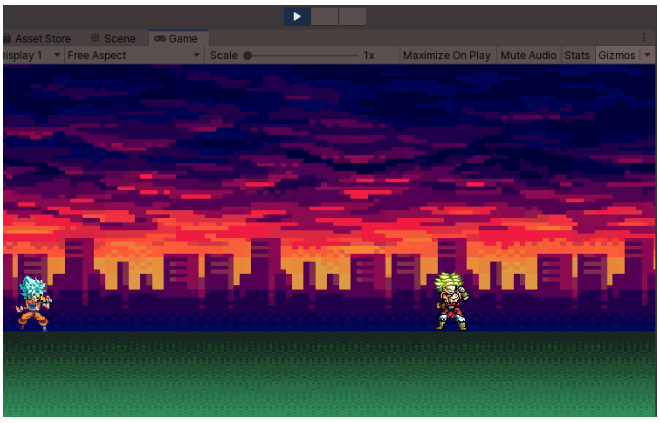
The code above decides how fast the double dash is. If the direction is 0 no speed is applied to the character movement because there is no double dash being performed. If the direction is 1 (which Is the left key) then the velocity of the rigidbody will be multiplied by dashSpeed. Since I made the variable “dashSpeed” public, I can control how fast the double dash is from the inspector. This works the same way for direction 2 which is the right key.

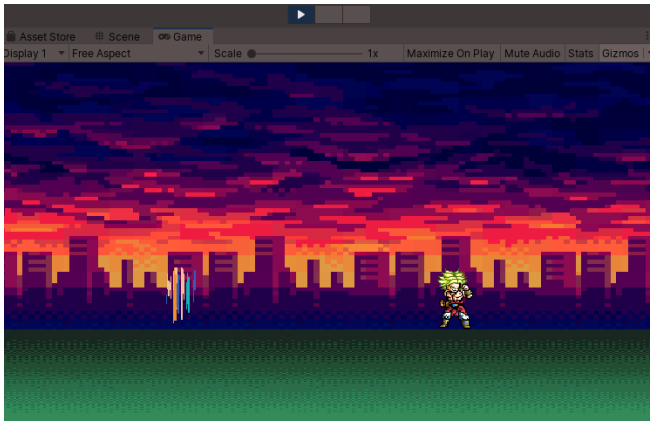
Test for dash animation and movement

3	Animations	Test if animations play if conditions are met	I,O,P	Good: “I” Bad: “G” Boundary: N/A	See if the animations for specific conditions are displayed	The dash animation for both Goku and Broly plays when I double tap left or right	none

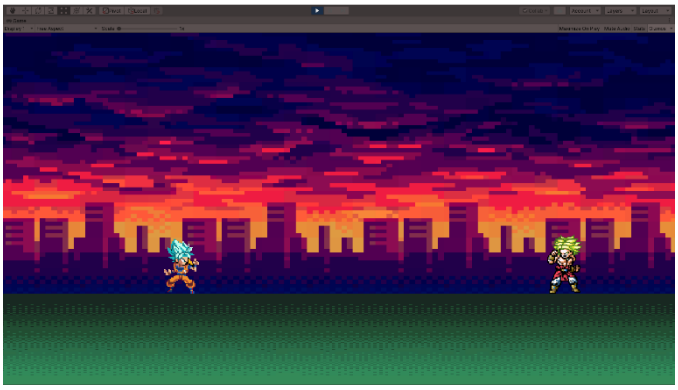
4	Movement	Use the horizontal and vertical inputs to see if character moves	W,S,A,D	Good: “W” Bad: “B” Boundary: N/A	The user will be moving in response to each of the user inputs so will the enemy	When left or right is double tapped both characters move in the direction that was input	none

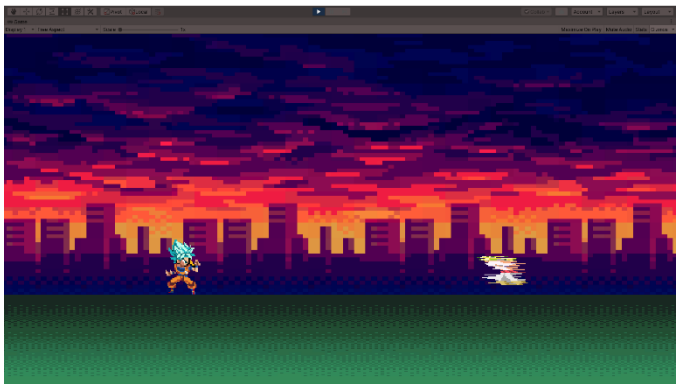
Goku idle **transitions to ->** **dash animation**



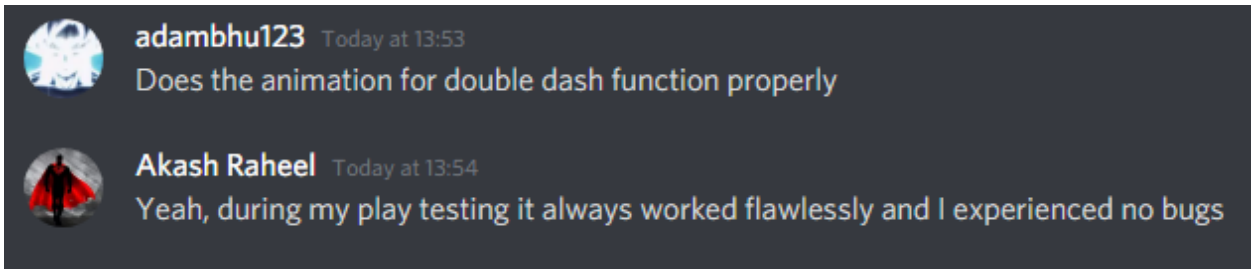


Broly idle **transitions to ->** **dash animation**



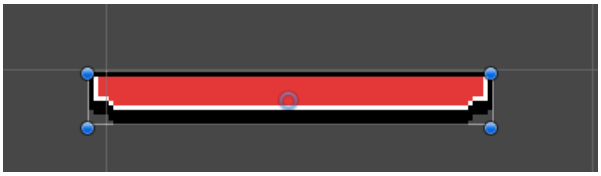


User feedback

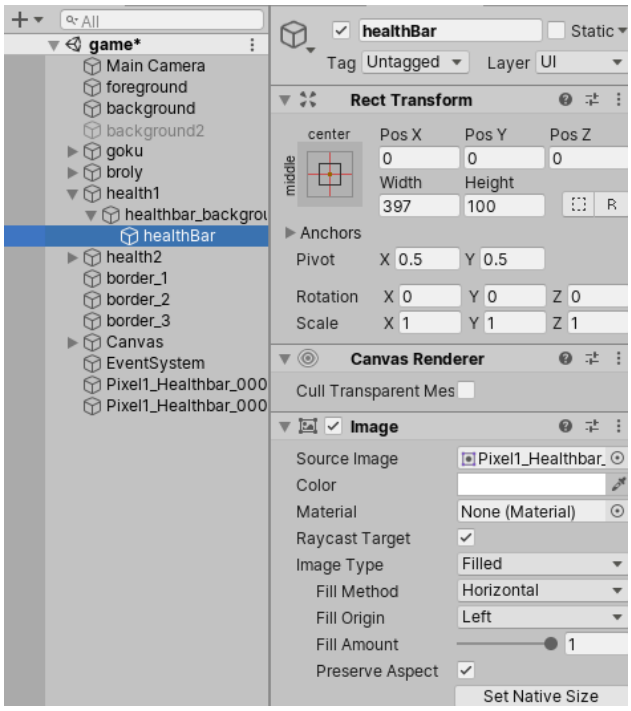


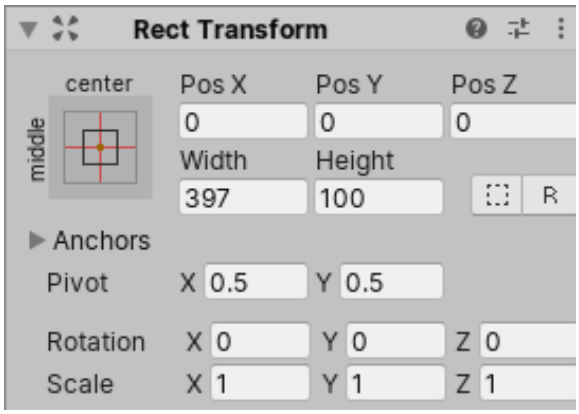
The user likes how the dash feels and has confirmed that it functions well without any bugs

Health bars

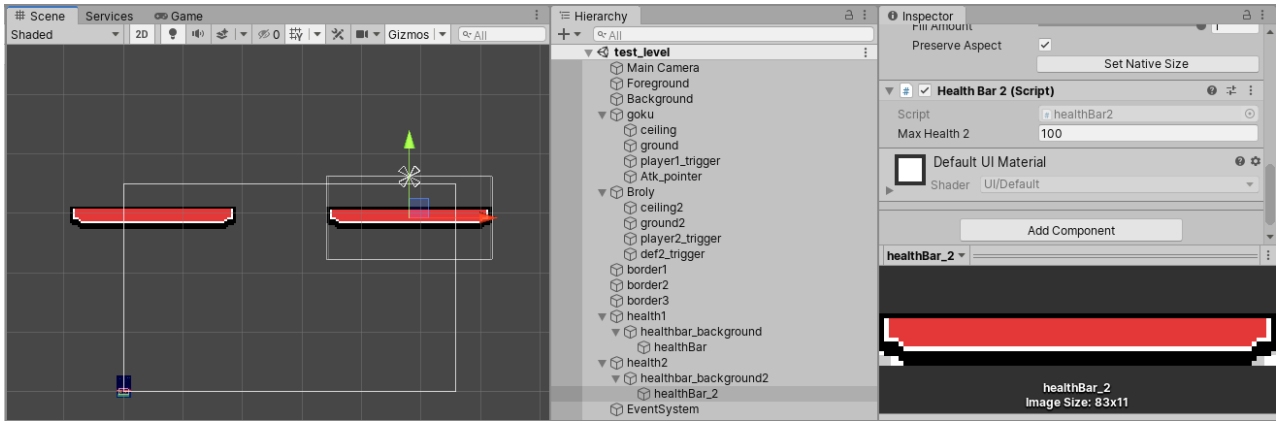


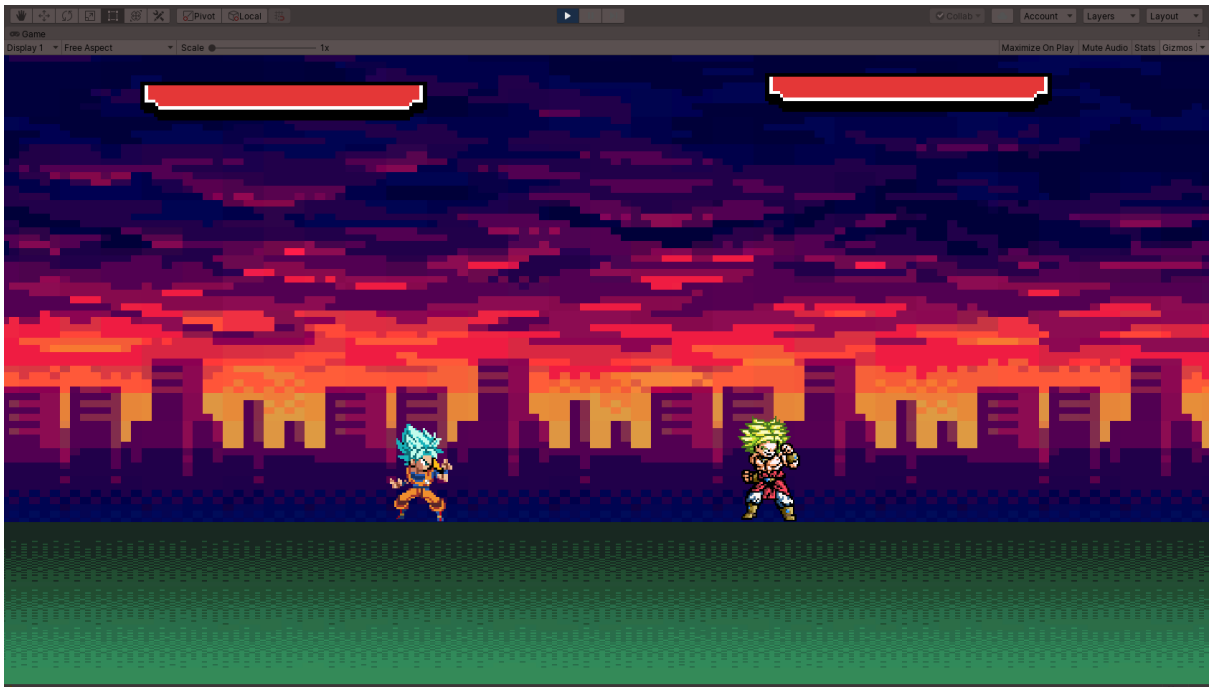
To make the health bars I first took an empty health bar sprite as a back ground. The health bar on the right will be used to fill It. In the inspector the image of the 2nd sprite will be a filled image type so that it will act like a bar that reduces horizontally. I also set the fill origin to left so the bar will move towards the left.





I repeated this process again for two health bars by duplicating the first one and dragging it across to the right and I used the Rect Transform to position them in the canvas.





```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class healthbar : MonoBehaviour
{
    Image healthBar;
    public float maxHealth = 100f;
    public static float health;

    void Start()
    {
        healthBar = GetComponent<Image>();
        health = maxHealth;
    }

    // Update is called once per frame

    void Update()
    {
        healthBar.fillAmount = health / maxHealth;
    }
}
```

This script is used to make the health bar image in the game to reduce by the attack value whenever either the player 1 or 2 gets hit. It will also be used as a reference by the attack trigger whenever the hp of the character themselves are reduced the image will also get shorter.

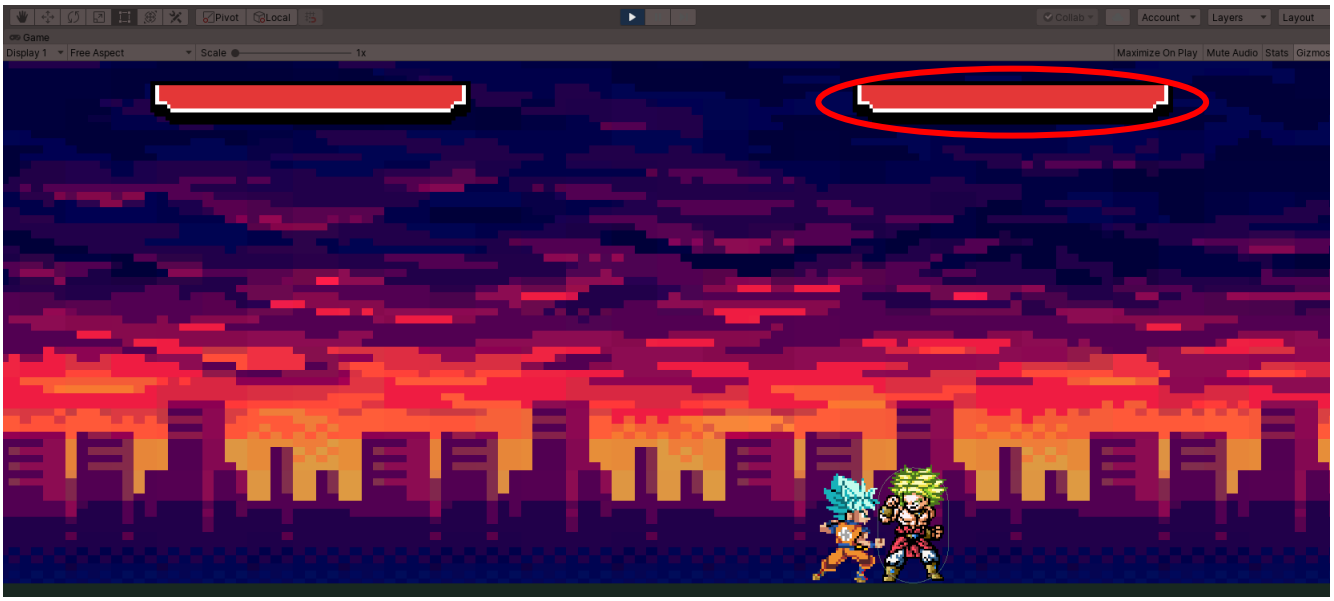
The reference is the script below when the attack trigger collides with the other player, the health bar script is referenced.

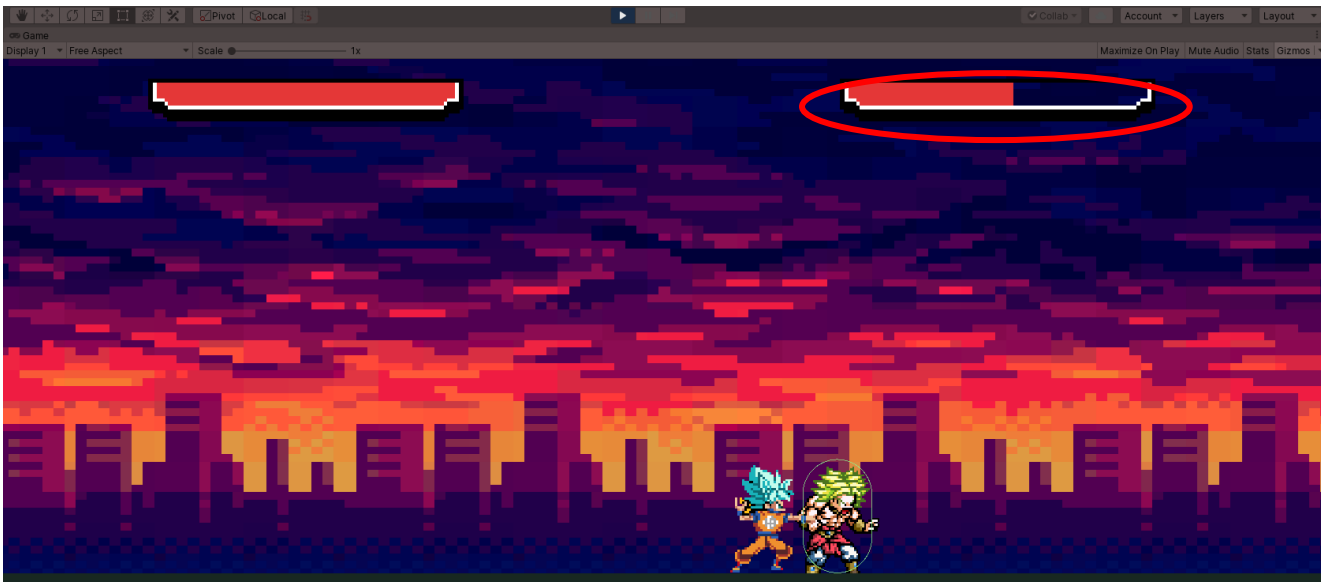
```
public class atkTrigger : MonoBehaviour
{
    public int dmg = 5;
    public int dmg_SA = 10;

    void OnTriggerEnter2D(Collider2D col)// this is a sunction that checks any 2d colliders that it come in contact with
    {
        if (col.isTrigger != true && col.CompareTag("player2"))
            //if the box collider this script is attached to is triggered through punch or kick and the contact object has a tag of player2
        {
            col.SendMessageUpwards("Damage", dmg);
            healthBar2.health2 -= 5f;
            // Damage function will be called and subtract 5 from the other player health in the player2 script which is called Dmg_taken
        }
```

Test for health bars

13	Health system	Check if the user and enemy health is decreased after taking damage	Use attacks to see if enemy is taking damage and check user is taking damage by letting enemy attack	**Good:** player1 health decreases after attack **Bad:** no damage dealt **Boundary:** health staying the same after being attacked in defending state	The health of both the opponents should decrease by the amount the attacker is applying to them. Example kick should do 5 damage and super should do 10	The health bar for Broly decreases proportionally to the amount of times hit by Goku this is also the same for Broly hitting Goku	none

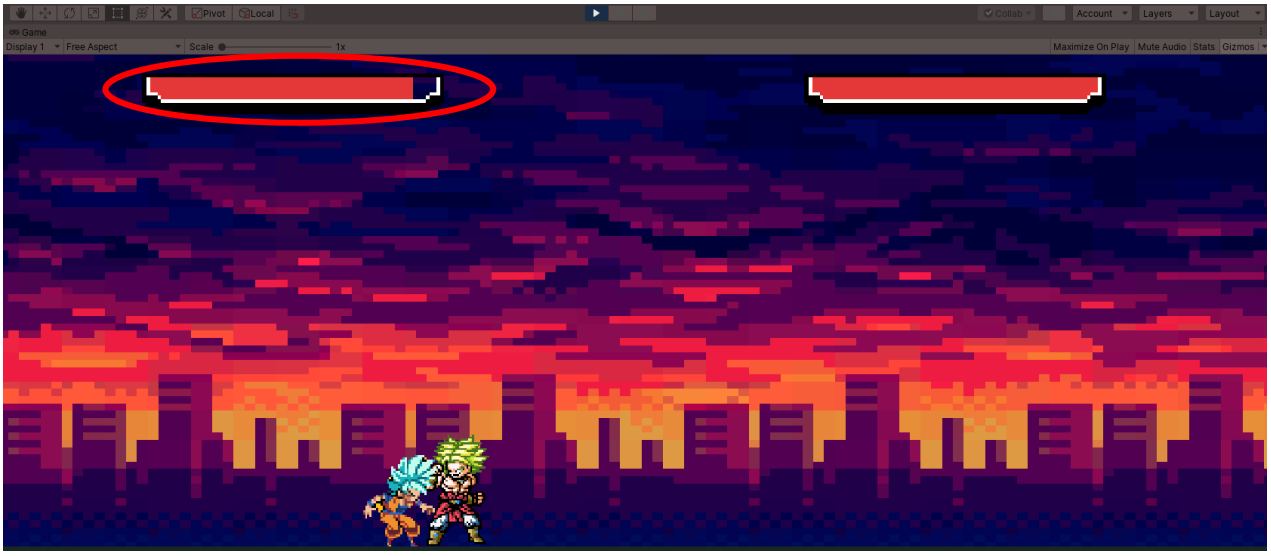
Broly Health



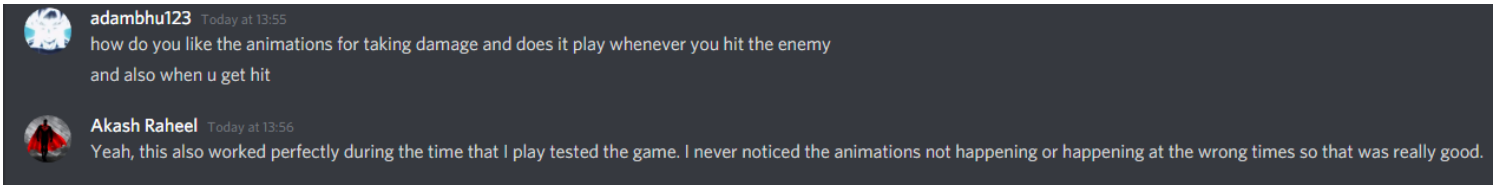


Goku Health


Pivot
Local
Account
Layers
Layout
Game
Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos


Pivot
Local
Account
Layers
Layout
Game
Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos

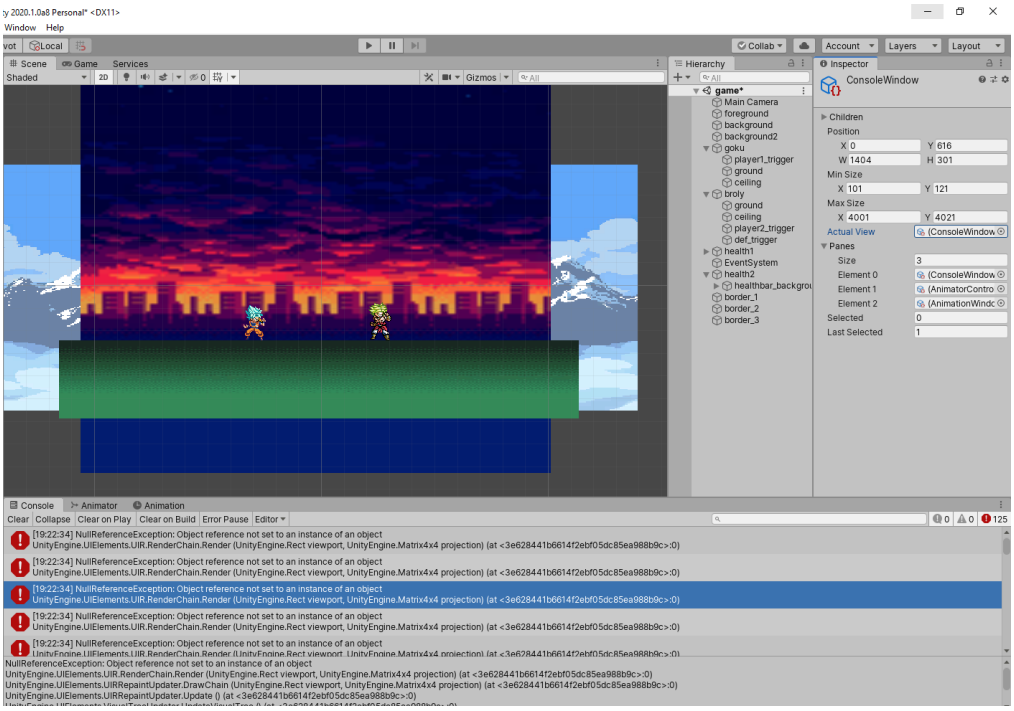
User feedback



The animations for taking damage played accordingly to the users actions and also didn't play at any other situations.

Error

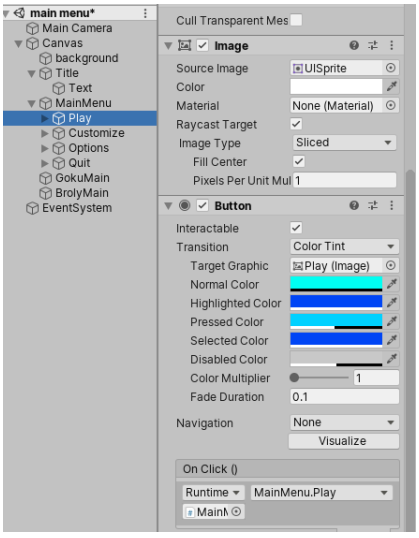
This error below stopped me from editing the animators for the characters. I tried to find a problem with my program but couldn't find any. So I just re imported everything again and it worked unity started working again.



Main menu

The main menu I made has 4 buttons, as of now only play and quit are functioning throughout the development I will try to implement the options and customize button.

```
public class MainMenu : MonoBehaviour
{
    public void Play()
    {
        SceneManager.LoadScene("Player1Select");
    }
    public void Options()
    {
        SceneManager.LoadScene("Controls");
    }

    public void QuitGame ()
    {
        Application.Quit();
        Debug.Log("Quit");
    }
}
```



This script will enable the player to load the player selection screen that will also lead then lead to player 2 selection screen. After selecting the characters, they will get to select what map they want to play in. This script will also load the controls option which is in development. For now, the final button they can press is quit which just closes the game.

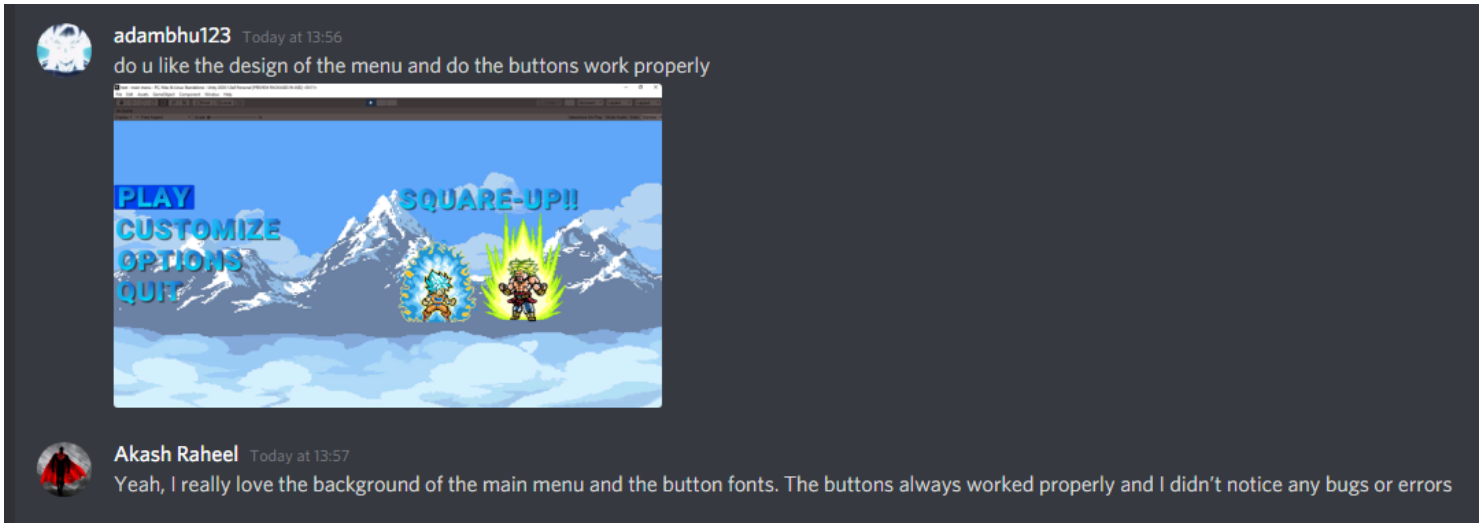
The button for each of the options the user can click will have the same elements as the one above. I've made it so that whenever the user hovers over a button it will be hight lighted in a darker blue and when pressed it change to a lighter blue. Using the button component has also allowed me to choose what colour they will be and how they should behave to user actions.

Options will display them with the controls the can change and quit button closes the game.

Testing menu buttons

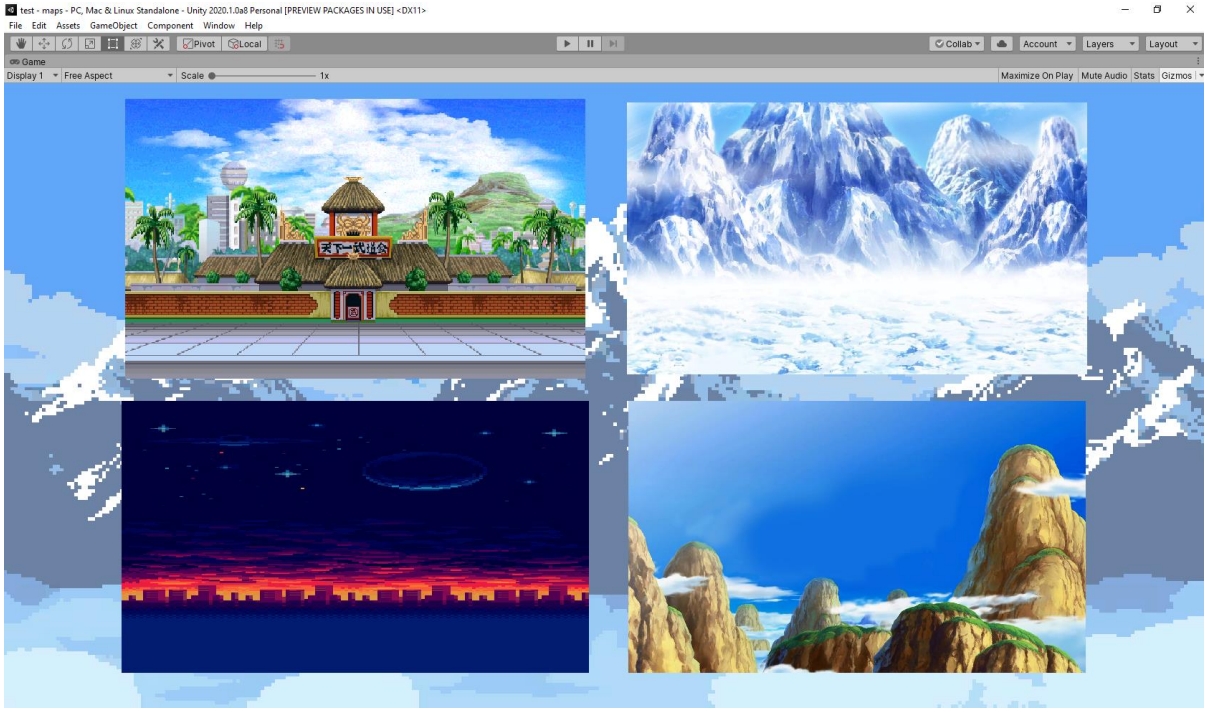
1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	All buttons are reacting to the mouse hovering over them and loads the next scene when clicked.	None

User feedback

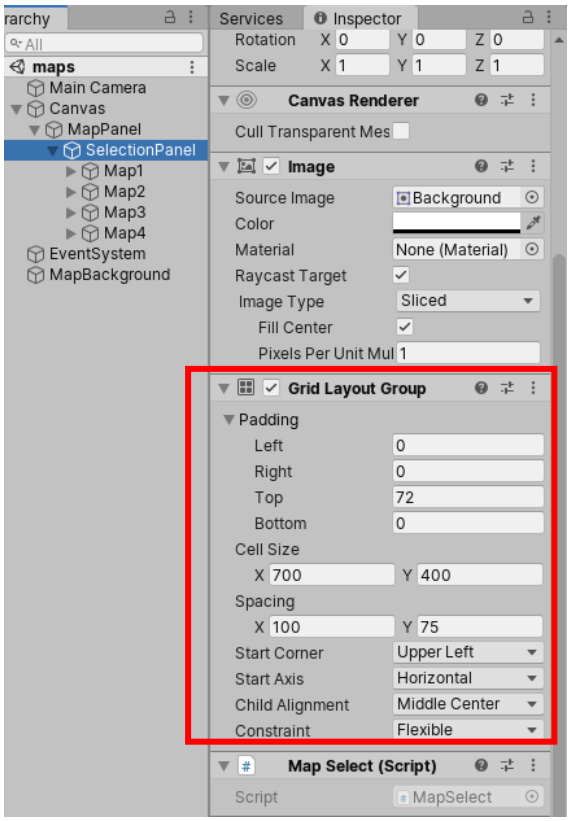


The user is able to use the menu with ease and doesn't have any issue with the design of the menu.

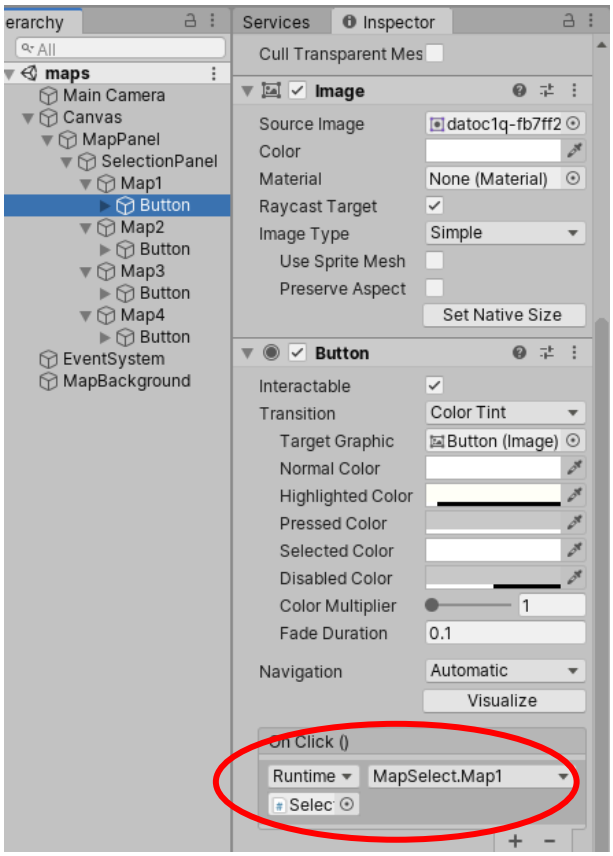
Maps



Below is the grid layout I used to make the four rectangles which I then inserted the map images to, I sized them to my liking by assigning what the values are for the cell size and the spacing between the images.



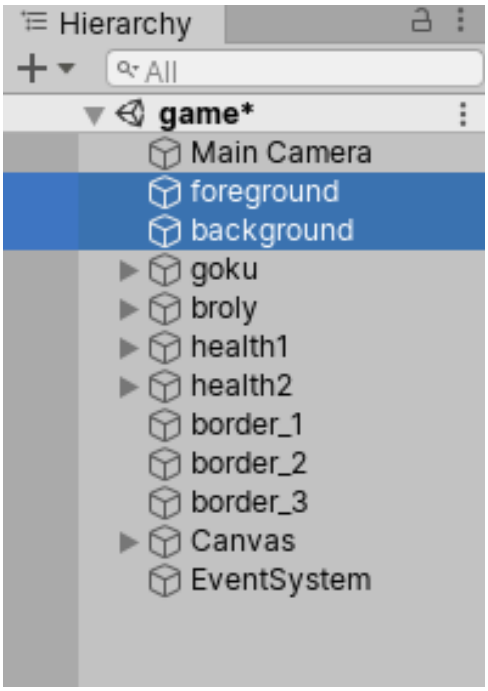
Then I assigned a button to each one that will give off a highlighted colour when hovered over. They also have the map select script in their “on click event”(circled red) so when clicked they will transition the screen to the map the user wants to play in. This process is used for each of the buttons.

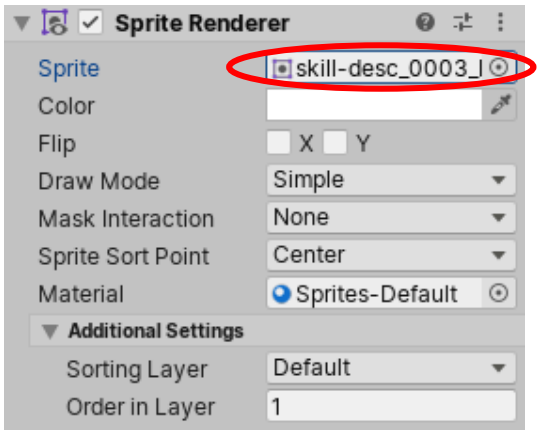


Below is the script that has the 4 functions that I made to load scenes depending on what map is being selected.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MapSelect : MonoBehaviour
{
    public void Map1()
    {

        SceneManager.LoadScene("WorldTournament");
    }
    public void Map2()
    {

        SceneManager.LoadScene("IceMap");
    }
    public void Map3()
    {

        SceneManager.LoadScene("TestScene");
    }
    public void Map4()
    {

        SceneManager.LoadScene("skyMap");
    }
}

```

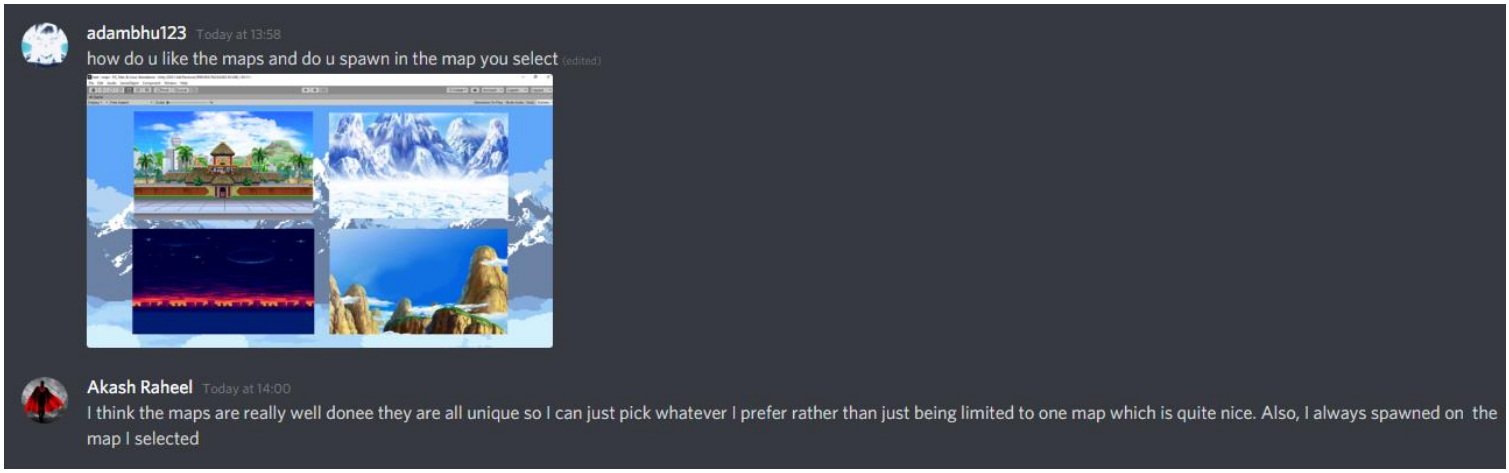
This script loads the map that the player clicks on. On each map I just copied elements from the original scene that had the foreground, background and borders and used them in each of the maps. The map select script is also used as a reference on the “on click” event for every button.





After making the scenes and referencing the code to each of the map buttons, I copied the components in the hierarchy of the first scene that I used to test the characters fight system and pasted them to the other scenes. To change the map background and foreground I went to the individual scenes and changed their sprite renderer to match the map the user will select in the map options. I also readjusted the colliders to fit the measurements of the new sprites. This way it saves time remaking the components for each of the characters and the box colliders.

User feedback

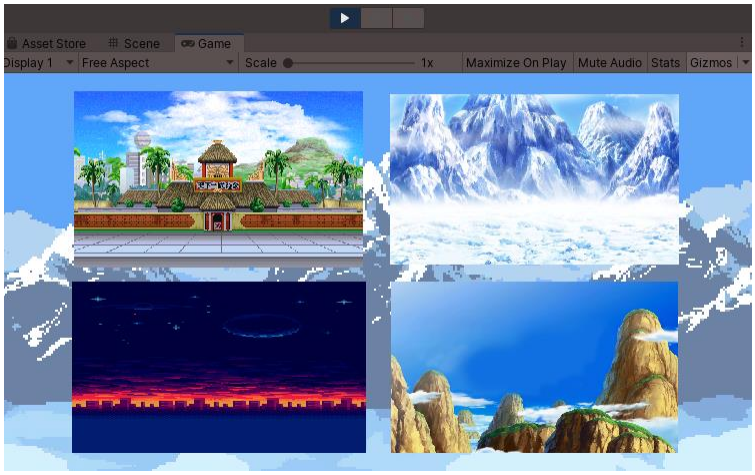


The user likes having several maps to play in and is also being spawned into the map they select; this mean I don't have to worry about the design of the maps and their function

Testing buttons for maps

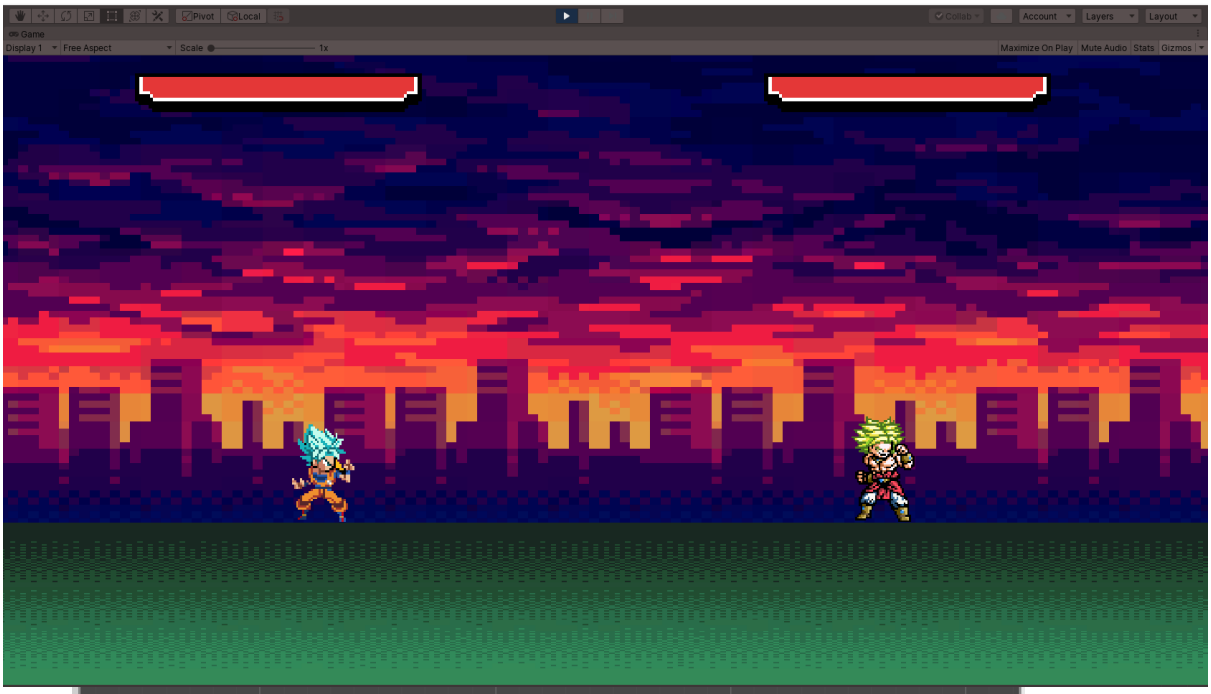
9	Maps	The maps options will allow the user to select what background they want to play in	Use mouse and buttons	**Good:** Wanted map is displayed in game **Bad:** map not appearing in game **Boundary:** N/A	User should be able to click on what map they want to play in and be transitioned to the game in the map they selected	The buttons for each of the map shows there is an interaction with the mouse and when clicked the scene transitions to the map that was selected	none

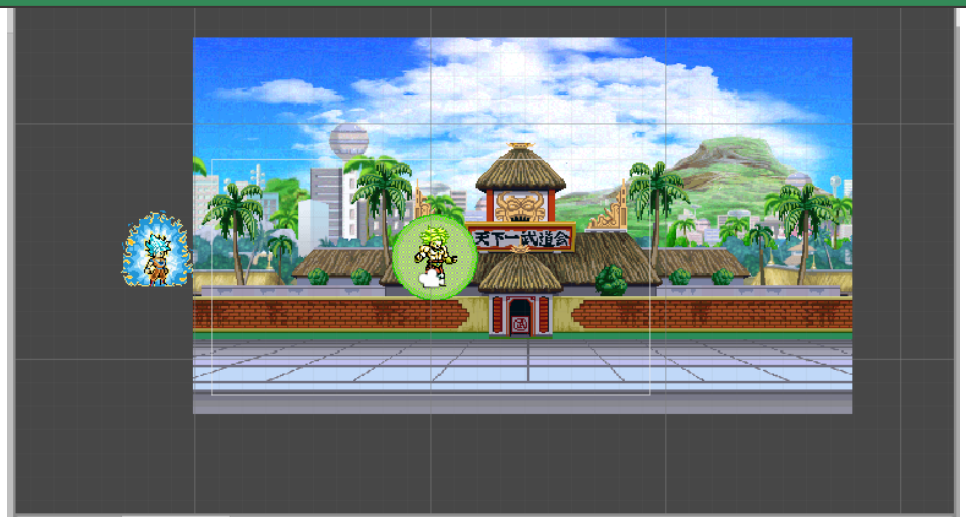
After hovering on top of the bottom left image the size of it shrinks and in play mode I can see that the user will be able to tell there is a clear interaction with the mouse. On the right image below there is a faint outline around the bottom left map. In game It is much clearer to see the interaction with the mouse.



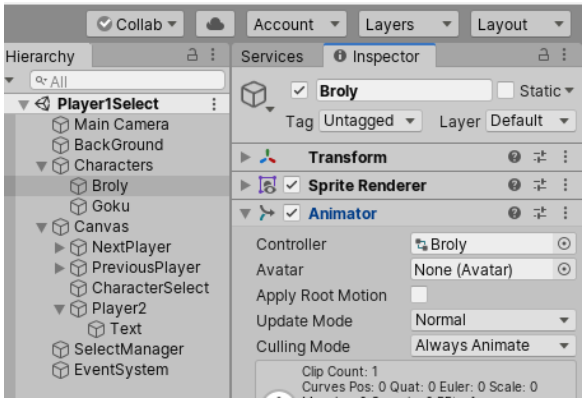
After clicking on the image the game successfully transitioned to the map and displayed the background, foreground, characters, health bars and their functions also work.

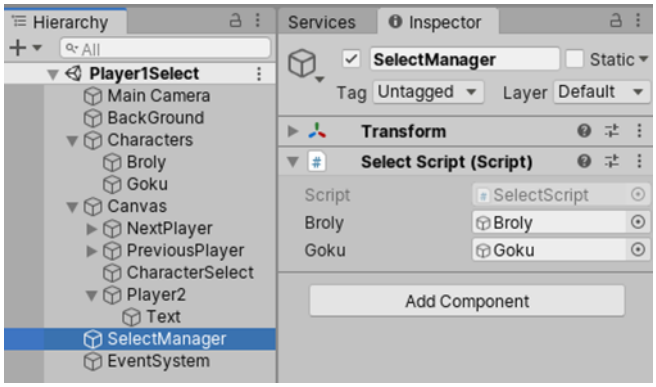
Player selection



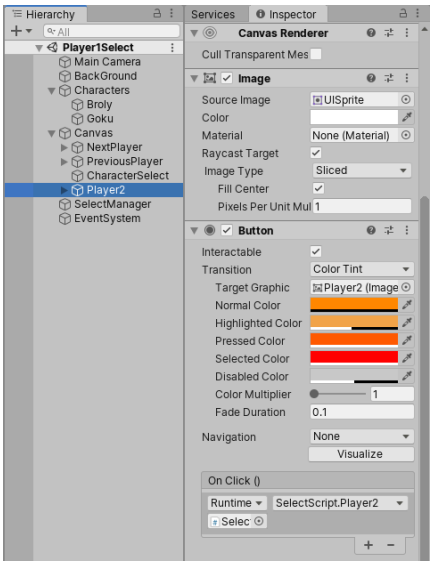


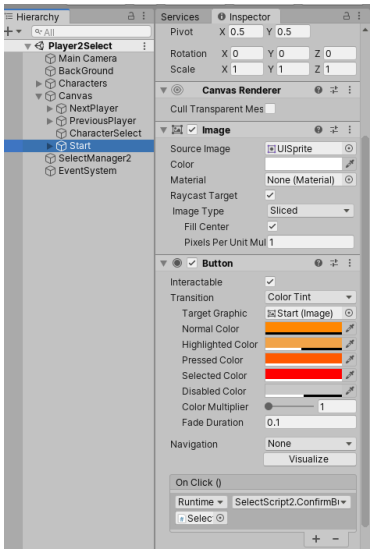
I have placed Broly inside the map and Goku outside so they swap positons whenever the player clicks on “next player” or “previous player”. By doing this it makes it easier to display what character they want and also select which they want to play with in the game. The first scene is for player 1 to choose what character they want. Then the player 2 button on the bottom right will change the scene to the player 2 selection screen.

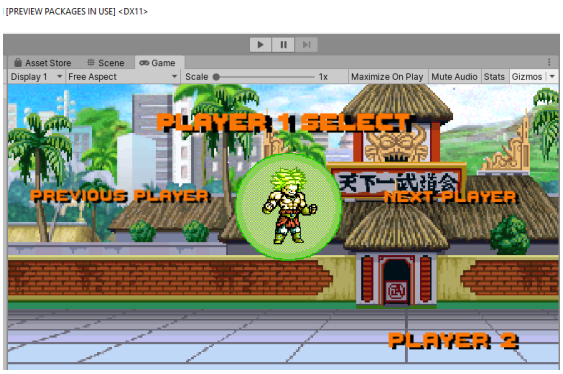




In the hierarchy I have put the sprites used and applied animations to them when the scene loads up. The canvas contains the title of the scene and contains the buttons which have the select manager as reference to change scenes when buttons are clicked. Below you can see the “player 2” and “start” button have the select manager script as reference for what characters will player 1 and 2 be when loading the map and change scenes. The select manager contains the sprites used to swap Goku and Broly in and out of the map.







On the left is scene for player 1 selection and then transitions to the right scene for player 2 selection. The transitions happen when the player clicks on the “PLAYER 2” button, then the “START” button on the right will got to the maps selection screen.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class SelectScript : MonoBehaviour
{
    public GameObject Broly;
    public GameObject Goku;
    private Vector3 CharPos;
    private Vector3 offScreen;
    private int CharInt = 1;
    private SpriteRenderer GokuRender, BrolyRender;

    private readonly string selectChar = "selectedCharacter";

    private void Awake()
    {

        CharPos = Broly.transform.position;
        offScreen = Goku.transform.position;
        GokuRender = Goku.GetComponent<SpriteRenderer>();
        BrolyRender = Broly.GetComponent<SpriteRenderer>();
    }

    public void NextChar()
    {
        switch (CharInt)
        {
            case 1:
                PlayerPrefs.SetInt(selectChar,1);
                GokuRender.enabled = false;
                Goku.transform.position = offScreen;
                Broly.transform.position = CharPos;
                BrolyRender.enabled = true;
                CharInt++;
                break;
            case 2:
                PlayerPrefs.SetInt(selectChar, 2);
                BrolyRender.enabled = false;
                Broly.transform.position = offScreen;
                Goku.transform.position = CharPos;
                GokuRender.enabled = true;
                CharInt++;
                ResetInt();
                break;
            default:
                ResetInt();
                break;
        }
```

This script takes the game object of Broly and Goku and sets a position inside the camera view and one outside.

The SpriteRenderer will use the characters sprite as reference for the positions that have been set. The switch function is used to determine what to do in certain cases that will require the sprites to switch positions.

In case 1 Goku sprite is disabled and its position is set to off screen, but Broly’s position is inside the scene. It also enables the Broly sprite. At the end of case 1 it will increase CharInt by 1.

The Reason for incrementing CharInt is because when “next player” button is clicked it will then go to case 2 which we need so that we can select the other character if we want to play as them.

In case 2 Broly sprite is disabled and its position is set to off screen. Now Goku will swap positions with Broly and be inside the scene and Goku’s sprite also becomes true. At the end of case 1 it will increase CharInt by 1 and resets the integer of NextChar.

The final case is the one where it is default which is when the user has not selected any character.

The public void NextChar is only for when the user is clicking the “next player” button.

```
public void PrevChar2()
{
    switch (CharInt2)
    {
        case 1:
            PlayerPrefs.SetInt(selectChar2, 1);
            Goku2Render.enabled = false;
            Goku2.transform.position = offScreen2;
            Broly2.transform.position = CharPos2;
            Broly2Render.enabled = true;
            ResetInt2();
            break;
        case 2:
            PlayerPrefs.SetInt(selectChar2, 2);
            Broly2Render.enabled = false;
            Broly2.transform.position = offScreen2;
            Goku2.transform.position = CharPos2;
            Goku2Render.enabled = true;
            CharInt2--;
            break;
        default:
            ResetInt2();
            break;
    }
}

private void ResetInt2()
{
    if (CharInt2 >= 2)
    {
        CharInt2 = 1;

    }
    else
    {
        CharInt2 = 2;
    }
}

public void ConfirmButton()
{

    SceneManager.LoadScene("maps");
}

public void Player2()
{

    SceneManager.LoadScene("Player2Select");
}
}
```

The public void PrevChar has the same purpose as the function above but only works for when the user click on the “previous player” button.

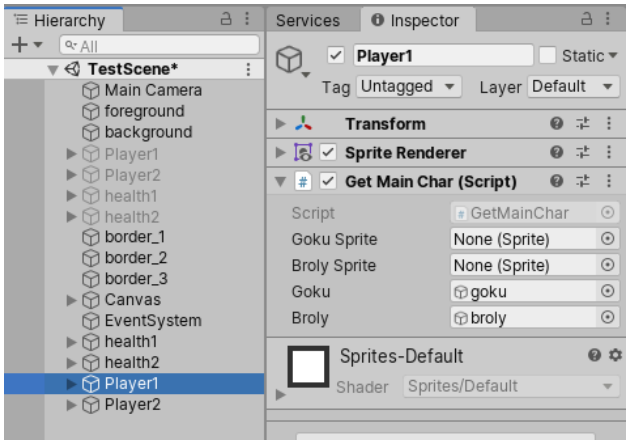
Below that function is the ResetInt function which checks if the value of Charint is bigger than or equal to 2, if so then CharInt is 1, else it stays 2. This is used so that whenever the we land on case 2 and we want to go back to case 1 we need to make the CharInt = 1 to pick whatever character was before or after.

If this function wasn’t used, then we won’t be able to use the buttons next player button or previous player button whenever we land on last character in the list.

Below the reset function is for the start button that will transition to the maps scene for the player to select their map. I have made to 2 copies of the same script so that both player 1 and player 2 can select their characters so the confirm button is used in player 2 selection, however the last function is used in player 1 selection screen to transition to player 2 screen.

Miscellaneous Files

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GetMainChar : MonoBehaviour
{
    public Sprite GokuSprite, BrolySprite;
    public GameObject Goku;
    public GameObject Broly;
    private SpriteRenderer mySprite;
    private readonly string selectChar = "selectedCharacter";
    void Awake()
    {
        mySprite = this.GetComponent<SpriteRenderer>();
    }

void Start()
    {
        int getChar;

        getChar = PlayerPrefs.GetInt(selectChar);

        switch (getChar)
        {
            case 1:
                mySprite.sprite = BrolySprite;
                Goku.SetActive(false);
                break;
            case 2:
                mySprite.sprite = GokuSprite;
                Broly.SetActive(false);
                break;
            default:
                mySprite.sprite = BrolySprite;
                Goku.SetActive(false);
                break;

        }
    }

}
```

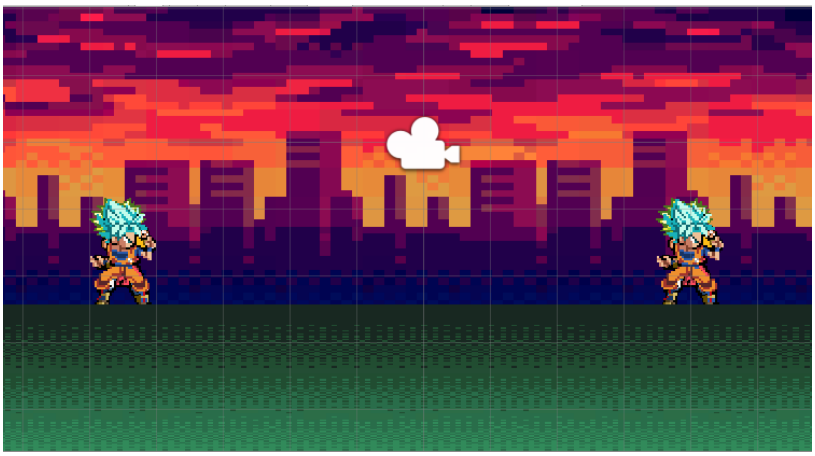


The script on the left has 2 versions, one for player 1 and player 2. Its purpose is to get the selected characters from player 1 and 2 selection screen and enable them in the map that the user chose to play in. SelectChar will determine what character was selected for player 1 and 2, depending on that getChar used by the switch function will enable and disable the sprites that match what the user has selected.

In case1 Broly is activated and Goku is set to false. Case 2 Goku is activated and Broly is false. The default case will be the same as case 1. This way the code will make sure that if the Broly sprite from player 1 and player 2 scenes was selected the Goku game object will deactivated since both of them will be in the same position. This also works the other way of Goku was selected which will make Broly false.

Below the script is the game objects I have referenced so that they can be activated and deactivated accordingly to what the user has selected.

The game objects that are in the Get Main Char script have all the scripts, animations, and child object that I have made from the beginning stages of developing the characters in the test scene. All I had to do was take health bars and player 1 and 2 components from the test scene and add them to all the other maps. This way the I can make it possible to select what characters I want to play as in any map.



In each of the map scenes I have overlapped 2 pairs of sprites for each character so that when player 1 and 2 choose what character they want to play as then the chosen one will be enabled and the other sprite will be disabled.

Testing buttons for character selection

Previous player button

1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	The previous player button is highlighted when hovered over and when clicked it emits a brighter orange and swaps to the Goku sprite	none

Next player button

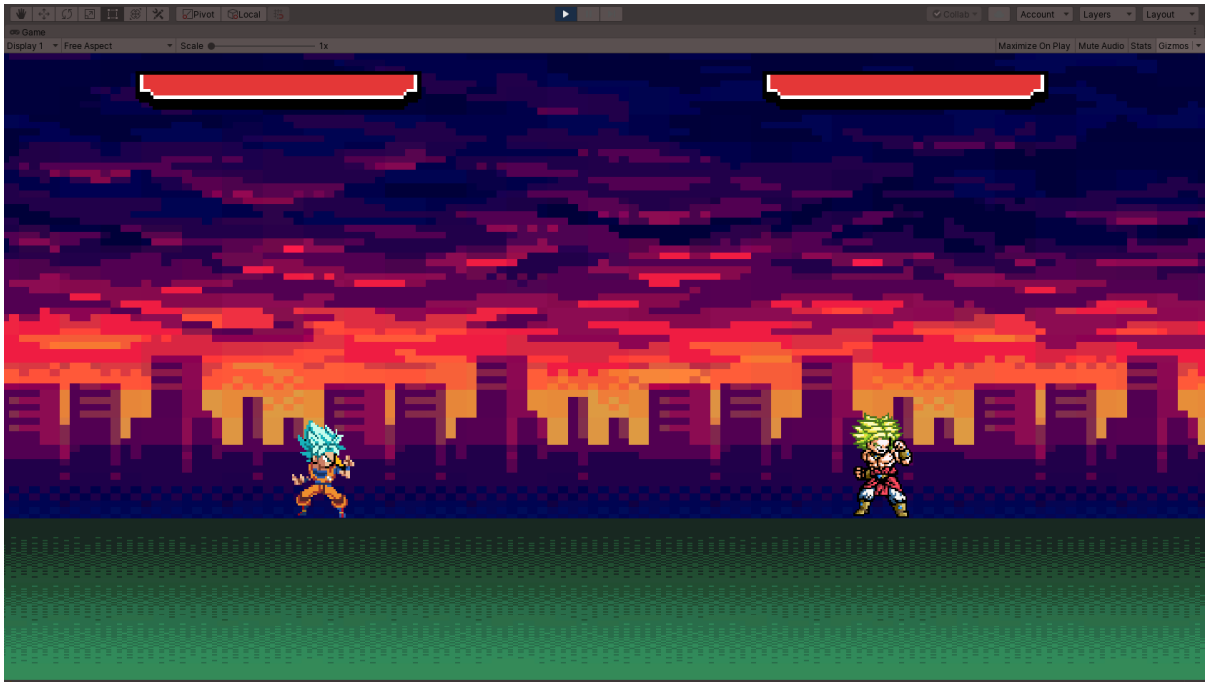
1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	The next player button is highlighted when hovered over and when clicked it emits a brighter orange and swaps to the Goku sprite	none

Player 2 button

1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	The player 2 button transitions to “player 2” scene and shows mouse interaction by emitting bright orange.	none

Start button

1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	The Start button transitions to the scene that I previously selected in the map scene. It also shows interaction with mouse by emitting bright orange	none

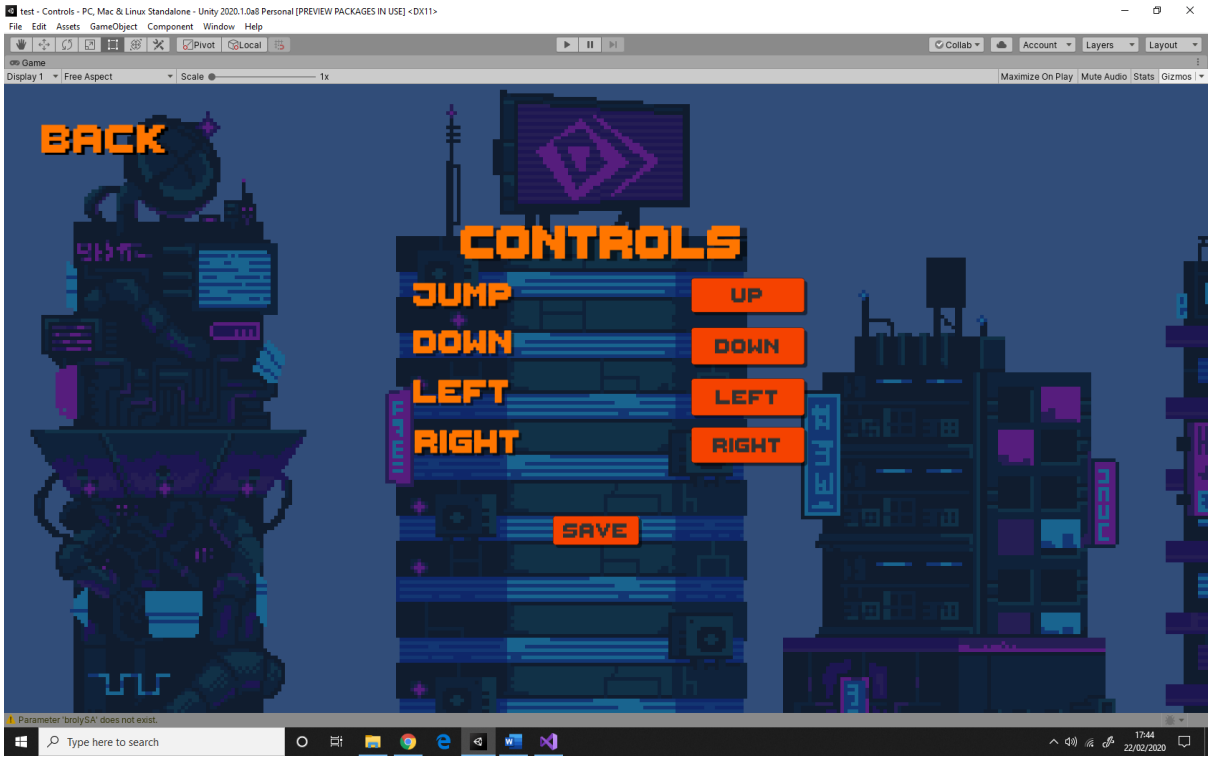


Here are the characters I chose from the player 1 and player 2 scenes which are shown in the map that I chose to play in from the maps menu.

User feedback

The client like the method of selecting the characters and doesn't have issues with the characters selected and spawns in the map with the selected characters.

Key binds



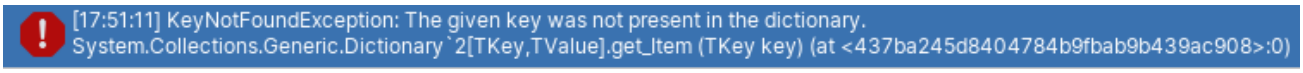
This scene is for the player to select what controls they want for movement, but I will be changing this for other inputs such as attacking and defending. The save button will enable the game to store what controls are assigned to what keys when playing the game.

```
public class keybinds : MonoBehaviour
{

    private Dictionary<string, KeyCode > keys = new Dictionary<string, KeyCode>();

   // private Dictionary<string, KeyCode, KeyCode> keys1 = new Dictionary<string, KeyCode, KeyCode>();

    //private IDictionary<string, Tuple<KeyCode, KeyCode>> keys1 = new Dictionary<string, Tuple<KeyCode, KeyCode>>();
    public Text up, left, down, right;

    private GameObject currentKey;

    // Start is called before the first frame update
    void Start()
    {
        keys.Add("Up", (KeyCode)System.Enum.Parse(typeof(KeyCode), PlayerPrefs.GetString("UP", "W")));
        keys.Add("Down", (KeyCode)System.Enum.Parse(typeof(KeyCode), PlayerPrefs.GetString("Down", "s")));
       // keys.Add("left", (KeyCode)System.Enum.Parse(typeof(KeyCode), PlayerPrefs.GetString("left", "a")));
       // keys.Add("right", (KeyCode)System.Enum.Parse(typeof(KeyCode), PlayerPrefs.GetString("right", "d")));
        //keys1.Add("Horizontal", (KeyCode)System.Enum.Parse(typeof(KeyCode), PlayerPrefs.GetString("a", "d")));
      //  keys.Add("Horizontal", (KeyCode)System.Enum.Parse(typeof(KeyCode), PlayerPrefs.GetString("Horizontal", "d")));
        //keys.Add("Horizontal1", (KeyCode)System.Enum.Parse(typeof(KeyCode), PlayerPrefs.GetString("Horizontal1", "d")));

        up.text = keys["UP"].ToString();
        down.text = keys["Down"].ToString();
        left.text = keys["Horizontal"].ToString();
       // right.text = keys["Horizontal1"].ToString();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(keys["UP"]))
        {
            Debug.Log("Up");
        }

        if (Input.GetKeyDown(keys["Down"]))
        {
            Debug.Log("Down");
        }


        if (Input.GetKeyDown(keys["Horizontal"]))
        {
            Debug.Log("left");
```

The script above is where I have a dictionary that takes the string(name) of the key code and the key in it to change it to a new dictionary with whatever key the user selects.

```
73
74      void OnGUI()
75      {
76          if(currentKey != null)
77          {
78              Event e = Event.current;
79              if (e.isKey)
80              {
81                  keys[currentKey.name] = e.keyCode;
82
83                  currentKey.transform.GetChild(0).GetComponent<Text>().text = e.keyCode.ToString();
84                  currentKey = null;
85              }
86          }
87      }
88
89      public void ChangeKey(GameObject clicked)
90      {
91          currentKey = clicked;
92      }
93
94      public void SaveKeys()
95      {
96          foreach (var key in keys )
97          {
98              PlayerPrefs.SetString(key.Key, key.Value.ToString());
99          }
100
101         PlayerPrefs.Save();
102     }
103
104     public void Back()
105     {
106         SceneManager.LoadScene("main menu");
107     }
108 }
109
```

I planned to make key binds for movement by making a dictionary however only the jump button can be changed, because I'm using horizontal movement that uses an axis not a single input this method doesn't allow me to assign keys to it. This may not be the case but I haven't found a way to do so.



The idea of being able to make key binds for player movement failed but this could instead be used for the attacks, and defence of the characters. This may actually be more relevant since movement in many pc games have the same W, A, S, D keys and the arrow keys when playing local 2 players, this may meet satisfy my client's needs more than trying to change movement.

Below is the test I did for the controls and stopping at save since that's where it stops working

Testing Controls scene buttons

Back button

1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	When “back” is clicked the screen transitions to the main menu	none

Back button **Transitions to ->** **Main menu**

Button interaction

1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	When the mouse hovers over the “UP” button it changes colour to a dull green then turns into bright green when clicked	None

Testing key binds for each button

1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	When each button Is clicked individually then a key is entered the button changes it’s value into the key entered	None

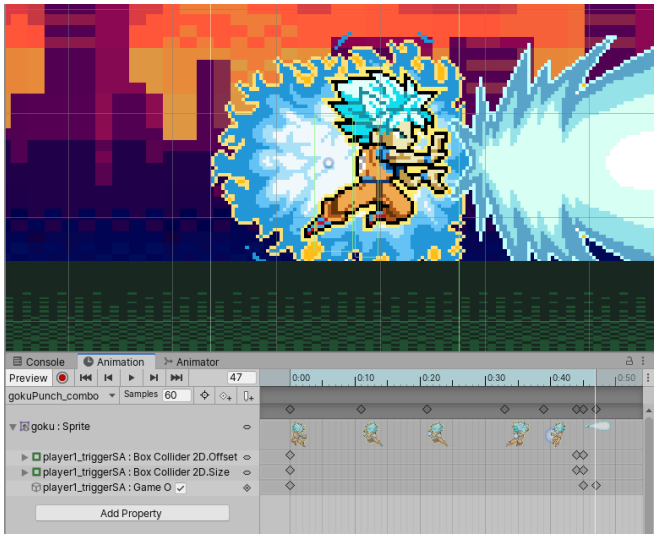
Below you can see that all values in the buttons are changed

Save button

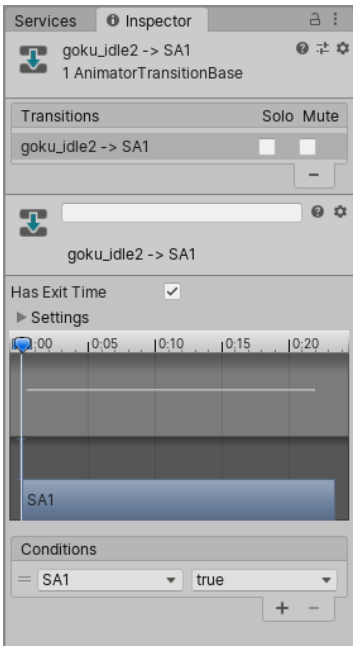
1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	Save button shows interaction with mouse	The save button is supposed to keep the changes made to each of the buttons by the user

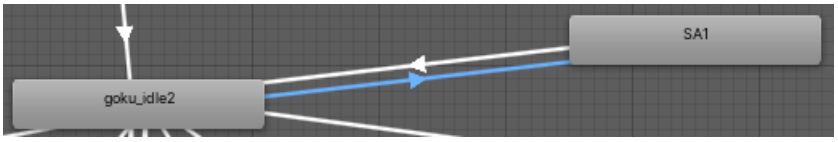
The right image is after coming back from the main menu and it shows that the keys weren’t saved

Super attacks



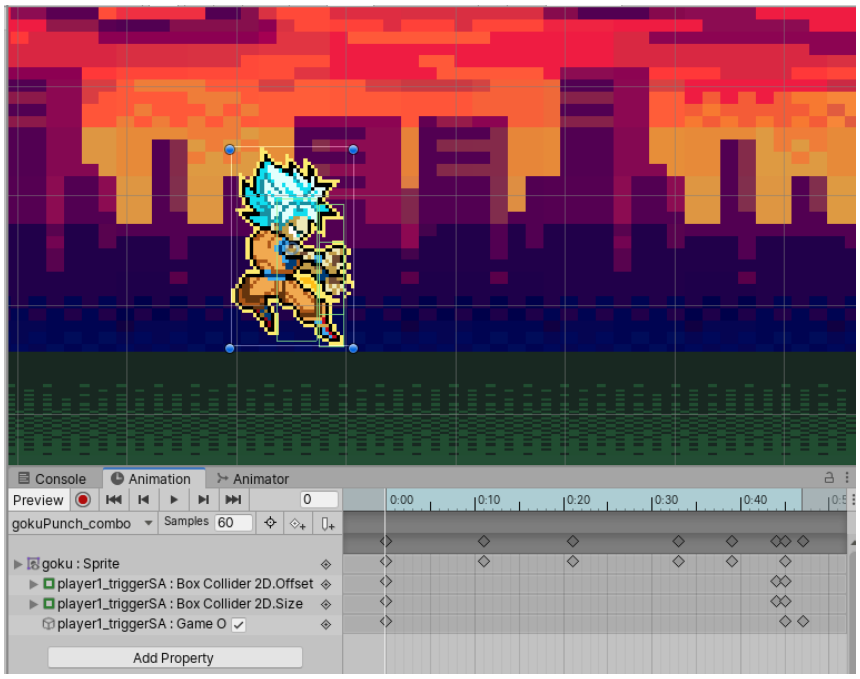
The super attack has 2 transitions, 1st is when its activated through the parameter “SA1” which ahs to be true. 2nd transition is when “SA1” is false which will bring the character back to idle.



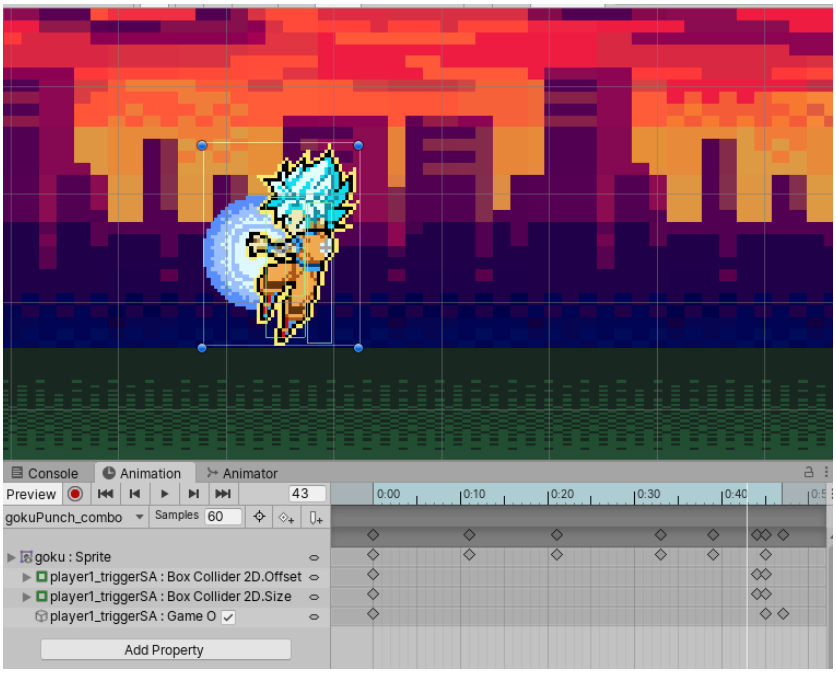


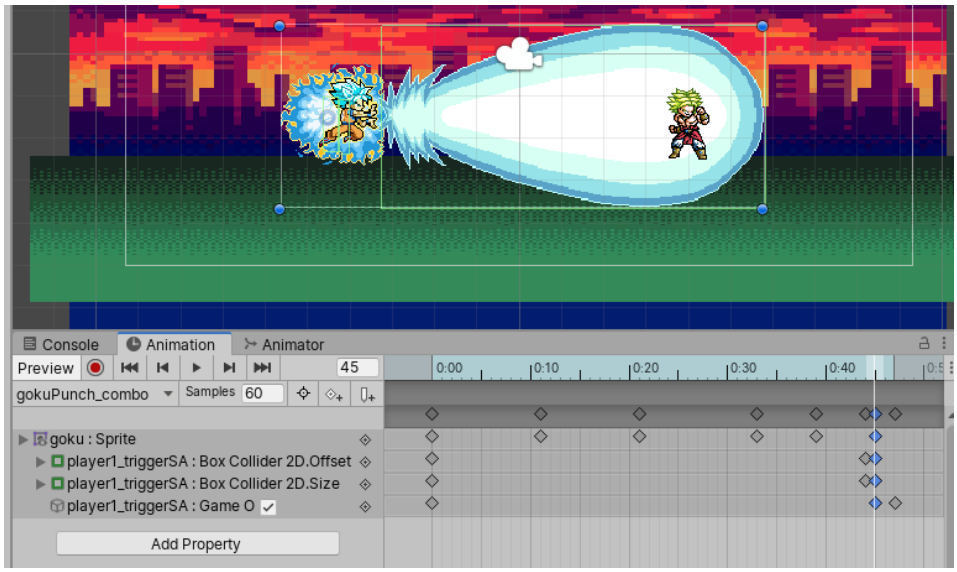

```
if (Input.GetKeyDown("1") && !SA)
    {
        SA = true;
        Debug.Log("sa");
        atkTimer = atkCd;
         // this checks if the player enters the button punch button which is i, if so then punch becomes true
         StartCoroutine(SAtrigger());

        // the trigger then also becomes true

        SAtotal += 1;
    }

IEnumerator SAtrigger()
    {
        yield return new WaitForSeconds(0.45f);
        player1_triggerSA.enabled = true;
    }

  if (SAtotal == 1)
    {
        SAtimer = SAcd;
        SAtotal = 0;
    }

  if (SAtimer > 0)
    {
        SAtimer -= Time.deltaTime;
        SA = false;
        player1_triggerSA.enabled = false;
        SAtotal = 0;
    }

 /*  else
    {
        SA = true;
        player1_triggerSA.enabled = true;
    }*/

    if (SA)
    {
        if (atkTimer > 0)
        {
            atkTimer -= Time.deltaTime;// if the atk timer is bigger then 0 then this line will decrease the time by delta time which works as a real timer
        }
        else
        {
            SA = false;
            player1_triggerSA.enabled = false;
            // if player is not using the super attack button then SA becomes false and the trigger will also be false
        }
    }

    animator.SetBool("SA1", SA);
    // this will set the punch animation in the animator of player 1 to true or false depending on the two if statements above
```

If the player presses the input for a super attack, the bool SA will be true and the co routine SAtrigger will initialise. The SAtrigger will wait for 0.45 seconds which is amount of time for the super attack to reach its peak point, then it will trigger the super attack trigger which I named “player_triggerSA.enabled”. This was made so that the box collider doesn’t instantly activate before the super attack reaches the other player. After the super attack is activated the script will make SA false and the super attack trigger also false. The last line will activate the animation whenever the super attack is activated.

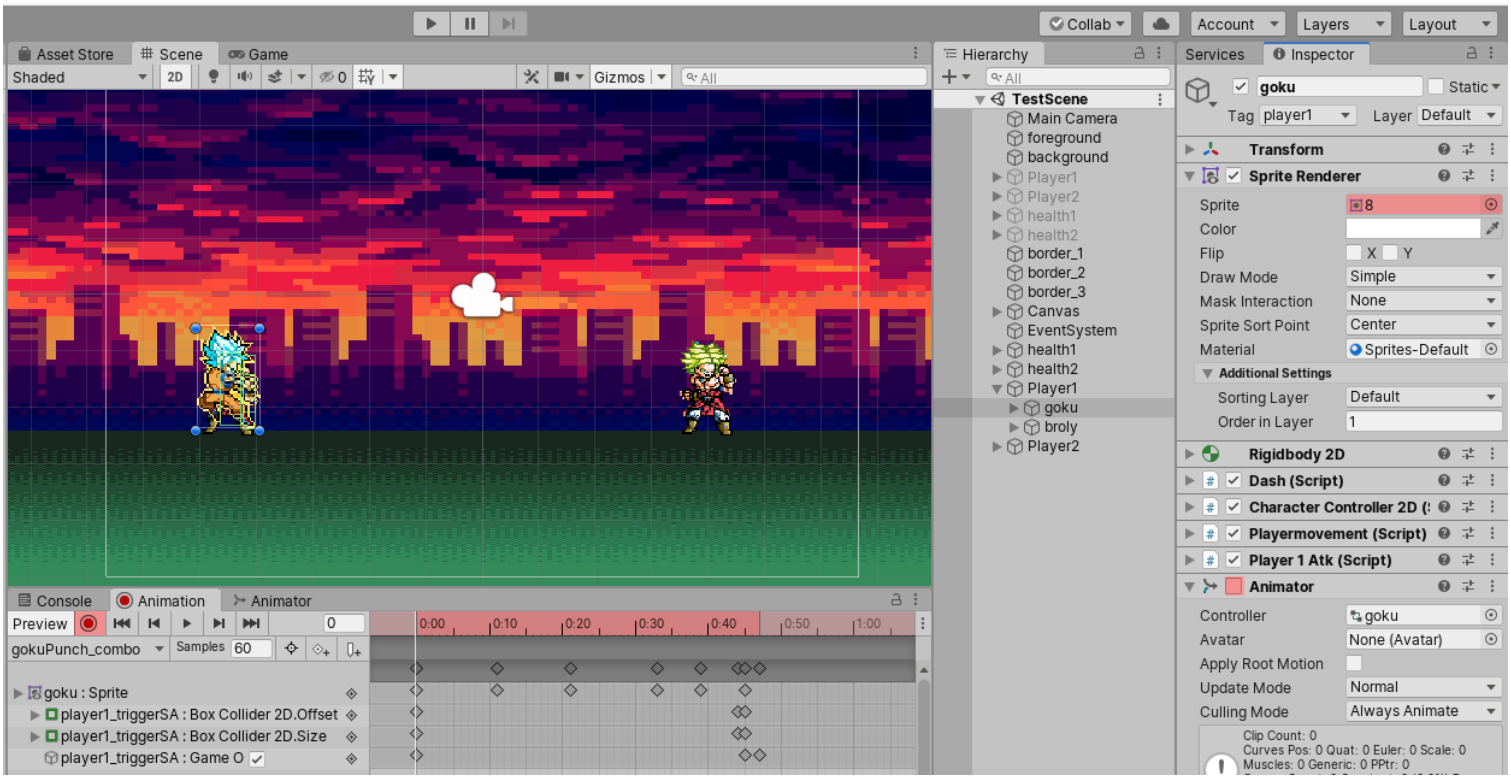


This is the start up frame for the animation where I recorded the box collider to stay inside the character until the frame below. This is because I want the collider to only expand when the super attack reaches the enemy. The frame below is the last frame where the collider is kept inside.

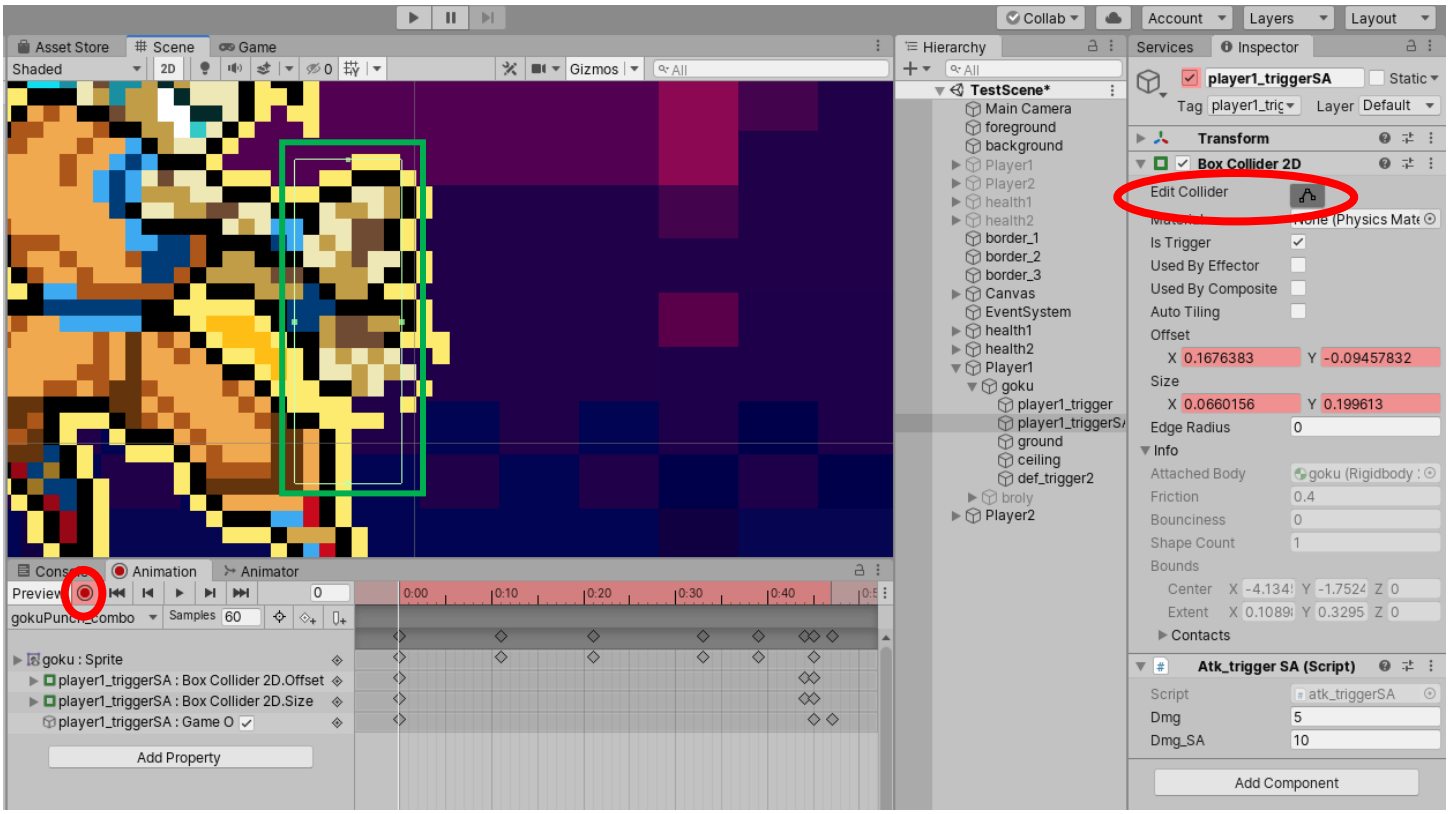




In the frame above is where I made the box collider to fully expand, this is is linked with the coroutine I made which waits for 0.45 seconds before triggering the super attack collider.



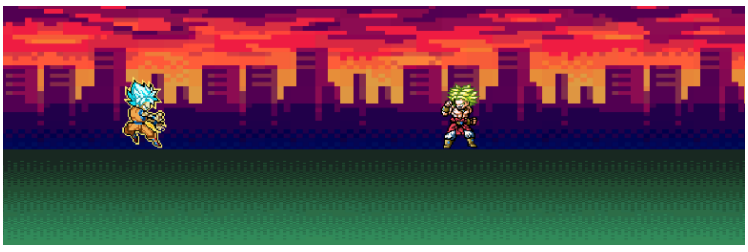
I was able to record how the box collider will change throught out the animation by clicking the record button. When clicked you can also see that the sprite renderer and animator are emiting a red light. This is because its part of the animtion of what sprite is being shown when I drag the pointer in the animation tab and the animator is the component playing the animation. While recording I selected “Edit collider” of the super attack trigger which enabeld me to change it to my liking of where it should be positioned in each frame. The screen shot of the box collider is below which surrounded by the green rectangle.

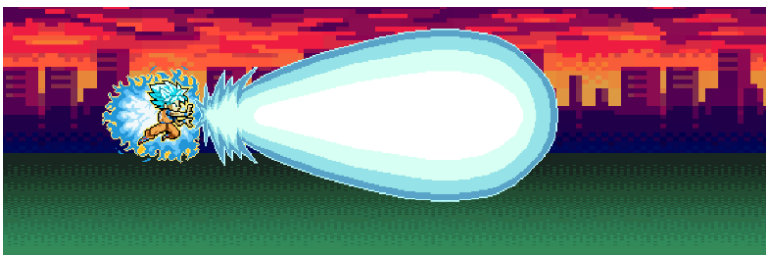


Testing Goku super attack

Animation

3	Animations	Test if animations play if conditions are met	I,O,P	Good: “I” Bad: “G” Boundary: N/A	See if the animations for specific conditions are displayed	The animation for the super attack was played when entered	none

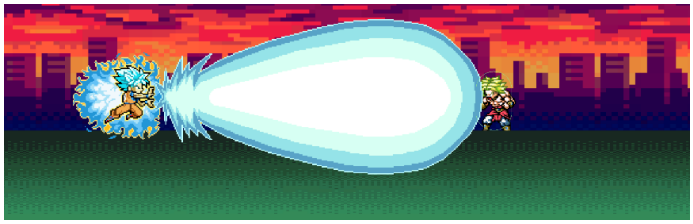


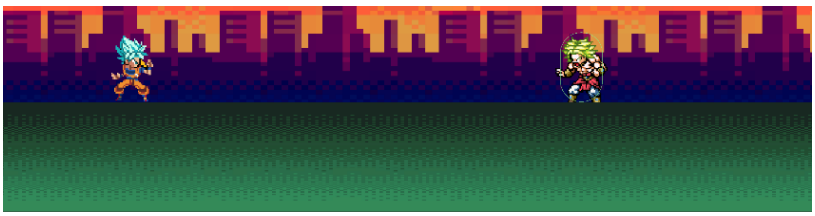


Testing colliders of super attack

2	Fight system/ box colliders	Use inputs assigned to each character move and check if box colliders are being activated	I,O,P,J,K,L	Good: “I” Bad: “v” Boundary: n/a	See if the attacks are activating /colliding with other character’s box colliders	When super attacked was entered the box collider around the blast showed that it hit Broly since he played the animation for taking damage.	none

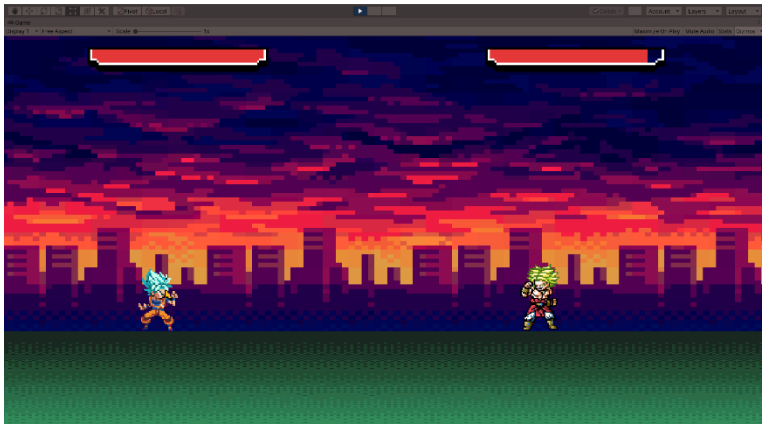
Below the animation for taking damage plays until the super attack has finished

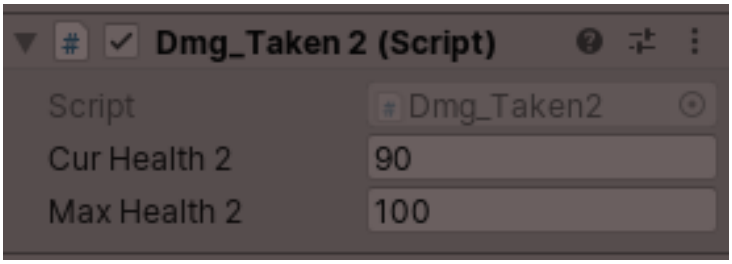




Health bars

13	Health system	Check if the user and enemy health is decreased after taking damage	Use attacks to see if enemy is taking damage and check user is taking damage by letting enemy attack	Good: player1 health decreases after attack Bad: no damage dealt Boundary: health staying the same after being attacked in defending state	The health of both the opponents should decrease by the amount the attacker is applying to them. Example kick should do 5 damage and super should do 10	After entering the super attack Broly’s health went down by 10 in the health bar and the code	none





As you can see below both the health bar image and the public value of Broly's health in the script decreased by 10.

User feedback

The user likes the animation of the super attack and has confirmed that it functions properly whenever its used

Pause menu

```
public class PauseMenu : MonoBehaviour
{
    public static bool Paused = false;

    public GameObject pauseMenuUI;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (Paused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        Paused = false;
    }

    void Pause()
    {
        pauseMenuUI.SetActive(true);
        Time.timeScale = 0f;
        Paused = true;
    }

    public void Menu()
    {
        SceneManager.LoadScene("main menu");
    }

    public void Quit()
    {
        Debug.Log("quit");
        Application.Quit();
    }
}
```

This script will enable the user to pause their game.

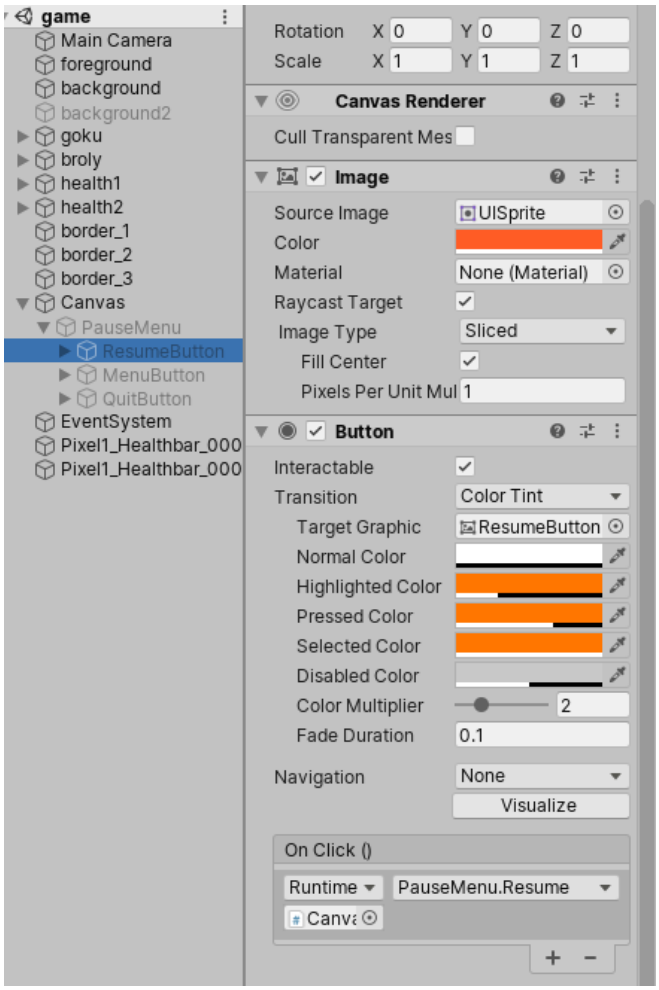
I have made the bool pause false since we don't want the game to be paused when the game is loaded. The game object pauseMenuUI will be used as reference so whenever the Esc button is pressed the user can use it as an interface to access the features in the pause menu.

When the escape button is pressed the game will pause if the pause is false or it resumes if the pause is true.

The resume function will make the pauseMenuUI false and paused false. This is used in void Update to trigger the resume button on the interface, time is set to 1f which multiplies time by 1 because we want the game to play normally when resumed. This is also the same case for the pause function but this time it triggers the pause menu to pop up and set variables back to true, also in this function time is multiplied by 0 so that everything will freeze and disable the user to move or attack with their characters.

The menu function will be made available when the game is paused so when the "main menu" button is pressed the scene manager will transition from the game to the main menu scene. The quit button will just close the game application.

In the hierarchy the pause menu UI is disabled since we don't want until the user presses escape. The child objects of the Pause Menu will have a On Click event that will use the Pause Menu script that's in the "Pause Menu" game object as a reference. This will allow each of those buttons to perform their task. I have also edited what the button does when it is clicked or highlighted.



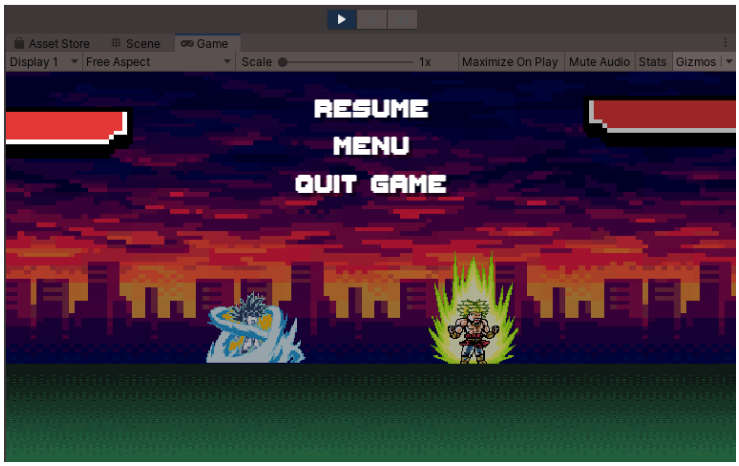
Testing pause menu

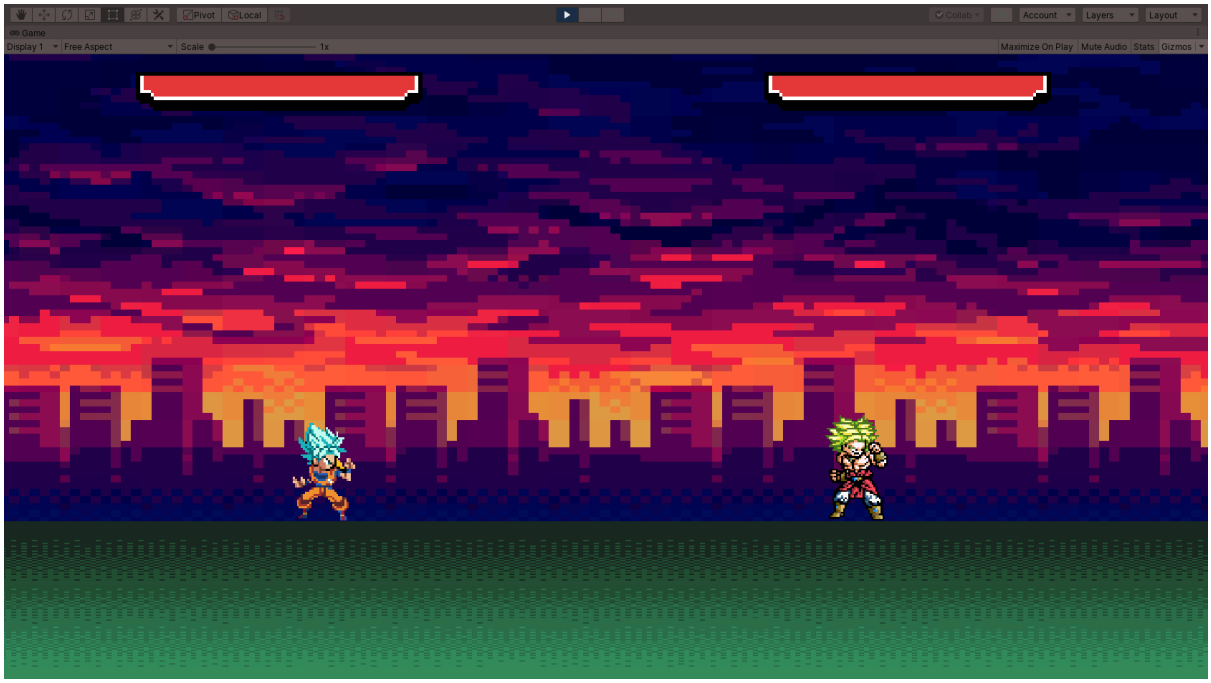
Above is the game when it is paused and the pause menu UI is activated, as you can see the animations of the characters are also paused which is what you want when the game is paused. The buttons are also highlighted and perform their task when clicked on.

Testing pause menu

Resume button

1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	When “Resume” button is clicked it emits bright orange and transitions back to the game	





Quit button

1	Buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: “q” button Boundary: n/a	Buttons should allow the user to access all the features and modes the game has	When “Quit” button is clicked the console outputs quit to make sure that the button is working	none

Evaluation

Stakeholders: black box test testing

Questions:

How is the character movement?

How is the jumping of the characters?

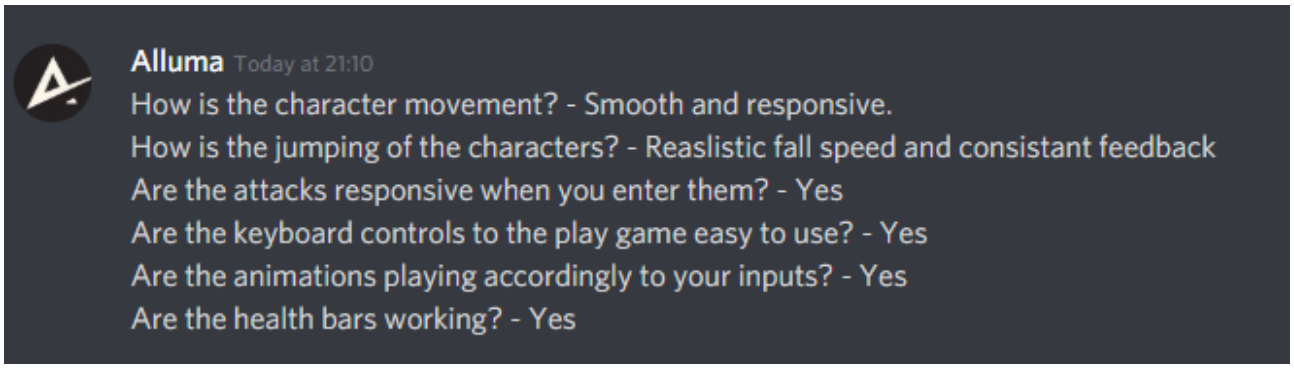
Are the attacks responsive when you enter them?

Are the keyboard controls to the play game easy to use?

Are the animations playing accordingly to your inputs?

Are the health bars working?

User: Luca Milazzo



User: Brandon Jackson

How is the character movement? “The character movement as of now is fluid and smooth”

How is the jumping of the characters? “The jumping is a little too high”

Are the attacks responsive when you enter them? “The attacks are very responsive when inputted”

Are the keyboard controls to the play game easy to use? The keyboard controls are easy to pick up and use”

Are the animations playing accordingly to your inputs? “The animations will sometimes not play according to my inputs, instead being a singular frame”

Are the health bars working? “The health bars are working”

Usability testing

Navigation:

For the menu the user will be able to navigate through the game using the mouse. Currently they can use the play button to load the game with the 2 characters for multiplayer between them and their friend, before entering they can also select what map they want to play in. The buttons are also highlighted when hovered over them to show that the button are working.

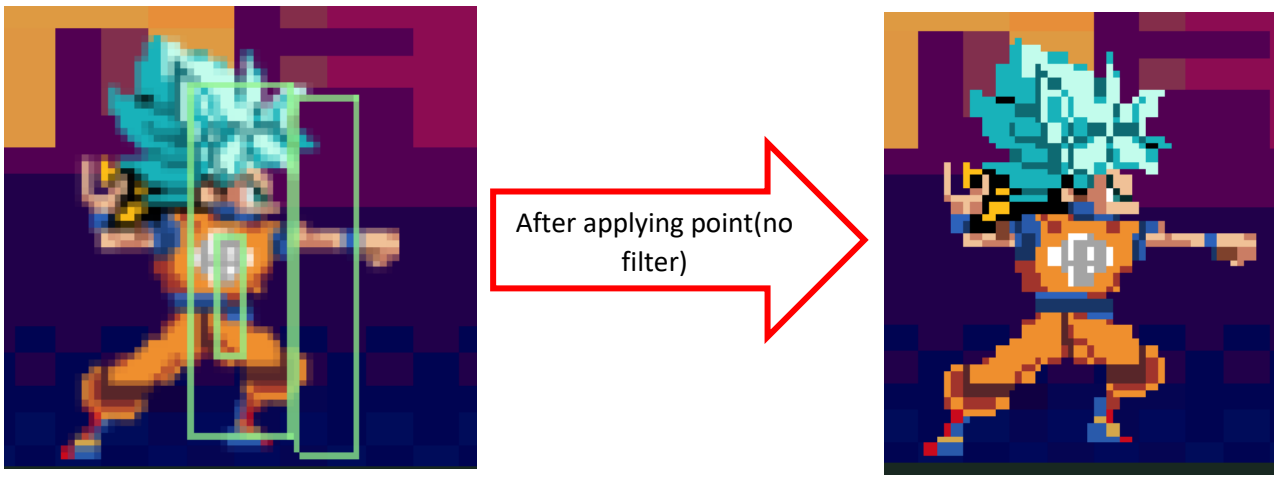
As you can see the play button emits a dark blue colour when hovered above, this is also the case for all the other button in the menu

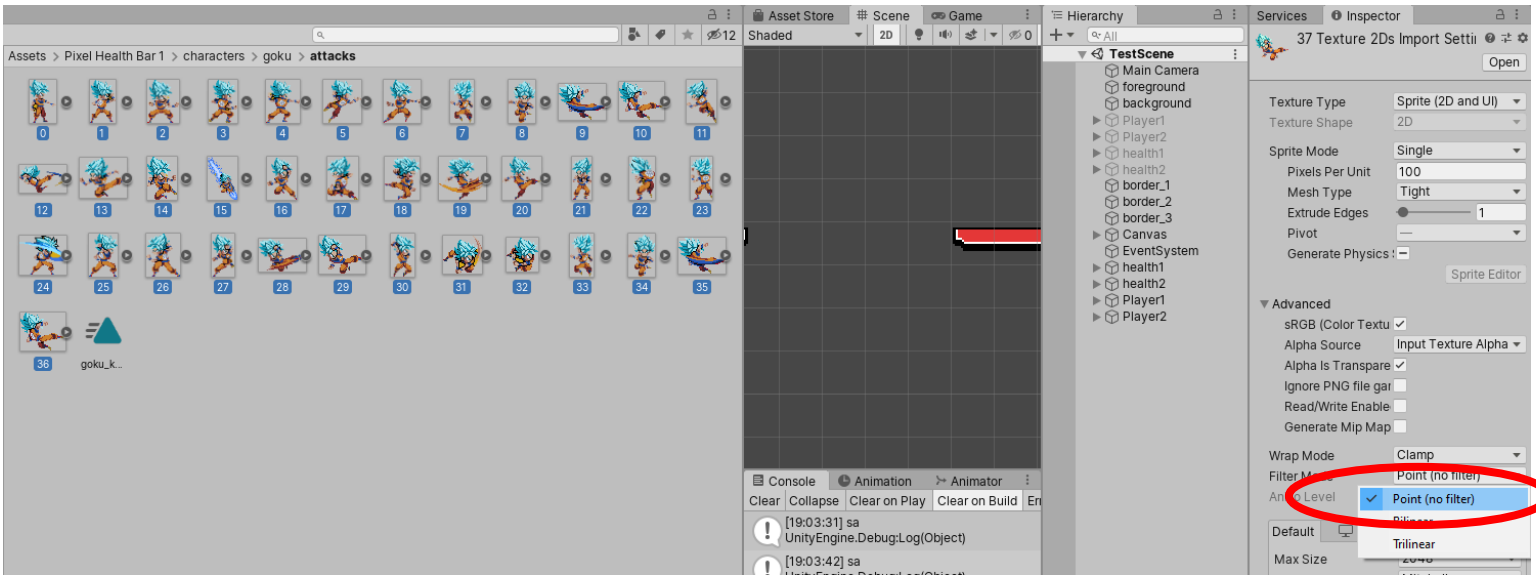
Consistency:

An Issue that my users had to deal with is the heath bar not working at some times. For example the players will take damage in one game and after someone wins and restart ,one of them will not take damage. I will need to go back to development and fix this issue. A feature of the game that is consistent is that the esc button always works to display the pause menu and freeze the game. It allows the user to go back to the menu or resume to the game they were playing when the buttons “Menu” or “Resume” are pressed. The “Quit” button also works in the pause menu and main menu.

Visual clarity:

The sprites I originally imported into the game were blurry so I made them point (no filter) to make them clearer, the other game objects are also visible for the player such as the foreground, background and health bars. They users are able to read the text and like the text theme for the retro based game. They also like the design of the characters.





Efficiency:

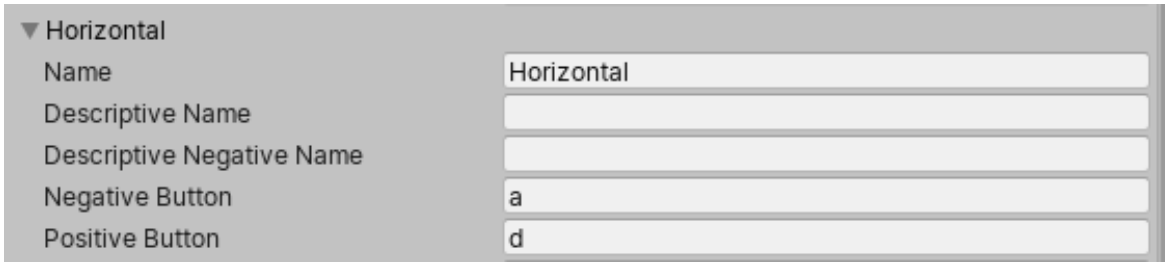
I have tried to make the scripts I have written use the least amount of lines needed for to make the functions of the game to work. I have deleted some lines of code that aren't relevant for the purpose that the script is in. Since my game use pixel art and is 2d it will have low impact on how pc's run the game. In the 2 screenshots below I've reused the same script but changed the variables and inputs for the 2 characters since its efficient and saves time.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Dash_2v1 : MonoBehaviour
{
    private Rigidbody2D rb2;
    public float dashSpeed2;
    private float dashTime2;
    public float startTime2;
    private int direction_2;
    public int rightTotal2 = 0;
    public float rightTime2 = 0;
    public int leftTotal2 = 0;
    public float leftTime2 = 0;
    public Animator anim2;

    void Start()
    {
        rb2 = GetComponent<Rigidbody2D>();
        dashTime2 = startTime2;
        anim2 = gameObject.GetComponent<Animator>();
    }
    void Update()
    {

        if (direction_2 == 0)
        {
            if (Input.GetKeyDown("d"))
            {
                rightTotal2 += 1;
                // if the player enters the right key which
            }

            if ((rightTotal2 == 1) && (rightTime2 < 0.2))
                rightTime2 += Time.deltaTime;
            //if the count for how many times they pressed t
            // then the timer activates

            if ((rightTotal2 == 1) && (rightTime2 >= 0.2))
            {
                rightTime2 = 0;
                rightTotal2 = 0;
                // if the count for left key pressed is stil

                anim2.SetBool("broly_dash", false);
                // this sets the double dash to false if the
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Dash : MonoBehaviour
{
    private Rigidbody2D rb;
    public float dashSpeed;
    private float dashTime;
    public float startTime;
    private int direction;
    public int rightTotal = 0;
    public float rightTime = 0;
    public int leftTotal = 0;
    public float leftTime = 0;
    public Animator anim;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        dashTime = startTime;
        anim = gameObject.GetComponent<Animator>();
    }
    void Update()
    {

        if( direction == 0)
        {
            if (Input.GetKeyDown(KeyCode.D))
            {
                rightTotal += 1;
                // if the player enters the right key which is
            }

             if((rightTotal ==1) && (rightTime < 0.2))
                rightTime += Time.deltaTime;
            //if the count for how many times they pressed th
            // then the timer activates

                if((rightTotal == 1) && (rightTime >= 0.2))
                {
                    rightTime = 0;
                    rightTotal = 0;
                  // if the count for left key pressed is stil

                    anim.SetBool("goku_dash", false);
                // this sets the double dash to false if the t
```

The character inputs are flexible to use since they are quite common in other fighting games. Such as the AWSD buttons to move, and the IOPJKL to attack on pc fighting games are often used. An issue that my users are currently facing is that they aren't given the option to customize their controls and restart with one click since I haven't set up the game to show a win scene at the game and offer them to restart. To play again with full hp they have to go back to menu and press play from there. I will need to implement this into the game afterwards.

Below are the controls I have assigned for movement and attacks which within the traditional format for most fighting games on pc.




```
if (Input.GetKeyDown("i") && !punch)
```

```
if (Input.GetKeyDown("o") && !kick)
```

```
if (Input.GetKeyDown("l") && !SA)
```

Error prevention:

An error that occurred while playing the game was already mentioned in the consistency section which is the health bars sometimes don't work for one of the characters. Another error is the second time going back to the menu and pressing play loads the game but doesn't play the intro animations for the characters and doesn't allow the user to move. Only way to play is by pressing "Esc" which enables the user to move the characters. The last one they have seen is when the attacks are pressed sequentially the character frames sometimes freeze on one sprite.

Above you can see that the animations for the characters intro is frozen, this happens when I load up the game the second time after going back to the menu through the pause menu.

Test No.	What test	How it's tested	Test data	Good/Bad/Boundary	Expected result	Actual result
1	Menu buttons	Click with mouse or navigate with buttons	Using mouse and buttons such as arrow keys	Good: left arrow key Bad: "q" button Boundary: mouse and keyboard	Menu should allow the user to access all the features and modes the game has	Hovering over button emits a colour to show interaction and currently the player can use the play button to load the game and quit button to close the game (page 79 for reference)
2	Fight system	Use inputs assigned to each character move	I,O,P,J,K,L	Good: "I" Bad: "v" Boundary: n/a	See if the attacks are landing on box collider	Attack lands (page 63)
3	Attack animations	Test if animations if conditions are met	I,O,P	Good: "I" Bad: "G" Boundary: N/A	See if the animations for specific attacks are displayed	Attack animations are playing when user enters specified key, (page 58 for reference)
4	Movement	Use the horizontal and vertical inputs to see if character moves	W,S,A,D	Good: "W" Bad: "B" Boundary: N/A	The user will be moving in response to each of the user inputs so will the enemy	Character moves left, right , and can jump. they can also double dash if the right or left is pressed twice. (54 and 72)
5	Spam counter code	See if attacks are disabled after using them after a certain number of times	I,O,P,	Good: "I" Bad: "C" Boundary: pressing "I" 3 times	The attack or action will be disabled for a specific amount of time	Partially done
6	Character selection	Check if the character chosen is loaded into	Using mouse and buttons	Good: use mouse to click character box Bad: click outside the box	The character should be loaded into the game	The user can navigate through different

		the fight and their move set also exist when using them		Boundary: the edge of the box	with their assigned attacks and abilities	characters and select which on e to spawn into the game with(page 91)
7	Customization	Check if skins, super attacks and skills are able to be used after unlocking them and using them in the game	Use mouse to select equips or use buttons	Good: use mouse to click what to purchase Bad: using mouse to click outside the boxes to purchase them Boundary: edge of box	The user should have access to the equips purchased in customization	In progress
8	Point tracker	Check if the points accumulated from playing is added to the point tracker and allows the user to purchase from the customization area	Beat level to see if point tracker value goes up	Good: outputs 100 after win Bad: outputs 0 after win Boundary: N/A	The value should increases when the levels from single player mode is won	In progress
9	Maps	The maps should be in the background of the fight after the user chooses which one they want to use	Use mouse and buttons	Good: Wanted map is displayed in game Bad: map not appearing in game Boundary: N/A	Maps will be displayed in the background	User can navigate through the different maps and load the game into whichever one they want (page 84)
10	Swap characters	Check if the user is able to swap characters	Input J	Good: character is swapped Bad: not swapped Boundary: not swapped If cool down is active	The character1 should swap with character2	User can select what character they want to play as
11	Defend	Use input to defend against enemy	Input K	Good: if the damage dealt is cancelled Bad: damage not cancelled	The user should be able to defend and negate the damage that	Character defends and damage being given is cancelled(page 69)

				Boundary: damage dealt after defence is disabled	is being dealt to them	
12	Guard break	Use input to break enemy guard/ defence	Input L	Good: the other character guard is broken Bad: not broken Boundary: N/A	The guard of the opponent should break and let the attacker continue to attack the opponent	In progress
13	Health system	Check if the user and enemy health is decreased after taking damage	Use attacks to see if enemy is taking damage and check user is taking damage by letting enemy attack	Good: player1 health decreases after attack Bad: no damage dealt Boundary: health staying the same after being attacked in defending state	The health of both the opponents should decrease by the amount the attacker is applying to them. Example kick should do 5 damage and super should do 10	HP for both players are decreasing when damage is being given and stop when not. The health bars also stop when players are defending.(page 63 and page 75)

Test 1 Menu buttons: The completed version should allow the user to use all the features of the game such as selecting maps, skins and characters. As of now the navigation works. When testing oh no the main menu the play button and quit button worked the first time, I tested them I also added the customise button for later on when I actually implement that into the game when. I tested if the buttons get highlighted when I hover over the buttons and they work fine, however I think I can make the design of the main menu better since it's just a test and not finalized yet.

Test 2 Fight system : The kicks and punches land but I need to enable a way for the super attacks to land. The fight system currently only works for normal attacks such as punches and kicks and defending. I still need to research how to make super attacks work with the health system. I've already found a way for the user to do combos such as pressing the kick button three times will allow them to do a super attack however it doesn't apply damage and punching three times doesn't play the kick animation three times it just freezes on one frame then plays the super attack. I need to tweak how the animations plays when the kick is pressed and mange the leeway time for the intervals between inputs.

Test 3 Attack animations: All animations should play when user enters actions through keyboard. When testing the attacks I had issues with the animations playing, such as kicks were playing twice when I only wanted to make it play it once. I figured that I had to look at how long the animation was and tweak the transition time and exit time so the animations play properly.

Test 4 Movement: The first time coding the movement script and testing was just trail and errors because I didn't understand 2d physics then, the first test for moving left and right wasn't so hard because it was just inputs and the game object moving, the jumping however had issues since the player was always jumping if the 2d collider wasn't positioned properly so the game object would be inside the floor. After the real issues appeared which were the animations for when the player moves. The first test I tried to use 2 different sprites for moving forward and backward but backward I had issues trying to make 2 different animation so I stuck to 1. Then the transitions between animations such as moving left then jump didn't look right because either it didn't play the jump animation or the 1st frame for jumping only played because I didn't give it enough time to play on screen. Eventually I understood the animator and set up the transitioning between animation to play how I wanted them to.

Test 11 Defend: First when I made the defend code it stopped the health system for the character stop decreasing when the character was defending however the health bars did not go down because I hadn't implemented them yet. The first test to make the animation for the health decrease did not work because I did not reference the box colliders, eventually I did find a way to make the health bar go down but after a short while it stopped working for unexcepted reasons then I tried other ways and tried to fix the problem but I could not find a way to make them work so I eventually found how to link my heath for script for when the characters normally fight each other. I linked the Script so it to my attack trigger so when the attack trigger lands on the other character it also links to the heath bar script and takes away the value of 5 from the health bar image.

Test 13 Health system: The health bars when 1st tested worked without errors , however when used with float values it didn't because of using several colliders, so further across the development I decided to use capsule soldiers since they don't have corners and when the characters interacted with each other the just slide of the each other when jumping, only one was needed for each character. This also helped with not using float values. However when I implemented the health bars I had issues to trying to find out how to make the hp image decrease when ever the player gets hit, I eventually found that I can just link the script for the health bars to the character attack triggers.

Success criteria

Criteria	Detail	progress
For the player to be able to use the menu with ease	**The menu will be a simple layout where they will be able to easily switch one to another**	**Complete**
For the player to pause the game whenever they want	**While fighting if they want to pause there will be a pause button in the game, when paused there will be a list of missions that they will be able to complete in that certain level. The menu will also be there for them to quit the game.**	**Complete**
Quit game	**The user should be able to quit from the main menu and while playing the game**	**Complete**
Customize characters	**There will be a separate area in the game where they are able to customize their characters through completing missions and earning points.**	**Not started**
To be able to play the game with simple inputs	**The game will include simple keyboard inputs that the player will easily learn**	**Complete**
Accumulate points from playing the game	**The game will have a point tracker and will inform them how many points they will get from that level**	**Not started**
Music in game	**The game will have an option where they can choose what music they want to play while the game is open**	**Not started**
Selecting backgrounds/ maps	**There will be an option for the player to choose what background they want to play in**	**Complete**
Key binds	**This will enable them to assign what key each attack should be to their liking**	**Partially done**

Selecting characters	**Players should be able to choose what character they want to play as before entering a match**	**Complete**
2 player support	**The game should be able to allow 2 users to play the game and supporting the appropriate controls when in a match**	**Complete**
Health system	**Players should be able to apply the correct damage value corresponding to what they input**	**Complete**
Timer for certain attacks	**Attacks such as super attacks should be set a timer for it to be used again instead of allowing the player to spam it**	**Complete**
Input limiter depending on timer	**Since my game is aimed at countering spam, I will be setting a limit of how many attacks the player can input so that they will have to think about how they will attack instead repeatedly using them**	**Complete**

1:The menu only has play button, customise button, option button and quit button. Currently the play button and quit button are useable, the option button that displays the key binds doesn't work for character movement. The response time after using the main menu buttons were quick when the user's tried them. However, the character animations stop in the main menu after playing the game and going back to it for the second time, I will have to resolve this in the future.

2:Pause menu works when the user hits the "Esc" button and the game freezes so even if the user inputs any commands for the characters their functions are disabled however some buttons such as the animations for double dash and attacks work but it doesn't allow the characters to actually move around in the game when paused. Resume button works to go back to playing again so does the quit button. The menu button also works to go back to the main menu, the response to the button inputs are quick and don't have any errors when used. In the future I am also thinking about adding more options into the pause menu such as volume, music selection and mapping input/ help section for what the inputs do.

4:The inputs to play the game are fairly simple because it uses the traditional inputs other fighting games have on keyboard even said by Brandon "the keyboard controls are easy to pick", but I'm also thinking of adding an option where the user is able to have their own controls and save them for when they want to play after closing the game.

7:I have made 4 different maps which the users can select and enter the game with whatever characters they have chosen. The characters now successfully spawn into the game and when the same character is chosen the movement for player 1 and player is also separate. However, the damage being dealt to same character is not fully functional because after a certain number of attacks the damage dealt each of them stops.

Limitations

Due to time constraints I won't be able to add a variety of characters into the game, since this will mean new animations and code for the character attacks and skills.

I wasn't able to add a AI to the game for story mode since it will require a script for the user to read while playing in single player mode and also making the AI will be difficult as it requires them to be more difficult as they progress through the game.

To add customizable skins for the sprites and skills may prove hard to implement into the game since I would need to save all their progress of how many points they have acquired to purchase them.

Maintenance

if this game were to have an online version, I would have to ask users to vote for what characters they would like in the game on a monthly basis so that I can keep the game fresh for the public. If there are new characters that will be put in the game, I would have to make new animations and code specific for that character.

I would also have to update the game and make users download the patches so that they can play the game bug free. In the patch I may add new assets that I may not release will its downloaded but whenever the times comes I will implement them into the public version.

To improve my game I think adding different modes to play in may also be satisfying certain users that may want extra content other than just fighting such as mini games, waves of enemies, leader board for a time rush mode where users compete for finishing a level in the fastest time.

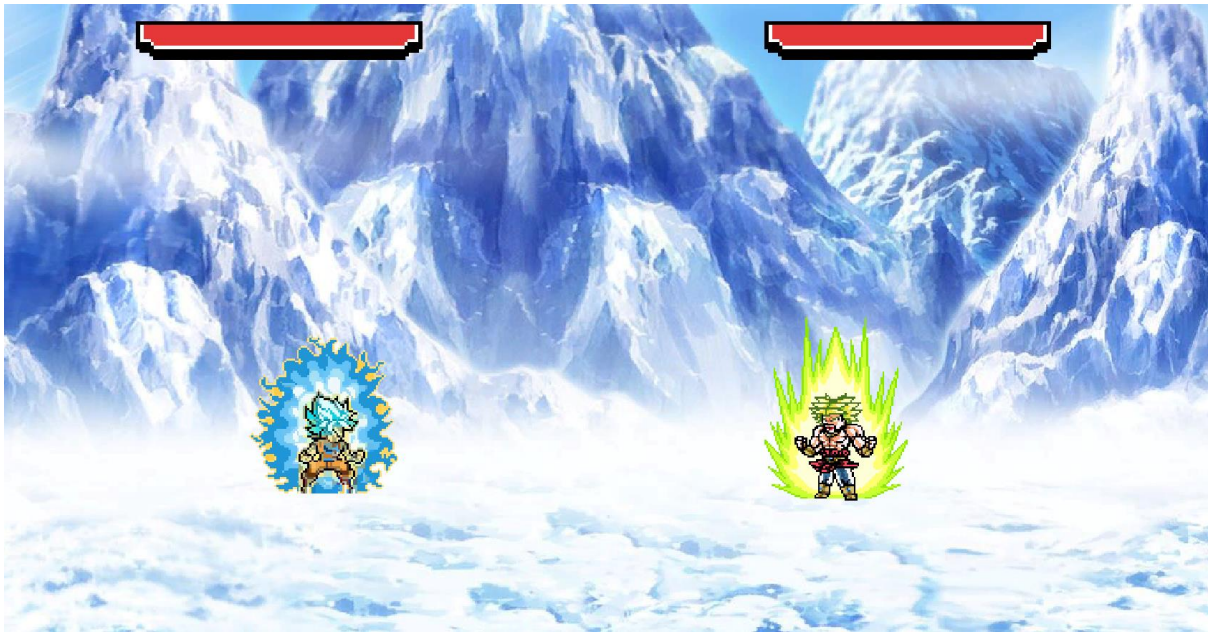
Evidence

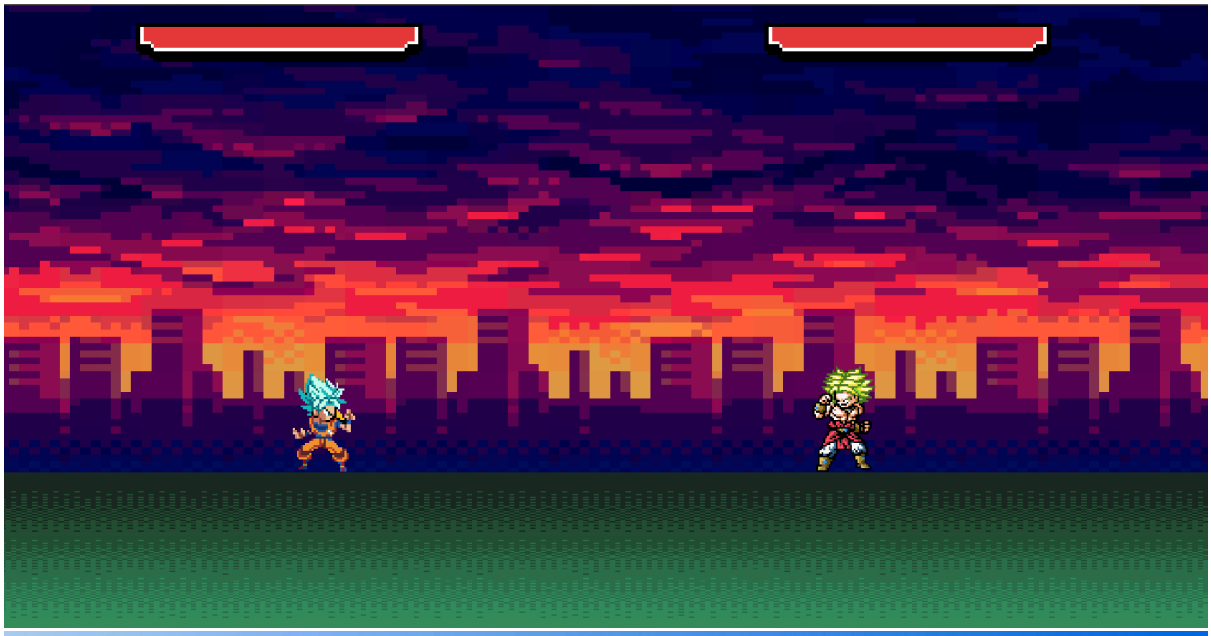

PLAY
CUSTOMIZE
OPTIONS
QUIT
SQUARE-UP!!


PLAYER 1 SELECT
PREVIOUS PLAYER
NEXT PLAYER
PLAYER 2


PLAYER 2 SELECT
PREVIOUS PLAYER
NEXT PLAYER
START


天下一武道会






RESUME
MENU
QUIT GAME


BACK
CONTROLS
JUMP
UP
DOWN
DOWN
LEFT
LEFT
RIGHT
RIGHT
SAVE

SCRIPTS/CODE

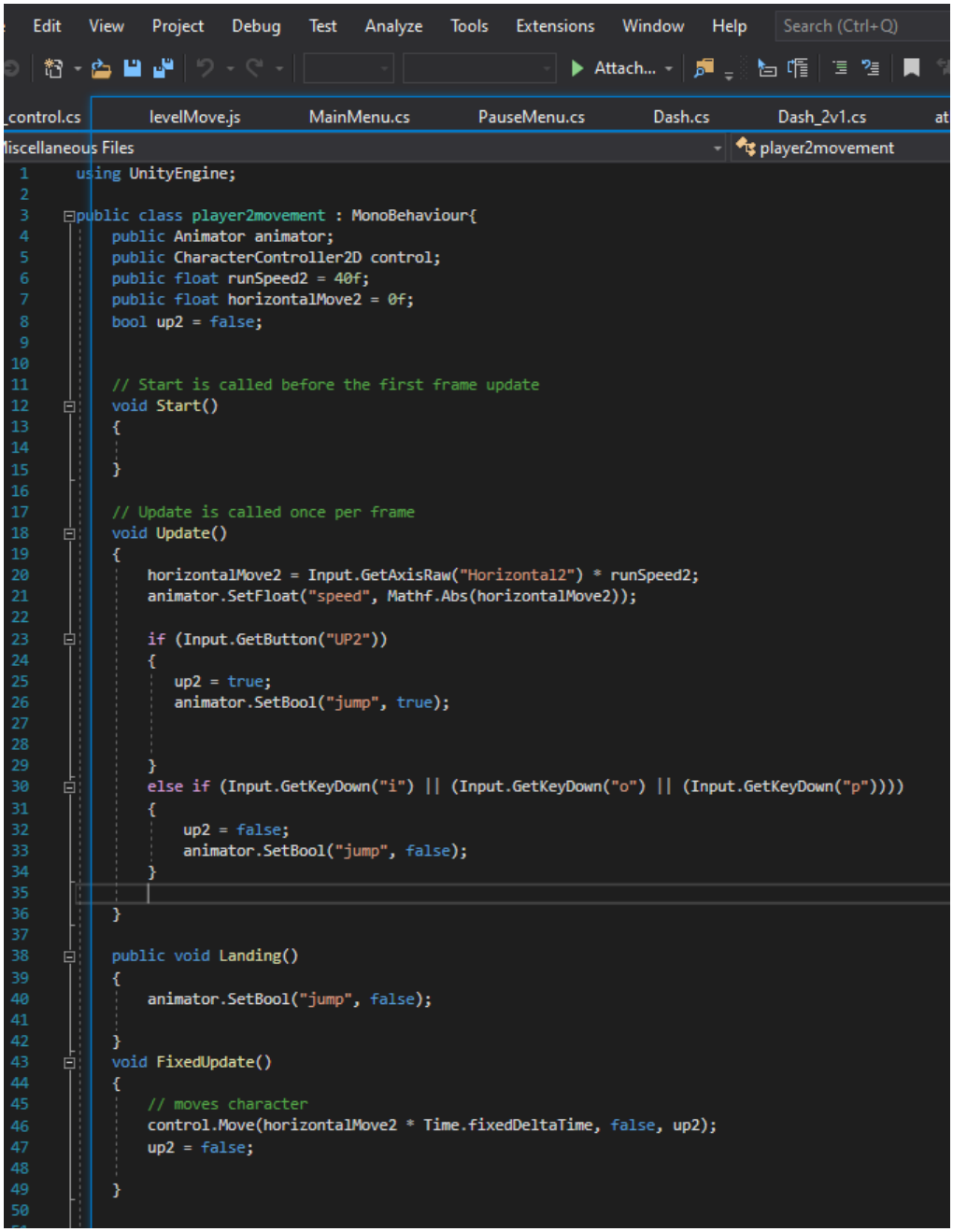
MOVEMENT

```
using UnityEngine;
using UnityEngine.Events;

2 references
public class CharacterController2D : MonoBehaviour
{
    [SerializeField] private float m_JumpForce = 400f;                          // Amount of force added when the player jumps.
    [Range(0, 1)] [SerializeField] private float m_CrouchSpeed = .36f;          // Amount of maxSpeed applied to crouching movement. 1 = 100%
    [Range(0, .3f)] [SerializeField] private float m_MovementSmoothing = .05f;  // How much to smooth out the movement
    [SerializeField] private bool m_AirControl = false;                         // Whether or not a player can steer while jumping;
    [SerializeField] private LayerMask m_WhatIsGround;                          // A mask determining what is ground to the character
    [SerializeField] private Transform m_GroundCheck;                           // A position marking where to check if the player is grounded.
    [SerializeField] private Transform m_CeilingCheck;                          // A position marking where to check for ceilings
    [SerializeField] private Collider2D m_CrouchDisableCollider;                // A collider that will be disabled when crouching


    const float k_GroundedRadius = .2f; // Radius of the overlap circle to determine if grounded
    private bool m_Grounded;            // Whether or not the player is grounded.
    const float k_CeilingRadius = .2f; // Radius of the overlap circle to determine if the player can stand up
    private Rigidbody2D m_Rigidbody2D;
    private bool m_FacingRight = true;  // For determining which way the player is currently facing.
    private Vector3 m_Velocity = Vector3.zero;

    [Header("Events")]
    [Space]

    public UnityEvent OnLandEvent;

    [System.Serializable]
    2 references
    public class BoolEvent : UnityEvent<bool> { }

    public BoolEvent OnCrouchEvent;
    private bool m_wasCrouching = false;

    0 references
    private void Awake()
    {
        m_Rigidbody2D = GetComponent<Rigidbody2D>();

        if (OnLandEvent == null)
            OnLandEvent = new UnityEvent();

        if (OnCrouchEvent == null)
            OnCrouchEvent = new BoolEvent();
    }
```

```
0 references
private void FixedUpdate()
{
    bool wasGrounded = m_Grounded;
    m_Grounded = false;

    // The player is grounded if a circlecast to the groundcheck position hits anything designated as ground
    // This can be done using layers instead but Sample Assets will not overwrite your project settings.
    Collider2D[] colliders = Physics2D.OverlapCircleAll(m_GroundCheck.position, k_GroundedRadius, m_WhatIsGround);
    for (int i = 0; i < colliders.Length; i++)
    {
        if (colliders[i].gameObject != gameObject)
        {
            m_Grounded = true;
            if (!wasGrounded)
                OnLandEvent.Invoke();
        }
    }
}


2 references
public void Move(float move, bool crouch, bool jump)
{
    // If crouching, check to see if the character can stand up
    if (!crouch)
    {
        // If the character has a ceiling preventing them from standing up, keep them crouching
        if (Physics2D.OverlapCircle(m_CeilingCheck.position, k_CeilingRadius, m_WhatIsGround))
        {
            crouch = true;
        }
    }

    //only control the player if grounded or airControl is turned on
    if (m_Grounded || m_AirControl)
    {

        // If crouching
        if (crouch)
        {
            if (!m_wasCrouching)
            {
                m_wasCrouching = true;
                OnCrouchEvent.Invoke(true);
            }

            // Reduce the speed by the crouchSpeed multiplier
            move *= m_CrouchSpeed;

            // Disable one of the colliders when crouching
            if (m_CrouchDisableCollider != null)
                m_CrouchDisableCollider.enabled = false;
```

```
        else
        {
            // Enable the collider when not crouching
            if (m_CrouchDisableCollider != null)
                m_CrouchDisableCollider.enabled = true;

            if (m_wasCrouching)
            {
                m_wasCrouching = false;
                OnCrouchEvent.Invoke(false);
            }
        }

        // Move the character by finding the target velocity
        Vector3 targetVelocity = new Vector2(move * 10f, m_Rigidbody2D.velocity.y);
        // And then smoothing it out and applying it to the character
        m_Rigidbody2D.velocity = Vector3.SmoothDamp(m_Rigidbody2D.velocity, targetVelocity, ref m_Velocity, m_MovementSmoothing);

        // If the input is moving the player right and the player is facing left...
        if (move > 0 && !m_FacingRight)
        {
            // ... flip the player.
            Flip();
        }
        // Otherwise if the input is moving the player left and the player is facing right...
        else if (move < 0 && m_FacingRight)
        {
            // ... flip the player.
            Flip();
        }
    }
    // If the player should jump...
    if (m_Grounded && jump)
    {
        // Add a vertical force to the player.
        m_Grounded = false;
        m_Rigidbody2D.AddForce(new Vector2(0f, m_JumpForce));
    }
}


2 references
private void Flip()
{
    // Switch the way the player is labelled as facing.
    m_FacingRight = !m_FacingRight;

    transform.Rotate(0f, 180f, 0f);
}
```

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
public class playermovement : MonoBehaviour {
    public Animator animator; // this references the animator which controls all the animations that the character does in the game
    public CharacterController2D control;// this is the script i will be referncing to make the charcter move smoothly

    public float runSpeed = 40f;
    public float horizontalMove = 0f;
    public float horizontalMove1 = 0f;
    // the variables above will control movement speed
    public bool up = false;

    void Start()
    {



    }

    public void Update()
    {
        {
            // horizontalMove = Input.GetAxisRaw("Horizontal") * runSpeed;
            // horizontalMove1 = Input.GetAxisRaw("Horizontal1") * runSpeed;

           horizontalMove = Input.GetAxisRaw("Horizontal") * runSpeed;

            //horizontalMove1 = Input.GetAxisRaw("Horizontal1") * runSpeed;

            //GetAxisRaw determines if the player is going left or right by the value -1 moving left and 1 when moving right then runSpeed multplies the value by 40 to give the character speed
            animator.SetFloat("speed", Mathf.Abs(horizontalMove));
           // animator.SetFloat("speed1", Mathf.Abs(horizontalMove1));
            // this is used to get the input from the keyboard for horizantal movement and runSpeed will control how fast they can move
            // the animator refernced will play the animation for moving left anf right
            //Mathf.Abs is used to keep the speed value always poistive even when it is going left which is -1
        }

        if (Input.GetButton("UP"))
        {
            up = true;
           //up1 = true;
            animator.SetBool("jump", true);
            // this is to figure out if the player is jumping or not, if os the animtion for jump will be played
        }
        else if (Input.GetKeyDown("i") || (Input.GetKeyDown("o") || (Input.GetKeyDown("p"))))
        {
            up = false;
           // up1 = false;
            animator.SetBool("jump", false);
        }
    }
```


Edit View Project Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Attach...
_control.cs
levelMove.js
MainMenu.cs
PauseMenu.cs
Dash.cs
Dash_2v1.cs
Miscellaneous Files
player2movement
using UnityEngine;
public class player2movement : MonoBehaviour{
public Animator animator;
public CharacterController2D control;
public float runSpeed2 = 40f;
public float horizontalMove2 = 0f;
bool up2 = false;
// Start is called before the first frame update
void Start()
{
}
// Update is called once per frame
void Update()
{
horizontalMove2 = Input.GetAxisRaw("Horizontal2") * runSpeed2;
animator.SetFloat("speed", Mathf.Abs(horizontalMove2));
if (Input.GetButton("UP2"))
{
up2 = true;
animator.SetBool("jump", true);
}
else if (Input.GetKeyDown("i") || (Input.GetKeyDown("o") || (Input.GetKeyDown("p"))))
{
up2 = false;
animator.SetBool("jump", false);
}
}
public void Landing()
{
animator.SetBool("jump", false);
}
void FixedUpdate()
{
// moves character
control.Move(horizontalMove2 * Time.fixedDeltaTime, false, up2);
up2 = false;
}

Dash code: player 1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Dash : MonoBehaviour
{
    private Rigidbody2D rb;
    public float dashSpeed;
    private float dashTime;
    public float startTime;
    private int direction;
    public int rightTotal = 0;
    public float rightTime = 0;
    public int leftTotal = 0;
    public float leftTime = 0;
    public Animator anim;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        dashTime = startTime;
        anim = gameObject.GetComponent<Animator>();
    }
    void Update()
    {

        if( direction == 0)
        {
            if (Input.GetKeyDown(KeyCode.D))
            {
                rightTotal += 1;
                // if the player enters the right key which is D then the count for how many times they pressed that will increase by 1
            }

             if((rightTotal ==1) && (rightTime < 0.2))
                rightTime += Time.deltaTime;
             //if the count for how many times they pressed they left key is 1 and the time for long it has been since they pressed it is less than 0.2 seconds
             // then the timer activates

                if((rightTotal == 1) && (rightTime >= 0.2))
                {
                    rightTime = 0;
                    rightTotal = 0;
                  // if the count for left key pressed is still 1 and has been more than or equal to 0.2 seconds then count goes back to 0 so will the timer

                    anim.SetBool("goku_dash", false);
                // this sets the double dash to false if the the above if statement is true

                }
```

```
if ((rightTotal == 2) && (rightTime < 0.2))
{
    direction = 2;
    rightTotal = 0;
    anim.SetBool("goku_dash", true);
    //if left key is pressed twice and the time that has elapsed after the first count is still less than 0.2 seconds then the double dash animtion plays for player1

}

else if (Input.GetKeyDown(KeyCode.A))
{
    leftTotal += 1;
    // if the player enters the right key which is D then the count for how many times they pressed that will increase by 1
}

if ((leftTotal == 1) && (leftTime < 0.2))
    leftTime += Time.deltaTime;
//if the count for how many times they pressed they left key is 1 and the time for long it has been since they pressed it is less than 0.2 seconds
// then the timer activates

if ((leftTotal == 1) && (leftTime >= 0.2))
{
    leftTime = 0;
    leftTotal = 0;
    // if the count for left key pressed is still 1 and has been more than or equal to 0.2 seconds then count goes back to 0 so will the timer

    anim.SetBool("goku_dash", false);
    // this sets the double dash to false if the the above if statement is true

}

if ((leftTotal == 2) && (leftTime < 0.2))
{
    direction = 1;
    leftTotal = 0;
    anim.SetBool("goku_dash",true);
    //if left key is pressed twice and the time that has elapsed after the first count is still less than 0.2 seconds then the double dash animtion plays for player1
}

/*   else if (Input.GetKeyDown(KeyCode.W))
    {
        direction = 3;
    }
    else if (Input.GetKeyDown(KeyCode.S))
    {
        direction = 4;
    }
    */

/ the code below will handle how fast the speed is of the double dash
```

```
        }
        else
        {
            if(dashTime <= 0)
            {
                direction = 0;
                dashTime = startTime;
                rb.velocity = Vector2.zero;
            }
            else
            {
                dashTime -= Time.deltaTime;
                if(direction == 1)
                {
                    rb.velocity = Vector2.left * dashSpeed;
                }else if( direction ==2)
                {
                    rb.velocity = Vector2.right * dashSpeed;
                }

               /* else if (direction == 3)
                {
                    rb.velocity = Vector2.up * dashSpeed;
                }
                else if (direction == 2)
                {
                    rb.velocity = Vector2.down * dashSpeed;
                }*/
            }
        }

    }
}
```

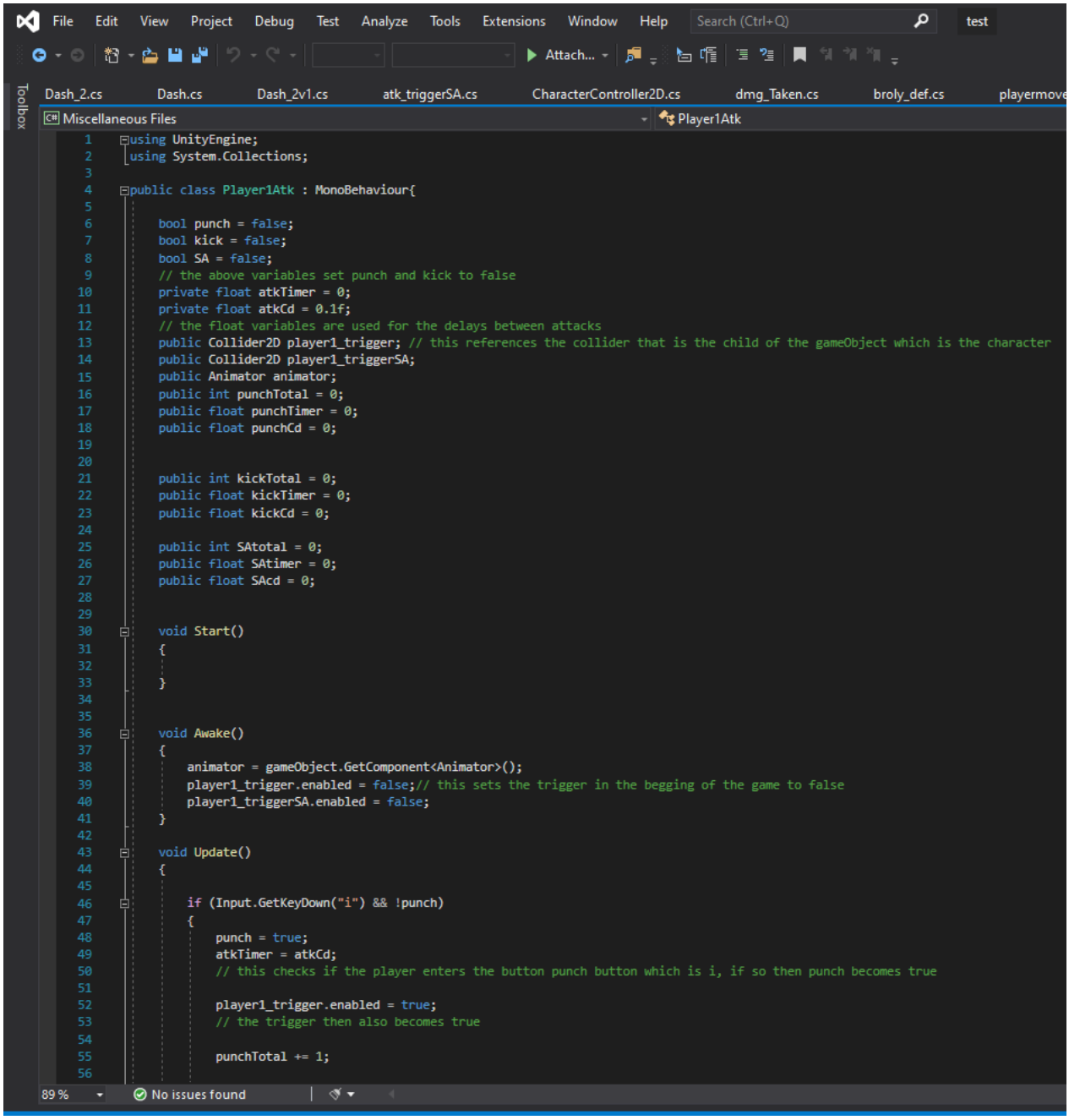
Dash: player 2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Dash_2 : MonoBehaviour
{
    private Rigidbody2D rb2;
    public float dashSpeed2;
    private float dashTime2;
    public float startTime2;
    private int direction_2;
    public int rightTotal2 = 0;
    public float rightTime2 = 0;
    public int leftTotal2 = 0;
    public float leftTime2 = 0;
    public Animator anim2;

    void Start()
    {
        rb2 = GetComponent<Rigidbody2D>();
        dashTime2 = startTime2;
        anim2 = gameObject.GetComponent<Animator>();
    }
    void Update()
    {

        if (direction_2 == 0)
        {
            if (Input.GetKeyDown(KeyCode.RightArrow))
            {
                rightTotal2 += 1;
                // if the player enters the right key which is D then the count for how many times they pressed that will increase by 1
            }

            if ((rightTotal2 == 1) && (rightTime2 < 0.2))
                rightTime2 += Time.deltaTime;
            //if the count for how many times they pressed they left key is 1 and the time for long it has been since they pressed it is less than 0.2 seconds
            // then the timer activates

            if ((rightTotal2 == 1) && (rightTime2 >= 0.2))
            {
                rightTime2 = 0;
                rightTotal2 = 0;
                // if the count for left key pressed is still 1 and has been more than or equal to 0.2 seconds then count goes back to 0 so will the timer

                anim2.SetBool("broly_dash", false);
                // this sets the double dash to false if the the above if statement is true

            }
```

```
if ((rightTotal2 == 2) && (rightTime2 < 0.2))
{
    direction_2 = 2;
    rightTotal2 = 0;
    anim2.SetBool("broly_dash", true);
    //if left key is pressed twice and the time that has elapsed after the first count is still less than 0.2 seconds then the double dash animtion plays for player1

}

else if (Input.GetKeyDown(KeyCode.LeftArrow ))
{
    leftTotal2 += 1;
    // if the player enters the right key which is D then the count for how many times they pressed that will increase by 1
}

if ((leftTotal2 == 1) && (leftTime2 < 0.2))
    leftTime2 += Time.deltaTime;
//if the count for how many times they pressed they left key is 1 and the time for long it has been since they pressed it is less than 0.2 seconds
// then the timer activates

if ((leftTotal2 == 1) && (leftTime2 >= 0.2))
{
    leftTime2 = 0;
    leftTotal2 = 0;
    // if the count for left key pressed is still 1 and has been more than or equal to 0.2 seconds then count goes back to 0 so will the timer

    anim2.SetBool("broly_dash", false);
    // this sets the double dash to false if the the above if statement is true

}

if ((leftTotal2 == 2) && (leftTime2 < 0.2))
{
    direction_2 = 1;
    leftTotal2 = 0;
    anim2.SetBool("broly_dash", true);
    //if left key is pressed twice and the time that has elapsed after the first count is still less than 0.2 seconds then the double dash animtion plays for player1
}

/*   else if (Input.GetKeyDown(KeyCode.W))
    {
        direction = 3;
    }
    else if (Input.GetKeyDown(KeyCode.S))
    {
        direction = 4;
    }
    */

// the code below will handle how fast the speed is of the double dash
```

```
    }
    else
    {
        if (dashTime2 <= 0)
        {
            direction_2 = 0;
            dashTime2 = startTime2;
            rb2.velocity = Vector2.zero;
        }
        else
        {
            dashTime2 -= Time.deltaTime;
            if (direction_2 == 1)
            {
                rb2.velocity = Vector2.left * dashSpeed2;
            }
            else if (direction_2 == 2)
            {
                rb2.velocity = Vector2.right * dashSpeed2;
            }

            /* else if (direction == 3)
             {
                 rb.velocity = Vector2.up * dashSpeed;
             }
             else if (direction == 2)
             {
                 rb.velocity = Vector2.down * dashSpeed;
             }*/
        }
    }

}
```

Attacks: player 1




```
using UnityEngine;
using System.Collections;

public class Player1Atk : MonoBehaviour{

    bool punch = false;
    bool kick = false;
    bool SA = false;
    // the above variables set punch and kick to false
    private float atkTimer = 0;
    private float atkCd = 0.1f;
    // the float variables are used for the delays between attacks
    public Collider2D player1_trigger; // this references the collider that is the child of the gameObject which is the character
    public Collider2D player1_triggerSA;
    public Animator animator;
    public int punchTotal = 0;
    public float punchTimer = 0;
    public float punchCd = 0;


    public int kickTotal = 0;
    public float kickTimer = 0;
    public float kickCd = 0;

    public int SAtotal = 0;
    public float SAtimer = 0;
    public float SAcd = 0;


    void Start()
    {

    }


    void Awake()
    {
        animator = gameObject.GetComponent<Animator>();
        player1_trigger.enabled = false;// this sets the trigger in the begging of the game to false
        player1_triggerSA.enabled = false;
    }

    void Update()
    {

        if (Input.GetKeyDown("i") && !punch)
        {
            punch = true;
            atkTimer = atkCd;
            // this checks if the player enters the button punch button which is i, if so then punch becomes true

            player1_trigger.enabled = true;
            // the trigger then also becomes true

            punchTotal += 1;
```

```
58
59          /*  if (punchTotal == 4)
60            {
61                punchTimer = punchCd;
62                punchTotal = 0;
63            }
64
65            if(punchTimer > 0)
66            {
67                punchTimer -= Time.deltaTime;
68                punch = false;
69                player1_trigger.enabled = false;
70                punchTotal = 0;
71            }
72            else
73            {
74                punch = true;
75                player1_trigger.enabled = true;
76            }*/
77
78            if (punch)
79            {
80                if (atkTimer > 0 )
81                {
82                    atkTimer -= Time.deltaTime;// if the atk timer is bigger then 0 then this line will decrease the time by delta time which works as a real timer
83                }
84                else
85                {
86                    punch = false;
87                    player1_trigger.enabled = false;
88                    // if player is not usinf the punch key button then punching becomes false and the trigger will also be false
89                }
90            }
91
92            animator.SetBool("punch",punch);
93            // this will set the punch animation in the animator of player 1 to true or false depending on the two if statements above
94
95            if (Input.GetKeyDown("o") && !kick)
96            {
97               kick = true;
98               atkTimer = atkCd;
99               // this checks if the player enters the button punch button which is i, if so then punch becomes true
100
101              player1_trigger.enabled = true;
102              // the trigger then also becomes true
103
104              kickTotal += 1;
105           }
106
107          // if (kickTotal == 4)
108           {
109              kickTimer = kickCd;
110              kickTotal = 0;
111           }
112
113         /*  if (kickTimer > 0)
```

Miscellaneous Files — Player1Atk

```
/*  if (kickTimer > 0)
  {
      kickTimer -= Time.deltaTime;
      kick = false;
      player1_trigger.enabled = false;
      kickTotal = 0;
  }*/

  if (kick)
  {
      if (atkTimer > 0)
      {
          atkTimer -= Time.deltaTime;// if the atk timer is bigger then 0 then this line will decrease the time by delta time which works as a real timer
      }
      else
      {
          kick = false;
          player1_trigger.enabled = false;
          // if player is not usinf the punch key button then punching becomes false and the trigger will also be false
      }
  }

  animator.SetBool("kick", kick);
  // this will set the punch animation in the animator of player 1 to true or false depending on the two if statements above

if (Input.GetKeyDown("1") && !SA)
  {
     SA = true;
     Debug.Log("sa");
     atkTimer = atkCd;
      // this checks if the player enters the button punch button which is i, if so then punch becomes true
      StartCoroutine(SAtrigger());

      // the trigger then also becomes true

      SAtotal += 1;
  }

IEnumerator SAtrigger()
  {
      yield return new WaitForSeconds(0.45f);
      player1_triggerSA.enabled = true;
  }

 if (SAtotal == 1)
  {
      SAtimer = SAcd;
      SAtotal = 0;
  }
```

```
 if (SAtimer > 0)
  {
      SAtimer -= Time.deltaTime;
      SA = false;
      player1_triggerSA.enabled = false;
      SAtotal = 0;
  }

 /*  else
   {
       SA = true;
       player1_triggerSA.enabled = true;
   }*/

  if (SA)
  {
      if (atkTimer > 0)
      {
          atkTimer -= Time.deltaTime;// if the atk timer is bigger then 0 then this line will decrease the time by delta time which works as a real timer
      }
      else
      {
          SA = false;
          player1_triggerSA.enabled = false;
          // if player is not using the super attack button then SA becomes false and the trigger will also be false
      }
  }

  animator.SetBool("SA1", SA);
  // this will set the punch animation in the animator of player 1 to true or false depending on the two if statements above
```

Attacks: player 2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player2_atk : MonoBehaviour
{
    bool punch_b = false;
    bool kick_b = false;
    bool SA = false;


    private float atkTimer = 0;
    private float atkCd = 0.3f;

    public Collider2D player2_trigger;
    public Collider2D player2_triggerSA;

    public Transform SA_firepoint;
    public GameObject SAPrefab;

    public Animator animator;

    public int SAtotal = 0;
    public float SAtimer = 0;
    public float SAcd = 0;

    void Awake()
    {
        animator = gameObject.GetComponent<Animator>();
        player2_trigger.enabled = false;

    }

    void Update()
    {

        if (Input.GetKeyDown(",") && !punch_b)
        {
            punch_b = true;
            atkTimer = atkCd;

            player2_trigger.enabled = true;

        }

        if (punch_b)
        {
            if (atkTimer > 0)
            {
                atkTimer -= Time.deltaTime;
            }
            else
            {
                punch_b = false;
                player2_trigger.enabled = false;
            }
```

Miscellaneous Files — Player2_atk

```
        if (punch_b)
        {
            if (atkTimer > 0)
            {
                atkTimer -= Time.deltaTime;
            }
            else
            {
                punch_b = false;
                player2_trigger.enabled = false;
            }
        }

        animator.SetBool("punch_b", punch_b);

        if (Input.GetKeyDown(".") && !kick_b)
        {
            kick_b = true;
            atkTimer = atkCd;

            player2_trigger.enabled = true;

        }

        if (kick_b)
        {
            if (atkTimer > 0)
            {
                atkTimer -= Time.deltaTime;
            }
            else
            {
                kick_b = false;
                player2_trigger.enabled = false;
            }

        }
        animator.SetBool("kick_b", kick_b);

        if (Input.GetKeyDown("m") && !SA)
        {
            SA = true;
            Debug.Log("sa");
            atkTimer = atkCd;
            // this checks if the player enters the button punch button which is i, if so then punch becomes true
            StartCoroutine(SAtrigger());

            Shoot();
            // the trigger then also becomes true

            SAtotal += 1;
        }

        IEnumerator SAtrigger()
        {
            yield return new WaitForSeconds(0.45f);
            player2_triggerSA.enabled = true;
        }

        if (SAtotal == 1)
        {
            SAtimer = SAcd;
            SAtotal = 0;
        }

        if (SAtimer > 0)
        {
            SAtimer -= Time.deltaTime;
            SA = false;
            player2_triggerSA.enabled = false;
            SAtotal = 0;
        }

        /*  else
          {
              SA = true;
              player1_triggerSA.enabled = true;
          }*/

        if (SA)
        {
            if (atkTimer > 0)
            {
                atkTimer -= Time.deltaTime;// if the atk timer is bigger then 0 then this line will decrease the time by delta time which works as a real timer
            }
            else
            {
                SA = false;
                player2_triggerSA.enabled = false;
                // if player is not usinf the punch key button then punching becomes false and the trigger will also be false
            }
        }

        animator.SetBool("brolySA", SA);


    }


    void Shoot()
    {
        Instantiate(SAPrefab, SA_firepoint.position, SA_firepoint.rotation);
    }
}
```

Attack triggers

Normal attacks

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class atkTrigger : MonoBehaviour
{
    public int dmg = 5;
    public int dmg_SA = 10;

    void OnTriggerEnter2D(Collider2D col)// this is a sunction that checks any 2d colliders that it come in contact with
    {
        if (col.isTrigger != true && col.CompareTag("player2"))
            //if the box collider this script is attached to is triggered through punch or kick and the contact object has a tag of player2
        {
            col.SendMessageUpwards("Damage", dmg);
            healthBar2.health2 -= 5f;
            // Damage function will be called and subtract 5 from the other player health in the player2 script which is called Dmg_taken
        }

```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class atk_trigger2 : MonoBehaviour
{
    public int dmg2 = 5;
    Animator animator;

    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.isTrigger != true && col.CompareTag("player1"))
        {
            col.SendMessageUpwards("Damage", dmg2);
            healthbar.health -= 5f;
        }
    }

```

Super attacks

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class atk_triggerSA : MonoBehaviour
{
    public int dmg = 5;
    public int dmg_SA = 10;

    void OnTriggerEnter2D(Collider2D col)// this is a sunction that checks any 2d colliders that it come in contact with
    {
        if (col.isTrigger != true && col.CompareTag("player2"))
        //if the box collider this script is attached to is triggered through punch or kick and the contact object has a tag of player2
        {
            col.SendMessageUpwards("Damage", dmg_SA);
            healthBar2.health2 -= 10f;
            // Damage function will be called and subtract 5 from the other player health in the player2 script which is called Dmg_taken
        }

        /*    if (col.isTrigger != true && col.CompareTag("def_2") && col.CompareTag("player2"))
            {
                col.SendMessageUpwards("Damage", dmg_cancel);
                dmg = 0;
                Debug.Log(dmg);
            }

        }
        */

```

Health system

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class dmg_Taken : MonoBehaviour
{
    public int curHealth = 0;
    public int maxHealth = 100;

    void start()
    {
        curHealth = maxHealth;// this sets the player2 helath to 100

    }

    void Update()
    {

        if (curHealth <= 0)
        {
            Destroy(gameObject);
        }
        // if the player health is below 0 then the player2 will be destroyed in the game
    }
    public void Damage(int damage)
    {
        curHealth -= damage;
        // this function will be refernced by the player1 trigger whenever player1 is attacking player2 hp will decrease the amount of damage it sends to this script

    }
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Dmg_Taken2 : MonoBehaviour
{
    public float curHealth2 = 0;
    public float maxHealth2 = 100;

    void start()
    {
        curHealth2 = maxHealth2;
    }

    void Update()
    {

        if (curHealth2 <= 0)
        {
            Destroy(gameObject);
        }

    }
    public void Damage(int damage)
    {
        curHealth2 -= damage;

    }
```

Health bars

Miscellaneous Files

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class healthbar : MonoBehaviour
{
    Image healthBar;
    public float maxHealth = 100f;
    public static float health;
    void Start()
    {
        healthBar = GetComponent<Image>();
        health = maxHealth;
    }

    // Update is called once per frame
    void Update()
    {
        healthBar.fillAmount = health / maxHealth;
    }
}

```

Miscellaneous Files — healthBar2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class healthBar2 : MonoBehaviour
{

    Image healthBar_2;
    public float maxHealth2 = 100f;
    public static float health2;
    void Start()
    {
        healthBar_2 = GetComponent<Image>();// this refernces the image for player2 health bar
        health2 = maxHealth2;
    }

    // Update is called once per frame
    void Update()
    {
        healthBar_2.fillAmount = health2 / maxHealth2;
    }


}

```

Defend/ guard code

llaneous Files goku_def

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class goku_def : MonoBehaviour
{

    public Animator anim;
    bool def_g = false;
    public Collider2D def2_trigger;
    // both of the above triggers will be used to reference player2's colliders that will be activated when the defend button is activated
    private float defTimer = 0;
    private float defCd = 3;
    // the 2 float variables above will be used to set the countdown of how long the def barrier lasts for

    void Awake()
    {
        def2_trigger.enabled = false;
        anim = gameObject.GetComponent<Animator>();
    }

    void Update()
    {

        if (Input.GetKeyDown("p") && !def_g)
        {
            def_g = true;
            def2_trigger.enabled = true;
            // if the user has entered the guard key then defend will be true and so will the trigger

            defTimer = defCd;// this will make the barrier last for 3 seconds
            transform.gameObject.tag = ("def");
            // this line of code will change the player2's tag from player2 to def_2 this way when player1 attacks it will detect the tag and cancel the damage being dealt
            Debug.Log(tag);
            print(tag);
            // the 2 lines above will print to the console if the tag has changed
        }

        else if (Input.GetKeyDown("i") || (Input.GetKeyDown("o")))
        {
         def_g = false;
         anim.SetBool("def_g", false);
        }

        if (def_g)
        {
            if (defTimer > 0)
            {
                defTimer -= Time.deltaTime;
                // this will act as a timer for the guard barrier to go down
            }
            else
            {
                def_g = false;
                def2_trigger.enabled = false;
                transform.gameObject.tag = ("player1");
                // if player 2 isnt defending anymore then teh tag will go back to player2 and player1 can return to doing damage
            }

            anim.SetBool("def_g", def_g);
            // this line will play the aniimtion of the guard barrier depending on whether player 2 has entered the guard button

        }
```

Miscellaneous Files broly_def

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class broly_def : MonoBehaviour
{

    public Animator anim;
    bool def_b = false;
    public CircleCollider2D def2_trigger;
    public Collider2D def_trigger;
    // both of the above triggers will be used to reference player2's colliders that will be activated when the defend button is activated
    private float defTimer = 0;
    private float defCd = 3;
    // the 2 float variables above will be used to set the countdown of how long the def barrier lasts for

     void Awake()
    {
        def2_trigger.enabled = false;
        anim = gameObject.GetComponent<Animator>();
    }

    void Update()
    {

        if (Input.GetKeyDown("/") && !def_b)
        {
            def_b = true;
            def2_trigger.enabled = true;
            // if the user has entered the guard key then defend will be true and so will the trigger

            defTimer = defCd;// this will make the barrier last for 3 seconds
            transform.gameObject.tag = ("def_2");
            // this line of code will change the player2's tag from player2 to def_2 this way when player1 attacks it will detect the tag and cancel the damage being dealt
            Debug.Log(tag);
            print(tag);
            // the 2 lines above will print to the console if the tag has changed


        }

        else if (Input.GetKeyDown("[4]") || (Input.GetKeyDown("[5]")))
        {
            def_b = false;
            anim.SetBool("def_b", false);
        }

        if (def_b)
        {
            if (defTimer > 0)
            {
                defTimer -= Time.deltaTime;
                // this will act as a timer for the guard barrier to go down
            }
            else
            {
                def_b = false;
                def2_trigger.enabled = false;
                transform.gameObject.tag = ("player2");
                // if player 2 isnt defending anymore then teh tag will go back to player2 and player1 can return to doing damage
            }

            anim.SetBool("def_b", def_b);
            // this line will play the aniimtion of the guard barrier depending on whether player 2 has entered the guard button

        }
```

Animation for damage taken

Miscellaneous Files

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class goku_dmg : MonoBehaviour
{
    public Animator onHit;
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("player2_trigger"))
        {
            onHit.SetBool("taken_g", true);
        }
    }

    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.CompareTag("player2_trigger"))
        {
            onHit.SetBool("taken_g", false);
        }
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class dmg_Anim : MonoBehaviour
{
    public Animator onHit;
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("player1_trigger"))
        {
            onHit.SetBool("taken_b", true);
        }
    }

    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.CompareTag("player1_trigger"))
        {
            onHit.SetBool("taken_b", false);
        }
    }
}
```

Main menu

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MainMenu : MonoBehaviour
{
    public void Play()
    {
        SceneManager.LoadScene("Player1Select");
    }
    public void Options()
    {
        SceneManager.LoadScene("Controls");
    }

    public void QuitGame ()
    {
        Application.Quit();
        Debug.Log("Quit");
    }
}

```

Pause menu

Miscellaneous Files

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class PauseMenu : MonoBehaviour
{
    public static bool Paused = false;

    public GameObject pauseMenuUI;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (Paused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        Paused = false;
    }

    void Pause()
    {
        pauseMenuUI.SetActive(true);
        Time.timeScale = 0f;
        Paused = true;
    }

    public void Menu()
    {
        SceneManager.LoadScene("main menu");
    }

    public void Quit()
    {
        Debug.Log("quit");
        Application.Quit();
    }
}

```

Map selection code

Miscellaneous Files

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MapSelect : MonoBehaviour
{
    public void Map1()
    {

        SceneManager.LoadScene("WorldTournament");
    }
    public void Map2()
    {

        SceneManager.LoadScene("IceMap");
    }
    public void Map3()
    {

        SceneManager.LoadScene("TestScene");
    }
    public void Map4()
    {

        SceneManager.LoadScene("skyMap");
    }
}

```

Character select code: Player 1

SelectScript.cs | MapSelect.cs | PauseMenu.cs | MainMenu.cs | healthba

Miscellaneous Files

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class SelectScript : MonoBehaviour
{
    public GameObject Broly;
    public GameObject Goku;
    private Vector3 CharPos;
    private Vector3 offScreen;
    private int CharInt = 1;
    private SpriteRenderer GokuRender, BrolyRender;

    private readonly string selectChar = "selectedCharacter";

    private void Awake()
    {

        CharPos = Broly.transform.position;
        offScreen = Goku.transform.position;
        GokuRender = Goku.GetComponent<SpriteRenderer>();
        BrolyRender = Broly.GetComponent<SpriteRenderer>();
    }

    public void NextChar()
    {
        switch (CharInt)
        {
            case 1:
                PlayerPrefs.SetInt(selectChar,1);
                GokuRender.enabled = false;
                Goku.transform.position = offScreen;
                Broly.transform.position = CharPos;
                BrolyRender.enabled = true;
                CharInt++;
                break;
            case 2:
                PlayerPrefs.SetInt(selectChar, 2);
                BrolyRender.enabled = false;
                Broly.transform.position = offScreen;
                Goku.transform.position = CharPos;
                GokuRender.enabled = true;
                CharInt++;
                ResetInt();
                break;
            default:
                ResetInt();
                break;
        }

    }

```

Miscellaneous Files

```
52
53      public void PrevChar()
54      {
55          switch (CharInt)
56          {
57              case 1:
58                  PlayerPrefs.SetInt(selectChar, 1);
59                  GokuRender.enabled = false;
60                  Goku.transform.position = offScreen;
61                  Broly.transform.position = CharPos;
62                  BrolyRender.enabled = true;
63                  ResetInt();
64                  break;
65              case 2:
66                  PlayerPrefs.SetInt(selectChar, 2);
67                  BrolyRender.enabled = false;
68                  Broly.transform.position = offScreen;
69                  Goku.transform.position = CharPos;
70                  GokuRender.enabled = true;
71                  CharInt--;
72                  break;
73              default:
74                  ResetInt();
75                  break;
76          }
77      }
78
79      private void ResetInt()
80      {
81          if(CharInt >= 2)
82          {
83              CharInt = 1;
84
85          }
86          else
87          {
88              CharInt = 2;
89          }
90      }
91
92      public void ConfirmButton()
93      {
94
95          SceneManager.LoadScene("TestScene");
96      }
97
98      public void Player2()
99      {
100
101         SceneManager.LoadScene("Player2Select");
102     }
103 }
104
```

Character select code: Player 2

Miscellaneous Files

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class SelectScript2 : MonoBehaviour
{
    public GameObject Broly2;
    public GameObject Goku2;
    private Vector3 CharPos2;
    private Vector3 offScreen2;
    private int CharInt2 = 1;
    private SpriteRenderer Goku2Render, Broly2Render;


    private readonly string selectChar2 = "selectedCharacter2";
    private void Awake()
    {
        CharPos2 = Broly2.transform.position;
        offScreen2 = Goku2.transform.position;
        Goku2Render = Goku2.GetComponent<SpriteRenderer>();
        Broly2Render = Broly2.GetComponent<SpriteRenderer>();
    }

    public void NextChar2()
    {
        switch (CharInt2)
        {
            case 1:
                PlayerPrefs.SetInt(selectChar2, 1);
                Goku2Render.enabled = false;
                Goku2.transform.position = offScreen2;
                Broly2.transform.position = CharPos2;
                Broly2Render.enabled = true;
                CharInt2++;
                break;
            case 2:
                PlayerPrefs.SetInt(selectChar2, 2);
                Broly2Render.enabled = false;
                Broly2.transform.position = offScreen2;
                Goku2.transform.position = CharPos2;
                Goku2Render.enabled = true;
                CharInt2++;
                ResetInt2();
                break;
            default:
                ResetInt2();
                break;
        }

    }

```

Miscellaneous Files

```
51
52      public void PrevChar2()
53      {
54          switch (CharInt2)
55          {
56              case 1:
57                  PlayerPrefs.SetInt(selectChar2, 1);
58                  Goku2Render.enabled = false;
59                  Goku2.transform.position = offScreen2;
60                  Broly2.transform.position = CharPos2;
61                  Broly2Render.enabled = true;
62                  ResetInt2();
63                  break;
64              case 2:
65                  PlayerPrefs.SetInt(selectChar2, 2);
66                  Broly2Render.enabled = false;
67                  Broly2.transform.position = offScreen2;
68                  Goku2.transform.position = CharPos2;
69                  Goku2Render.enabled = true;
70                  CharInt2--;
71                  break;
72              default:
73                  ResetInt2();
74                  break;
75          }
76      }
77
78      private void ResetInt2()
79      {
80          if (CharInt2 >= 2)
81          {
82              CharInt2 = 1;
83
84          }
85          else
86          {
87              CharInt2 = 2;
88          }
89      }
90
91      public void ConfirmButton()
92      {
93
94          SceneManager.LoadScene("maps");
95      }
96
97      public void Player2()
98      {
99
100         SceneManager.LoadScene("Player2Select");
101     }
102 }
```

In game character player 1

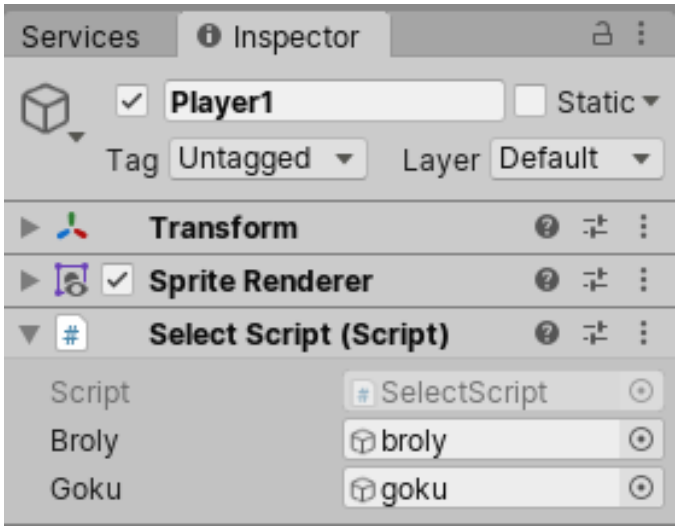
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GetMainChar : MonoBehaviour
{
    public Sprite GokuSprite, BrolySprite;
    public GameObject Goku;
    public GameObject Broly;
    private SpriteRenderer mySprite;
    private readonly string selectChar = "selectedCharacter";
    void Awake()
    {
        mySprite = this.GetComponent<SpriteRenderer>();
    }

    void Start()
    {
        int getChar;

        getChar = PlayerPrefs.GetInt(selectChar);

        switch (getChar)
        {
            case 1:
                mySprite.sprite = BrolySprite;
                Goku.SetActive(false);
                break;
            case 2:
                mySprite.sprite = GokuSprite;
                Broly.SetActive(false);
                break;
            default:
                mySprite.sprite = BrolySprite;
                Goku.SetActive(false);
                break;

        }
    }

}

```

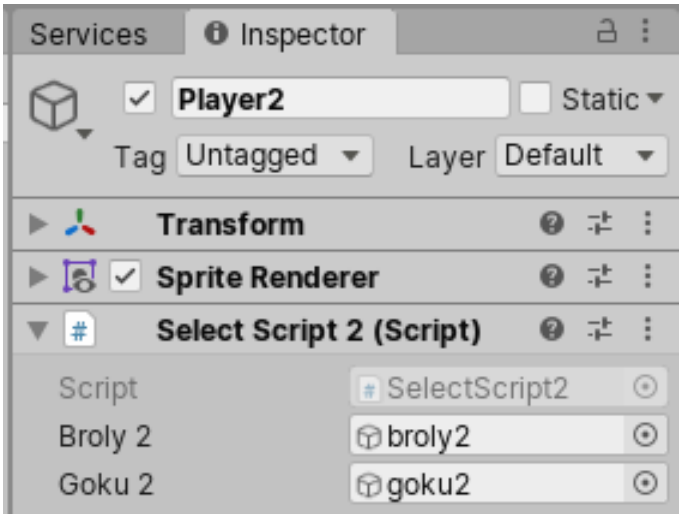
In game character player 2

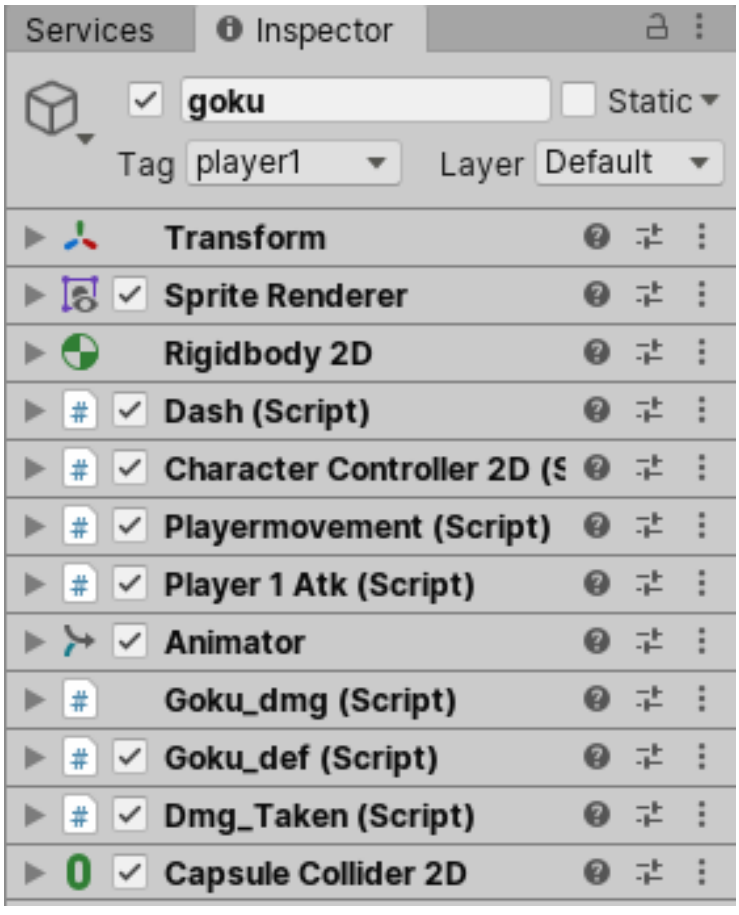
Miscellaneous Files

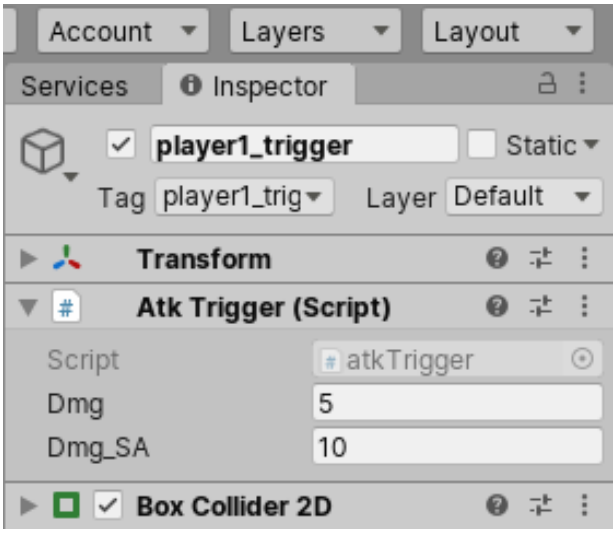
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MainChar2 : MonoBehaviour
{
    public Sprite GokuSprite2, BrolySprite2;
    public GameObject Goku2;
    public GameObject Broly2;
    private SpriteRenderer Sprite;
    private readonly string selectChar2 = "selectedCharacter2";
    void Awake()
    {
        Sprite = this.GetComponent<SpriteRenderer>();
    }

    void Start()
    {
        int getChar2;

        getChar2 = PlayerPrefs.GetInt(selectChar2);

        switch (getChar2)
        {
            case 1:
                Sprite.sprite = BrolySprite2;
                Goku2.SetActive(false);
                break;
            case 2:
                Sprite.sprite = GokuSprite2;
                Broly2.SetActive(false);
                break;
            default:
                Sprite.sprite = BrolySprite2;
                Goku2.SetActive(false);
                break;
        }
    }

}

```

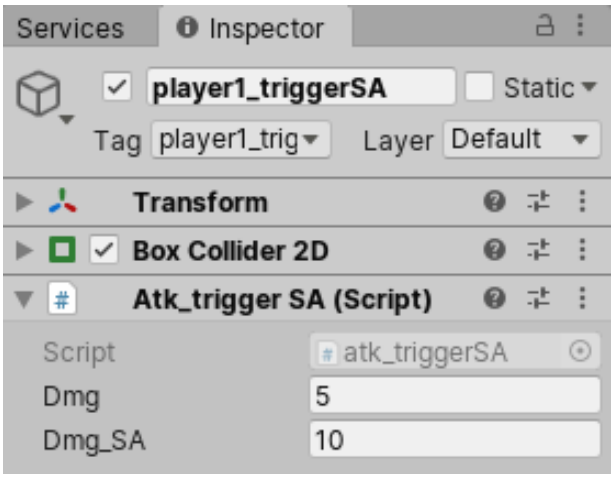
Hierarchy

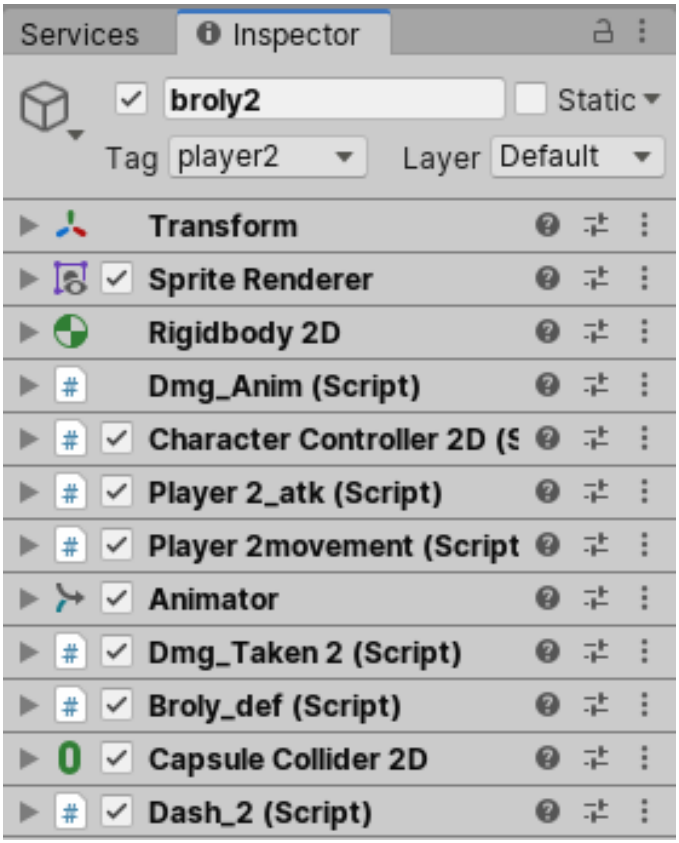

Services
Inspector
Player1
Static
Tag Untagged
Layer Default
Transform
Sprite Renderer
Select Script (Script)
Script SelectScript
Broly broly
Goku goku

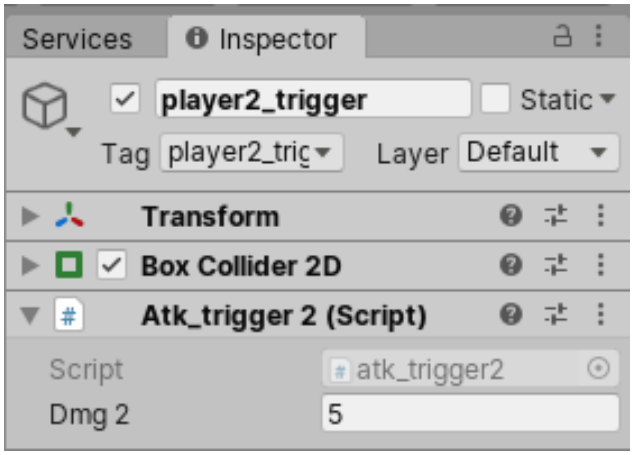

Services
Inspector
Player2
Static
Tag Untagged
Layer Default
Transform
Sprite Renderer
Select Script 2 (Script)
Script SelectScript2
Broly 2 broly2
Goku 2 goku2

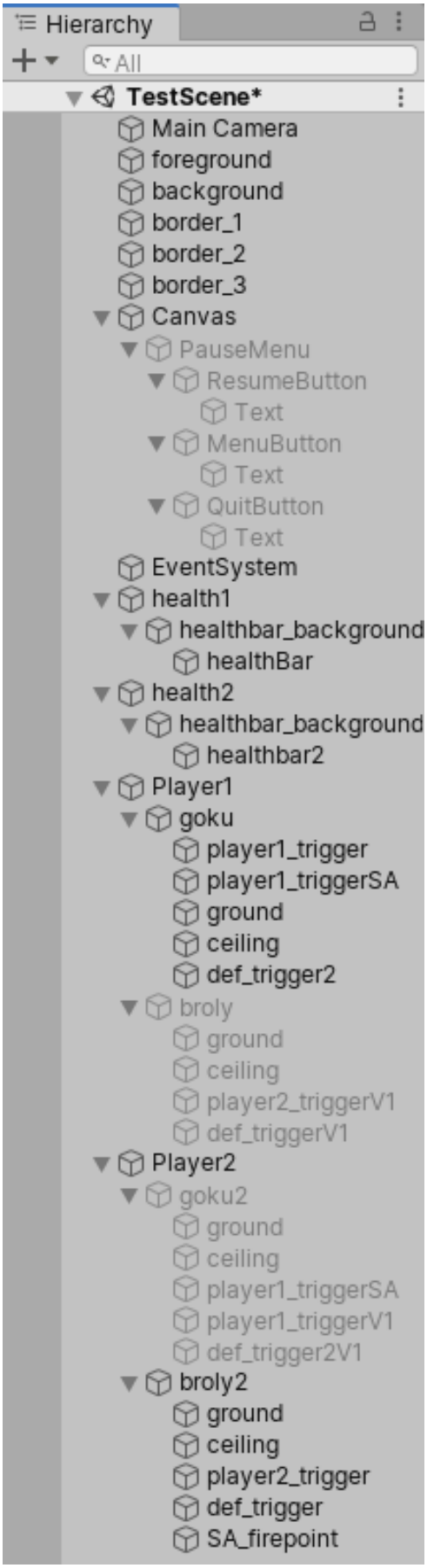

Services
Inspector
goku
Static
Tag player1
Layer Default
Transform
Sprite Renderer
Rigidbody 2D
Dash (Script)
Character Controller 2D (S
Playermovement (Script)
Player 1 Atk (Script)
Animator
Goku_dmg (Script)
Goku_def (Script)
Dmg_Taken (Script)
Capsule Collider 2D

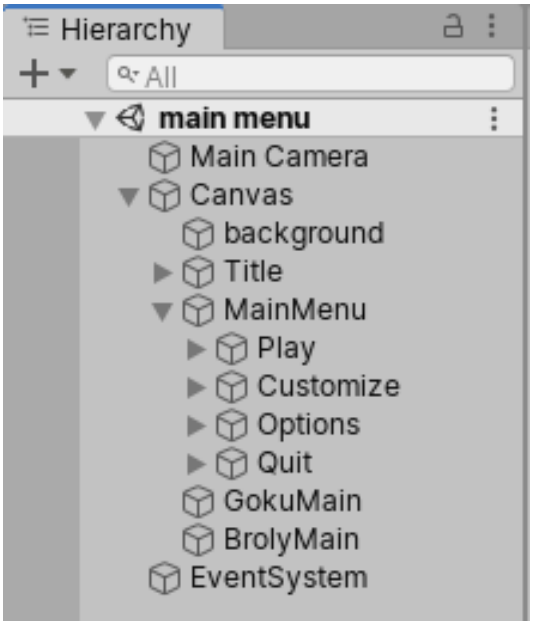

Account
Layers
Layout
Services
Inspector
player1_trigger
Static
Tag player1_trig
Layer Default
Transform
Atk Trigger (Script)
Script atkTrigger
Dmg 5
Dmg_SA 10
Box Collider 2D

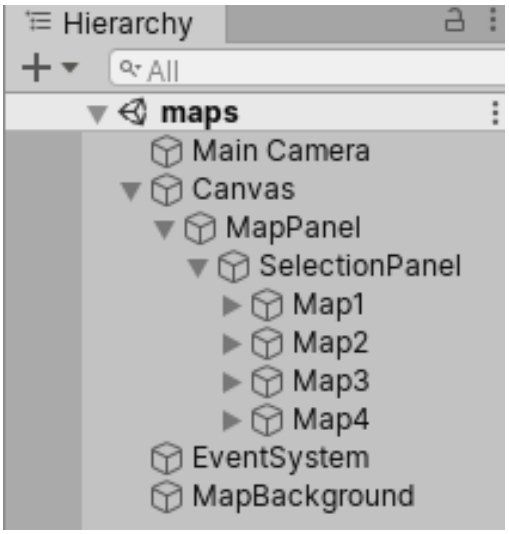

Services
Inspector
player1_triggerSA
Static
Tag player1_trig
Layer Default
Transform
Box Collider 2D
Atk_trigger SA (Script)
Script atk_triggerSA
Dmg 5
Dmg_SA 10

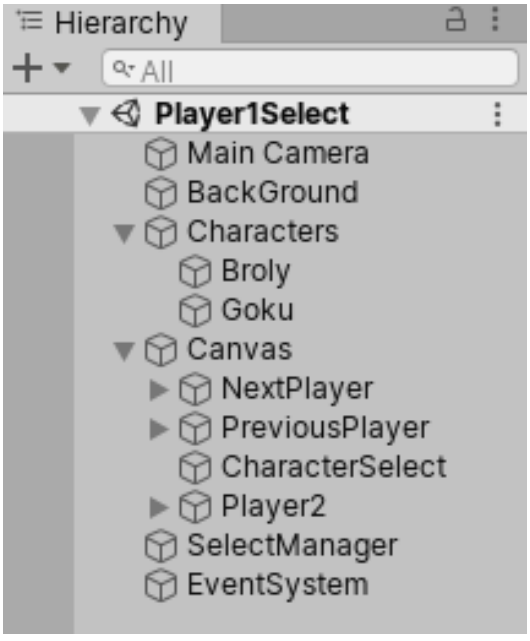

Services
Inspector
broly2
Static
Tag player2
Layer Default
Transform
Sprite Renderer
Rigidbody 2D
Dmg_Anim (Script)
Character Controller 2D (S
Player 2_atk (Script)
Player 2movement (Script
Animator
Dmg_Taken 2 (Script)
Broly_def (Script)
Capsule Collider 2D
Dash_2 (Script)

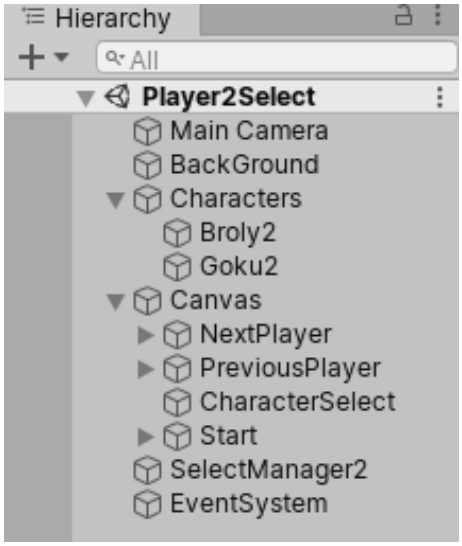

Services
Inspector
player2_trigger
Static
Tag player2_tric
Layer Default
Transform
Box Collider 2D
Atk_trigger 2 (Script)
Script
atk_trigger2
Dmg 2
5

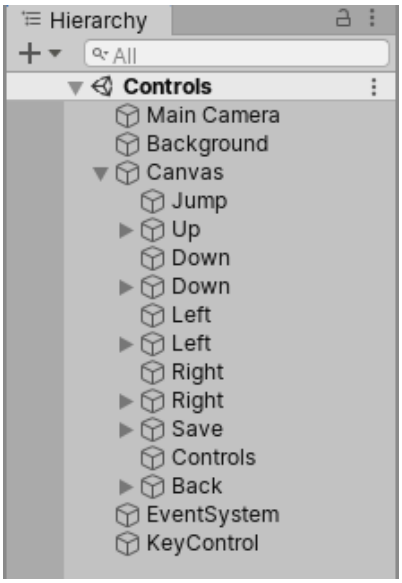

Hierarchy
All
TestScene*
Main Camera
foreground
background
border_1
border_2
border_3
Canvas
PauseMenu
ResumeButton
Text
MenuButton
Text
QuitButton
Text
EventSystem
health1
healthbar_background
healthBar
health2
healthbar_background
healthbar2
Player1
goku
player1_trigger
player1_triggerSA
ground
ceiling
def_trigger2
broly
ground
ceiling
player2_triggerV1
def_triggerV1
Player2
goku2
ground
ceiling
player1_triggerSA
player1_triggerV1
def_trigger2V1
broly2
ground
ceiling
player2_trigger
def_trigger
SA_firepoint

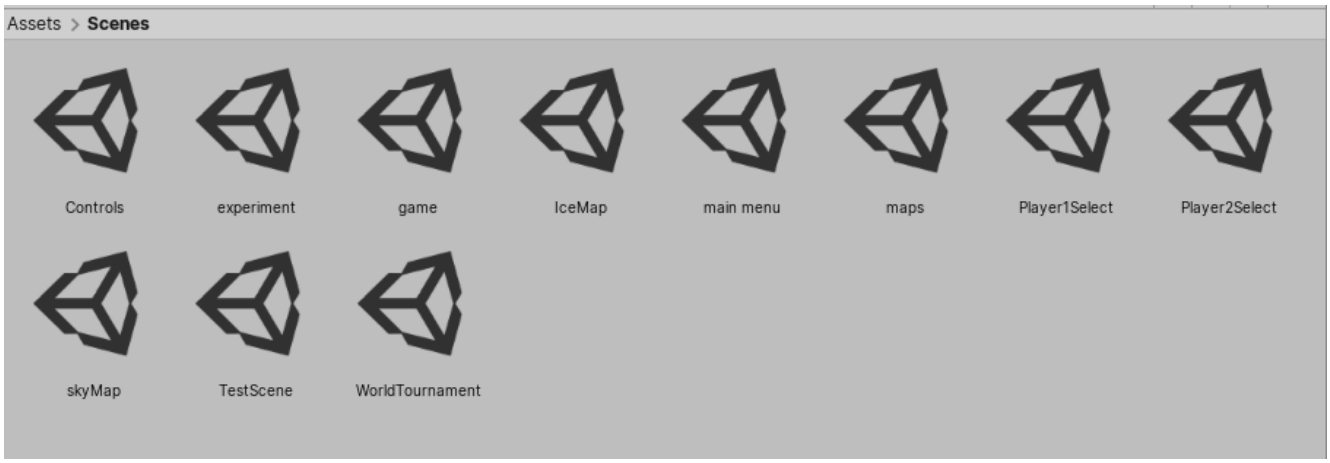

Hierarchy
All
main menu
Main Camera
Canvas
background
Title
MainMenu
Play
Customize
Options
Quit
GokuMain
BrolyMain
EventSystem


Hierarchy
All
maps
Main Camera
Canvas
MapPanel
SelectionPanel
Map1
Map2
Map3
Map4
EventSystem
MapBackground

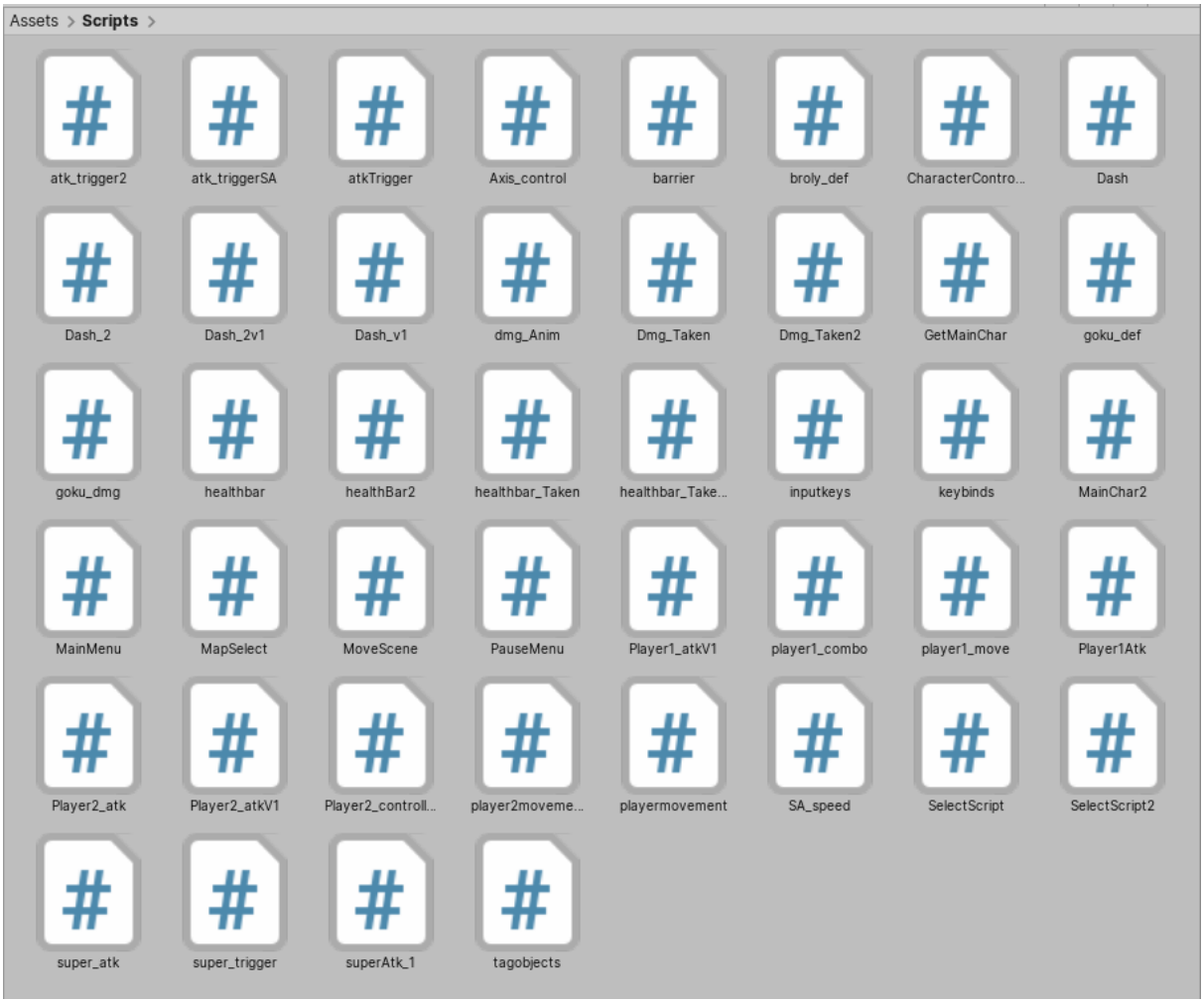

Hierarchy
All
Player1Select
Main Camera
BackGround
Characters
Broly
Goku
Canvas
NextPlayer
PreviousPlayer
CharacterSelect
Player2
SelectManager
EventSystem


Hierarchy
All
Player2Select
Main Camera
BackGround
Characters
Broly2
Goku2
Canvas
NextPlayer
PreviousPlayer
CharacterSelect
Start
SelectManager2
EventSystem

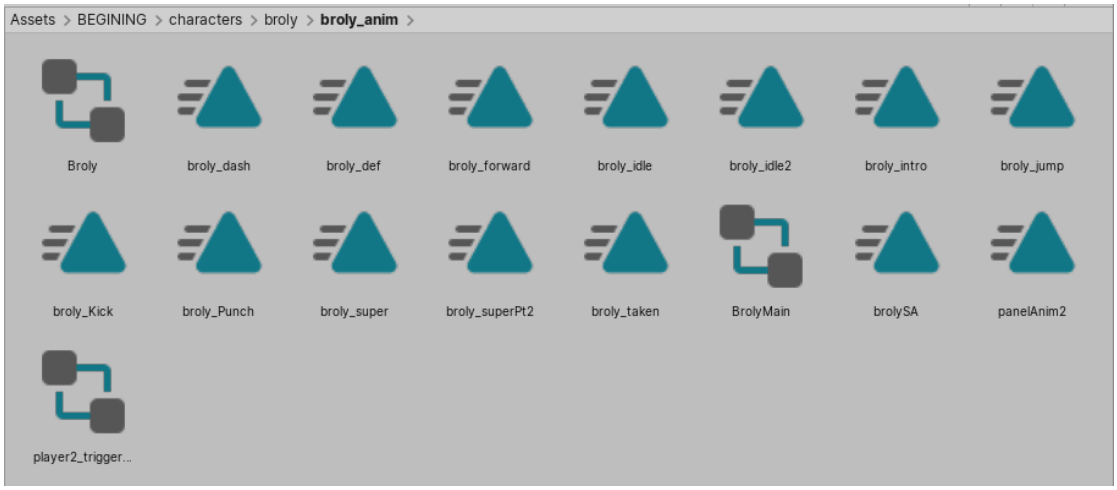

Hierarchy
All
Controls
Main Camera
Background
Canvas
Jump
Up
Down
Down
Left
Left
Right
Right
Save
Controls
Back
EventSystem
KeyControl

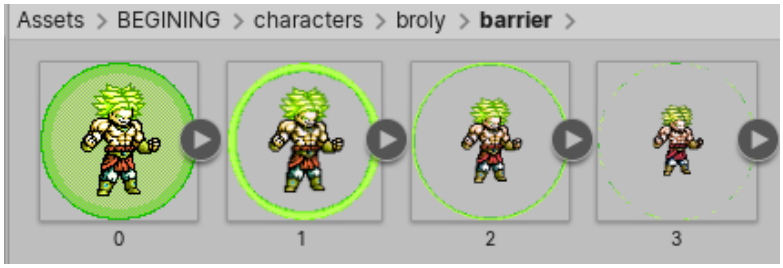

Assets > Scenes
Controls
experiment
game
IceMap
main menu
maps
Player1Select
Player2Select
skyMap
TestScene
WorldTournament

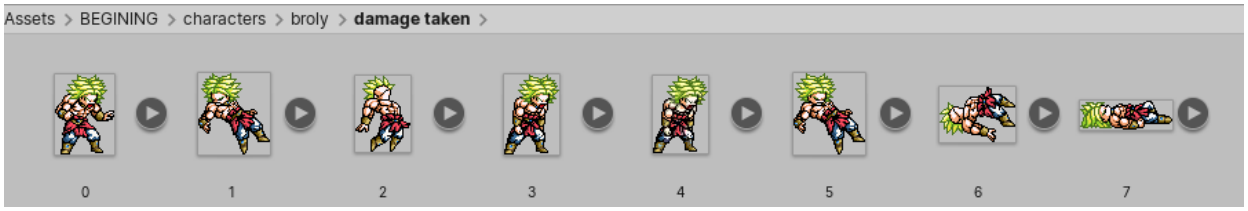

Assets > maps
datoc1q-fb7ff2c1...
dragon_ball_supe...
glacial_mountains...
near-buildings-bg
skill-desc_0000_f...
skill-desc_0003_bg
skyline-b
skymap

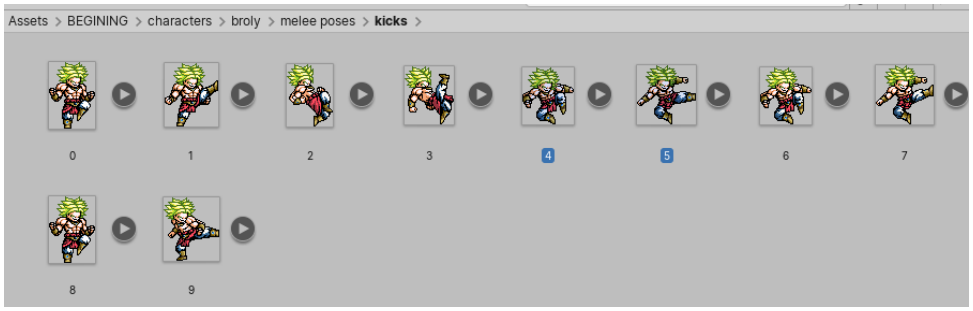

Assets > Scripts >
atk_trigger2
atk_triggerSA
atkTrigger
Axis_control
barrier
broly_def
CharacterContro...
Dash
Dash_2
Dash_2v1
Dash_v1
dmg_Anim
Dmg_Taken
Dmg_Taken2
GetMainChar
goku_def
goku_dmg
healthbar
healthBar2
healthbar_Taken
healthbar_Take...
inputkeys
keybinds
MainChar2
MainMenu
MapSelect
MoveScene
PauseMenu
Player1_atkV1
player1_combo
player1_move
Player1Atk
Player2_atk
Player2_atkV1
Player2_controll...
player2moveme...
playermovement
SA_speed
SelectScript
SelectScript2
super_atk
super_trigger
superAtk_1
tagobjects

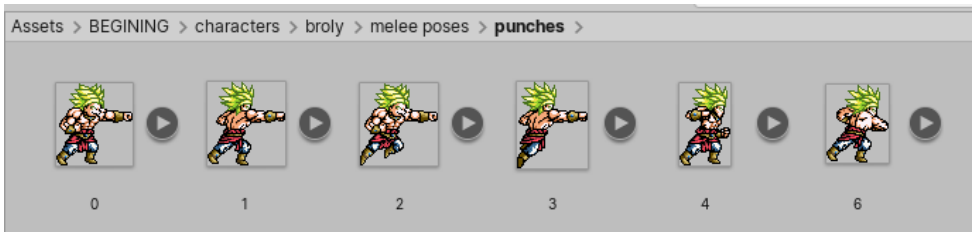

Assets > BEGINING > characters > goku > goku_animation >
goku
Goku
goku_attack1
goku_backward
goku_comboV1
goku_comboV2
goku_dash
goku_def
goku_forward
goku_idle
goku_idle2
goku_intro
goku_jump
goku_kick
goku_kick1
goku_punch
goku_punch1
goku_super
goku_taken
gokuPunch_co...
SelectGoku


Assets > BEGINING > characters > broly > broly_anim >
Broly
broly_dash
broly_def
broly_forward
broly_idle
broly_idle2
broly_intro
broly_jump
broly_Kick
broly_Punch
broly_super
broly_superPt2
broly_taken
BrolyMain
brolySA
panelAnim2
player2_trigger...


Assets > BEGINING > characters > broly > barrier >
0
1
2
3

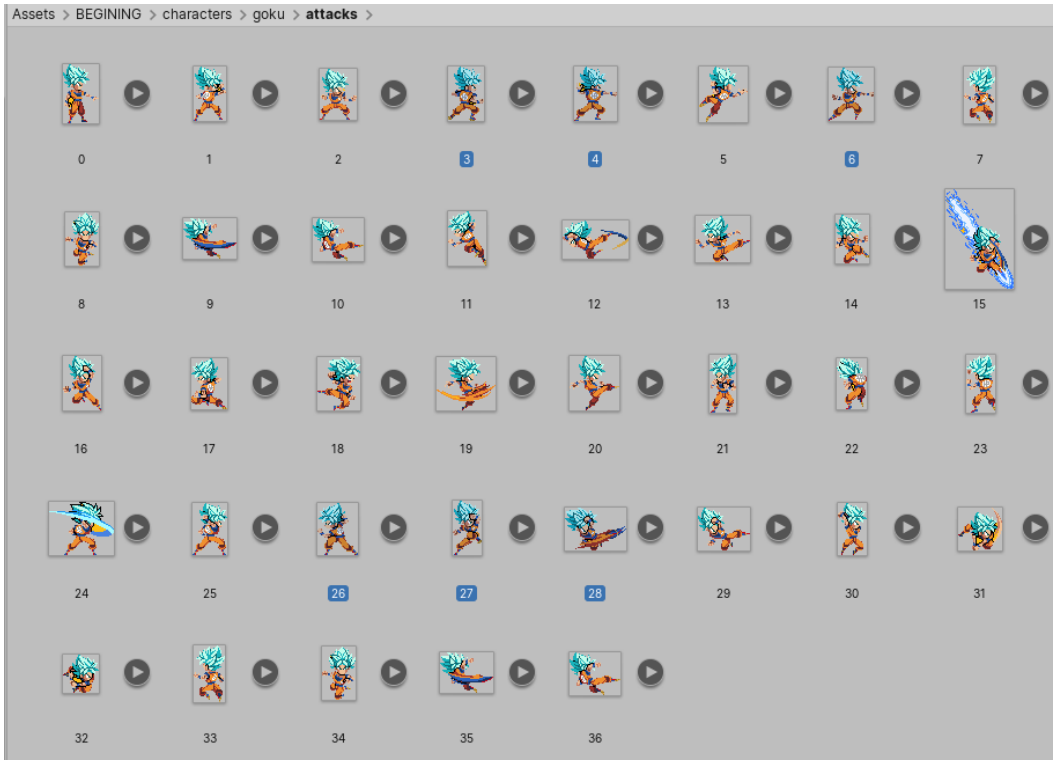

Assets > BEGINING > characters > broly > damage taken >
0
1
2
3
4
5
6
7

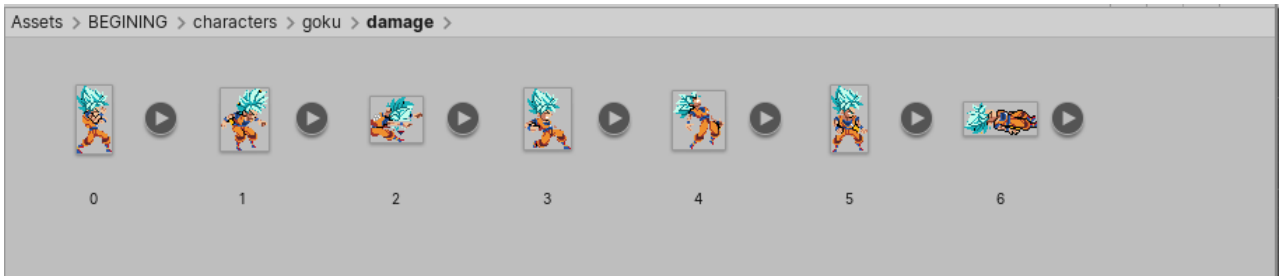

Assets > BEGINING > characters > broly > melee poses > kicks >
0
1
2
3
4
5
6
7
8
9

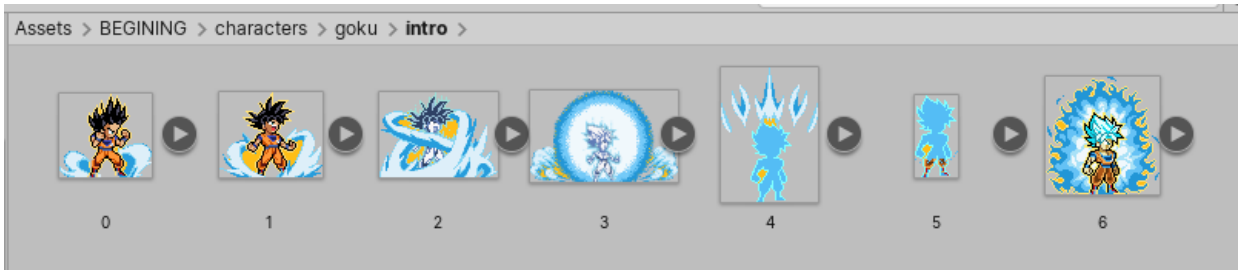

Assets > BEGINING > characters > broly > melee poses > punches >
0
1
2
3
4
6

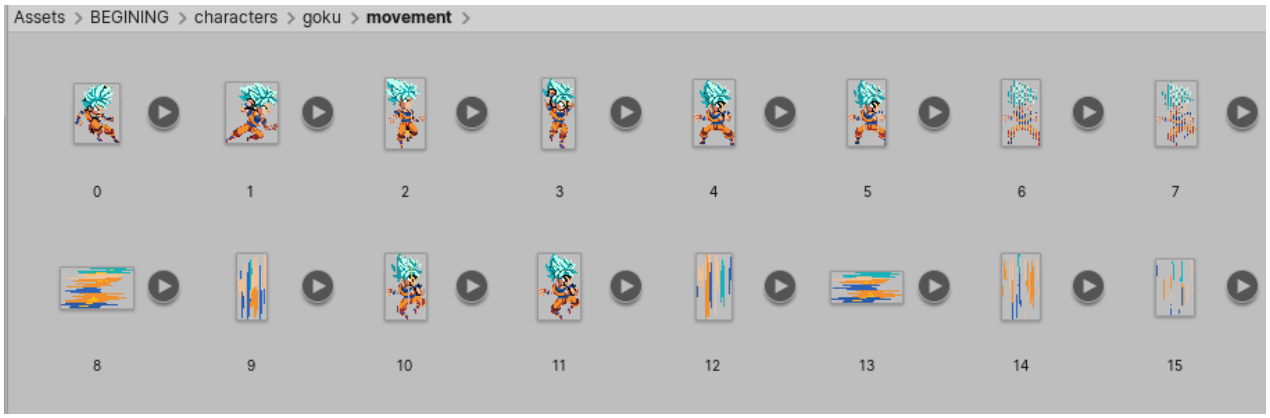

Assets > BEGINING > characters > broly > movements >
0
1
2
3
4
5
6
7
8
9


Assets > BEGINING > characters > broly > power up >
0
1
2
3
4
5
6
7
8
9
10
Broly

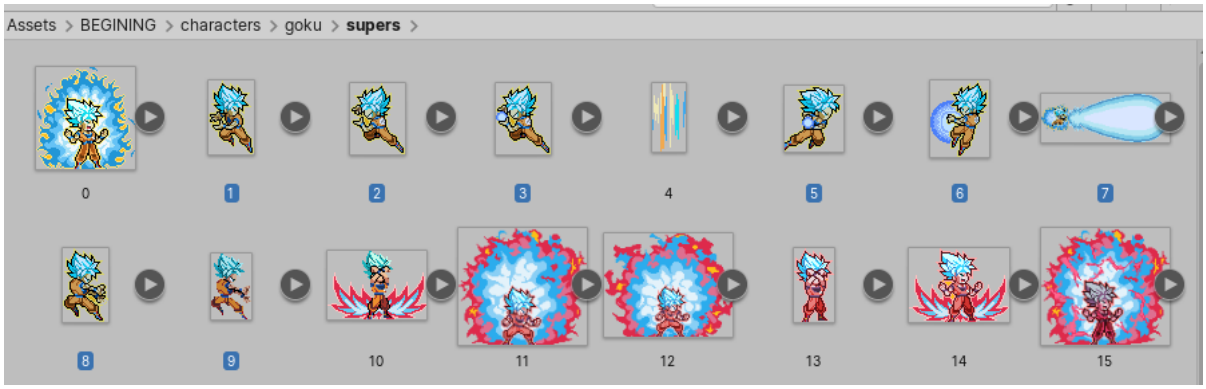

Assets > BEGINING > characters > goku > attacks >
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36

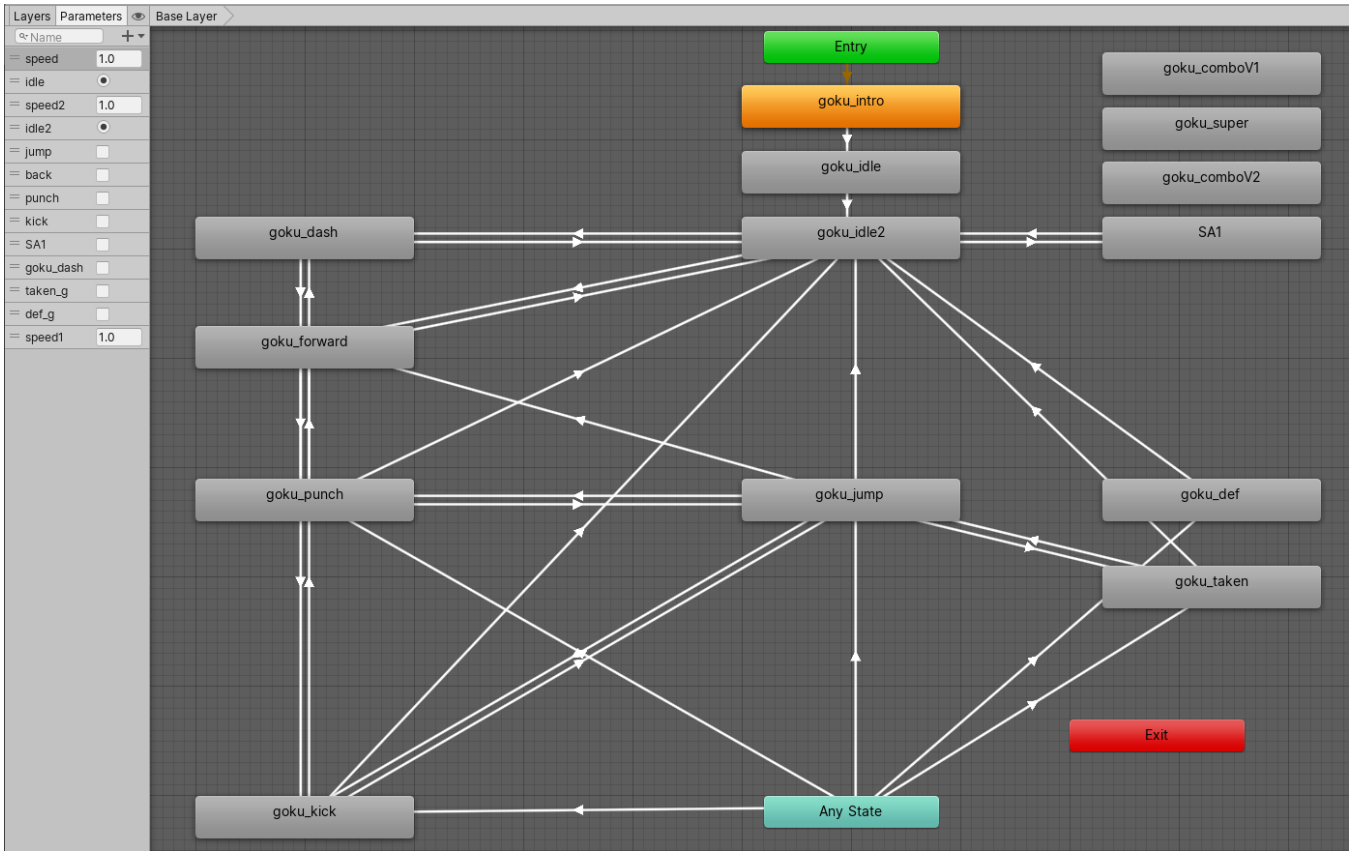

Assets > BEGINING > characters > goku > damage >
0
1
2
3
4
5
6

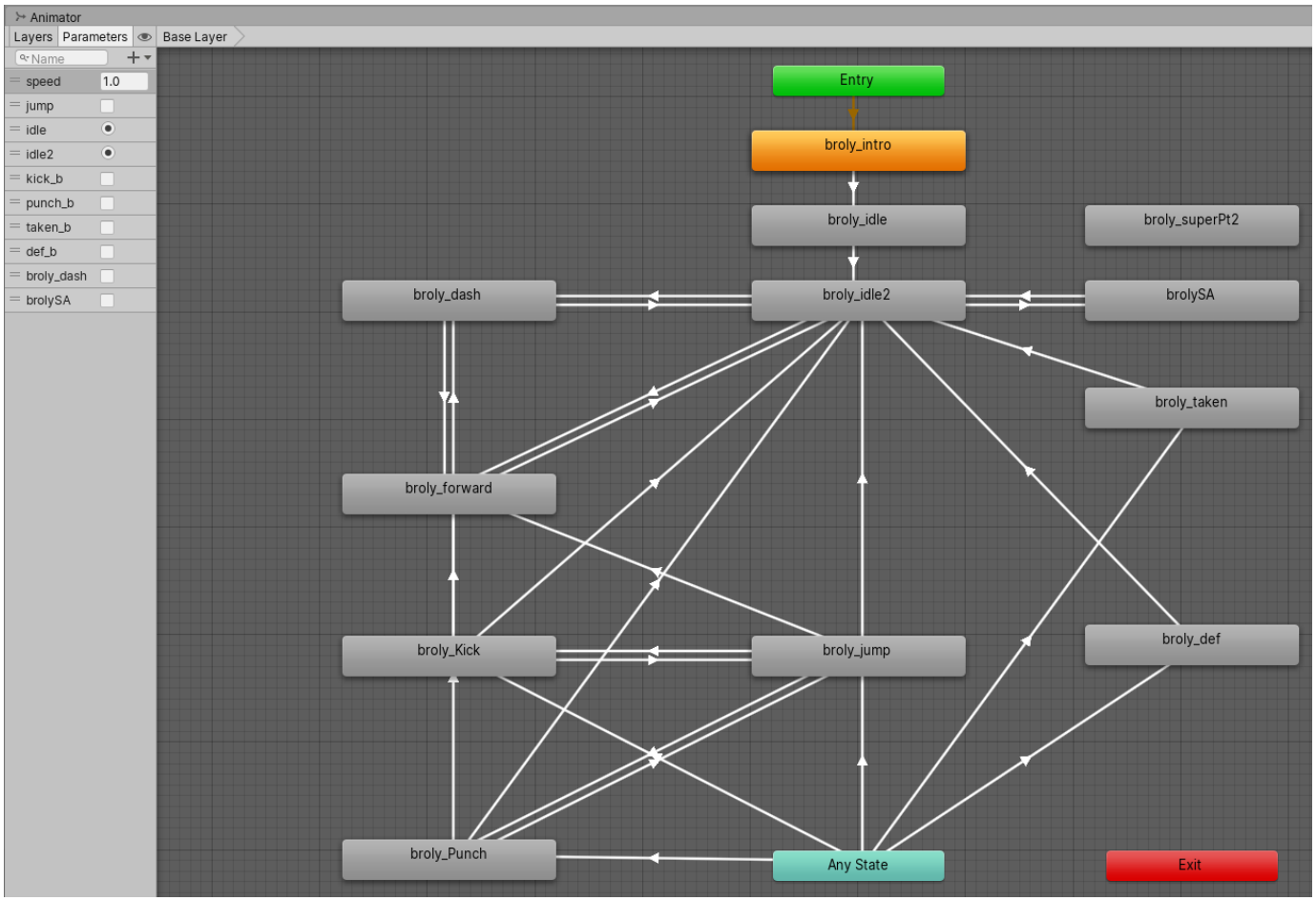

Assets > BEGINING > characters > goku > intro >
0
1
2
3
4
5
6


Assets > BEGINING > characters > goku > movement >
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


Assets > BEGINING > characters > goku > stand >
0
1
2
3
4
5
6
7
8
9
10
11


Assets > BEGINING > characters > goku > supers >
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


Layers
Parameters
Base Layer
Name
speed 1.0
idle
speed2 1.0
idle2
jump
back
punch
kick
SA1
goku_dash
taken_g
def_g
speed1 1.0
Entry
goku_intro
goku_idle
goku_idle2
goku_comboV1
goku_super
goku_comboV2
SA1
goku_dash
goku_forward
goku_punch
goku_jump
goku_def
goku_taken
Exit
goku_kick
Any State


Animator
Layers
Parameters
Base Layer
Name
speed 1.0
jump
idle
idle2
kick_b
punch_b
taken_b
def_b
broly_dash
brolySA
Entry
broly_intro
broly_idle
broly_idle2
broly_superPt2
broly_dash
brolySA
broly_taken
broly_forward
broly_Kick
broly_jump
broly_def
broly_Punch
Any State
Exit