Skill Issue

Project Documentation

Student Name: Juan Franco Lagar

Student ID: 38218784

Supervisors: Prof. Stephan Günzel and BA Ignacio Rubio

Semester: WS22/23

Game Design

Abstract

The fighting game genre is notoriously known as hard to learn and even harder to master, even inaccessible for most players due the high skill level the games demand of their players. The objective of this project is to provide new players with an entry point to traditional fighting games that does not require a lot of technical skill to be able to play. The project should have most of the core elements of a fighting game for players to interact with, and simplifying some of the most demanding elements for more accessibility. The knowledge from this bachelor project’s thesis is used as the base for this project’s design elements.

Table of Contents

Introduction

The Skill Issue project is a three button traditional 2D fighting game, where two characters fight each other until one depletes the others health or the time runs out. The game can be played against another player or against a CPU using one of four variations of a single base character. These variations are fire, water, wind and earth, the variations are simple versions of popular fighting game character archetypes.

The game uses three buttons for attacking, hence why it’s referred as a three button game. The three attacking buttons are the “*Light*”, “*Heavy*” and “*Special*” buttons. The Special button is different for every element in order to accommodate for the previously mention archetypes, meanwhile the Light and Heavy buttons are shared between variations as they have more general applications.

In addition to the three attack buttons the characters can move around the screen using four directional buttons. The left and right buttons function for horizontal movement, the up button for jumping which can be combined with horizontal movement for diagonal jumps and the down button to crouch. If a player presses the horizontal button in the opposite direction as the enemy would allow them to block incoming attacks.

Development

Personal Remarks

I chose to work on a fighting game for my bachelor project from very early on, since it's the game genre that really got me into game development. I think that fighting games are a really fun and rewarding once you get into them but I also understand that they are not for everyone. Regardless of that I want to share my passion for these games with as many people as possible.

This is a very personal project because with it I wanted to lay a foundation for other fighting games I want to develop on my own in the future. For example, the project structure and base scripts can modified to suit different kinds of fighting games from traditional games like "*Street Fighter*" to anime games like "*BlazBlue*". I plan to use the things I created for this project and the knowledge I gained developing it to create more games.

Concept

The original concept for the game was a game 2D pixel-art game with as few of the entry walls discussed in the thesis, the biggest one to avoid being long combos and complex inputs. As well a simple enough characters for players to understand without issues what is going on on the screen. The game combat would mostly be done in the ground where player can do combos and special attacks, as well as block; while jumping players are left vulnerable in air since they cannot block in the air, this turns jumping into a risky decision to close or create space fast, hit crouching opponents and jumping over projectile.

To avoid having just one character, which can end up being boring, I decided to give the base character some variations. These variations would also introduce players to common character archetypes or play-styles. The differences between variations is that their Special attacks would have different properties, but they would have the same base animation but different special effects. For example, every character has an uppercut animation but the fire element special attack is a jumping fire uppercut meanwhile the earth element special is the same uppercut animation but a projectile appears instead.

To add flavor to the variations each one would be given a different element with the elements of fire, water, wind and earth. Water and fire would be the most balanced variations as they represent the "*shoto*" archetype as they have very similar skills in the form of a moving forward kick, a projectile attack and an uppercut attack, the main difference is that water's forward kick has a longer reach and fire's uppercut attack is a jumping attack with more vertical range. Earth and wind are two opposite elements so they represent two opposite play-styles, the "*grappler*" and "*zoner*", earth specials are short ranged and instead of a projectile it has a grab that deals more damage than average; wind on the other hand has the fastest projectiles and instead of a forward moving kick the kick moves backwards and creates an upwards diagonal projectile.

Scratched ideas for the project was a “story mode” where the player would do tasks asked by a spirit of their chosen element and fight the fighters chosen by opposite elements spirit.. These tasks are meant to replace the traditional tutorial mode of other games and offer the player a safe space where they can learn about the game and genre, for each tasks the player gets limited attack options and they have to beat an opponent with the same options as them or having to develop a certain strategy to beat an opponent. After successfully doing a task the player unlock a special move becoming “stronger”. This idea was scratch early on mostly due to the complexity of programming the adequate AI opponent for each tasks. Another big idea that tied in to this story mode is to allow players to customize which specials attacks their character would have, for example the fire element uppercut and wind projectiles, but this idea added a layer of complexity the game didn’t need and went against the main purpose of the elemental variations. As well of customizing the character appearance but this didn’t make it due to the need for several different art assets.

Early Development

I started working on the project during the summer holidays. I already knew I wanted to do a fighting game and that they take a really long time to make, so I decided to do some preparation before properly starting the project. The first thing I did was create the git repository alongside the “*aseprite*” file to draw the sprites on. I decided to work first with assets creating since at the time I didn’t really know exactly how I wanted to approach coding the game first but I had a rough idea of the amount of animations I would need.

The first sprites I drew were simple sprites with the intention to be used for testing rather than for the final game, the first animation done was the idle animation just to get a feel for the proportions of the character then I wanted to do the forward and backwards walk cycle, for this animations I used as reference the walk cycles for Kyo Kusanagi from “*The King of Fighters 2002*”. When I finished drawing the walk cycle I created a list with the number of animations that single character would need in a text file called “*BATasks*”.

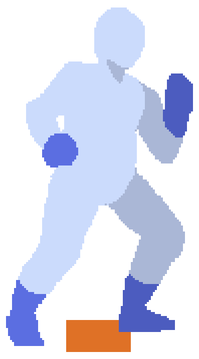


Figure 1: Test version of the character

With the first animations done I took a break from art and focused on programming, so I created the 2D Unity Project. Then when the project was created I created the first scripts which were the core scripts for the game. The first one is based on the design pattern: the command pattern, which allows the game to turn key presses into actions in the game. The command pattern binds one or more actions to a command class which then can be easily changed or reference in code. The main reason why I used the command pattern was to be able to reduce the amount of code I had to change when wanting to use several buttons to do the same thing, for example instead of performing a punch when a button is pressed is easier to ask the character to perform it when a command is execute through button press or through another script; this approach made creating a functioning AI way easier since the AI would just send commands to the character's Input Handler script instead of simulating button presses.

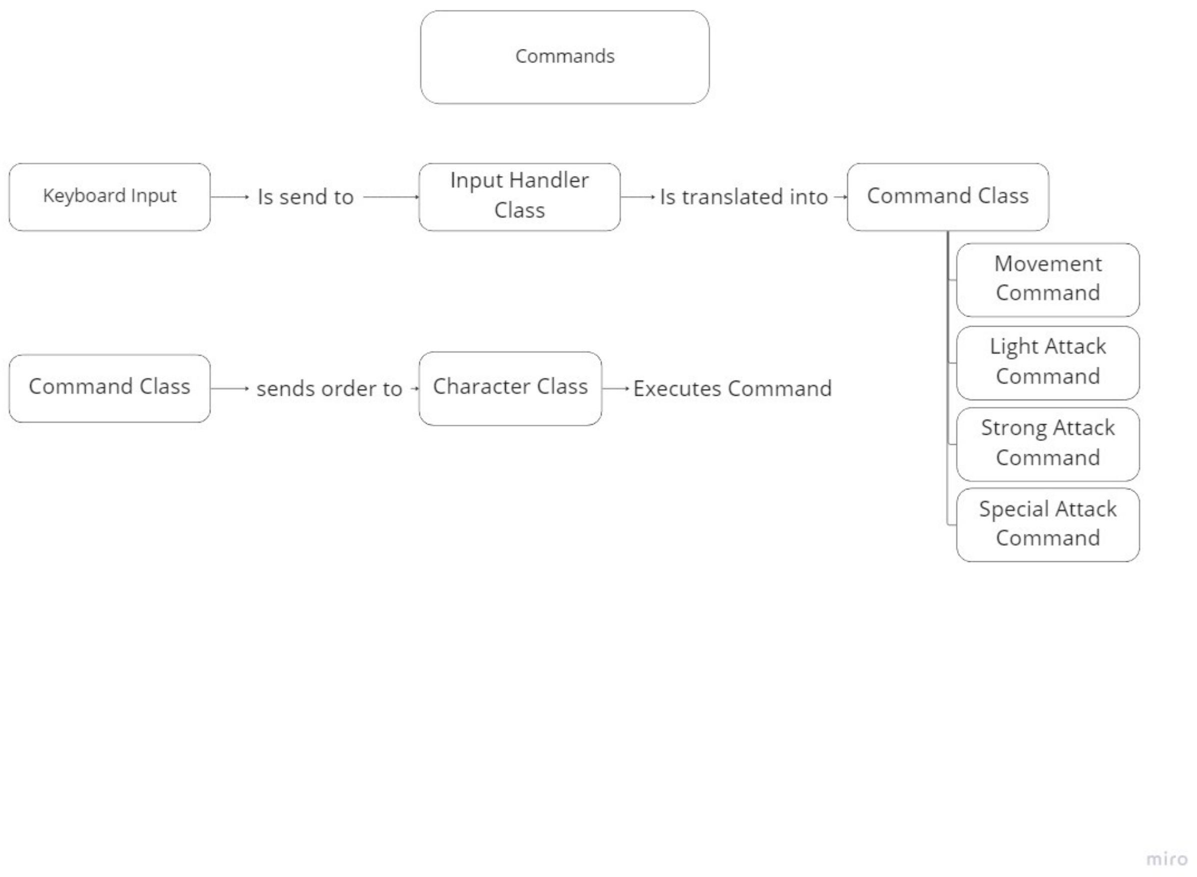


Figure 2: Commands workflow diagram

The above image illustrates how the command system roughly works, when a player presses a button the button information gets send to the Input Handler class where it gets translated into one of four Command classes with each one having different instructions for the character. For example when the J key is pressed the Input Handler translates it into a Light Attack Command which tells the player to perform a punch (Light Attack).

The second script I worked on was the State Machine script. A State Machine its the class that switches between the different states classes and a set of smaller states called Action States that instead of being a whole class they are just an Enum. Characters have different states depending on what they are doing in the screen. I created the three main States called Standing State, Crouching State and Jumping state both derivative from the base class State. The smaller Action States are: None, Attacking, Blocking, Hit and Landing.

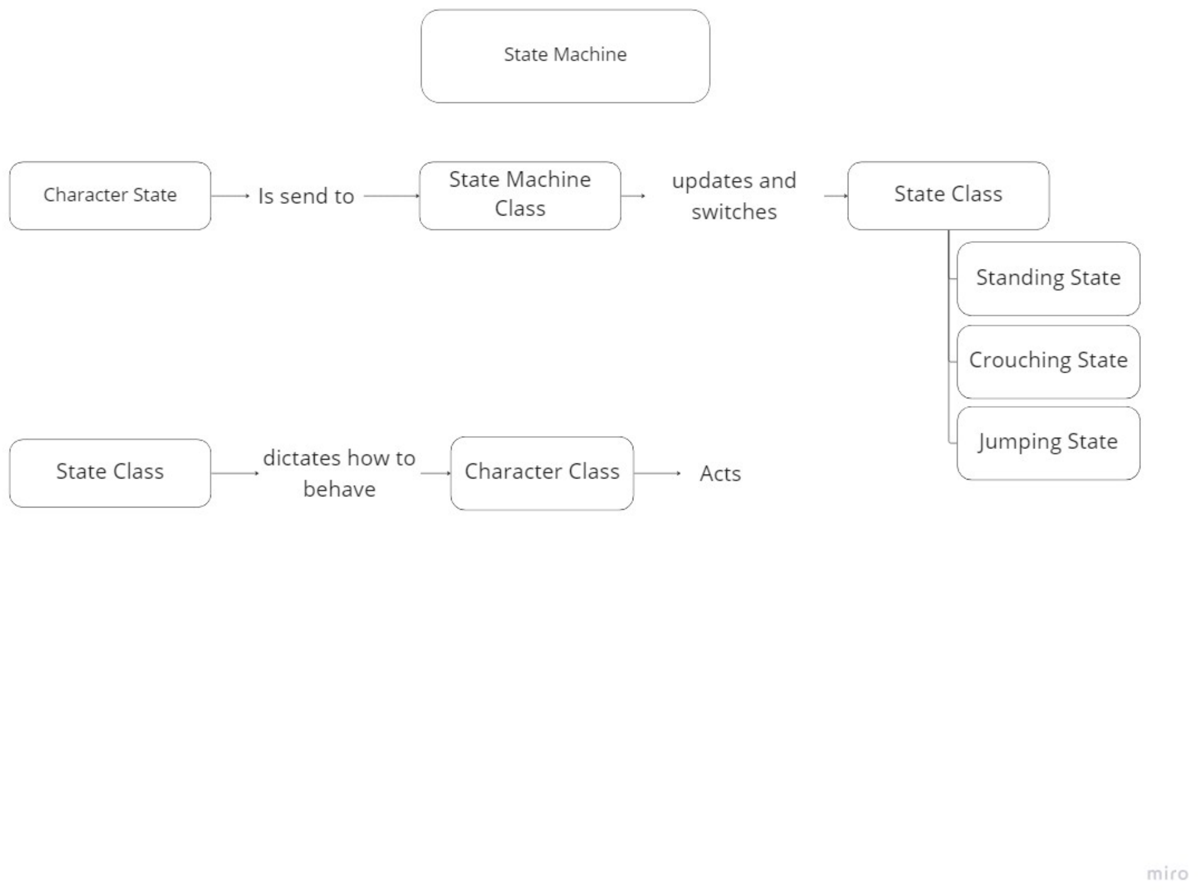


Figure 3: State Machine workflow diagram

This diagram shows how the State Machine works. Alongside the commands the state machine allows the character to perform the adequate action, for example the state machine prevents the character from acting if it’s currently getting hit or allows them to jump if they are grounded.

And the third script I worked on was the main script: the Character class. The Character class contains all the information and actions of each character. The class needs information from the Input Handler and State Machine for acting but it handles all the action logic on it’s own. Alongside the Character class I created a Scripteable Object called Attack Data that is used to store all the information belonging from one attack, including damage and attack animation.

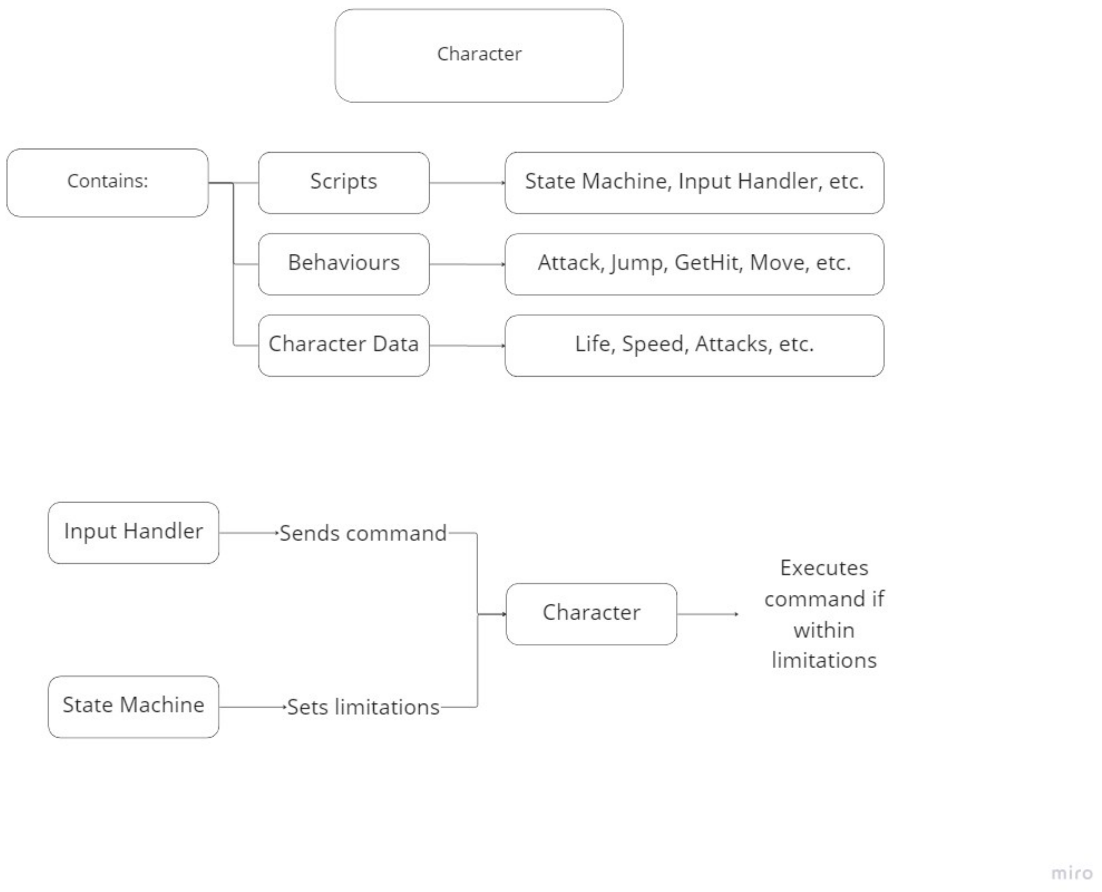

Character
Contains:
Scripts
State Machine, Input Handler, etc.
Behaviours
Attack, Jump, GetHit, Move, etc.
Character Data
Life, Speed, Attacks, etc.
Input Handler
Sends command
Character
Executes command if within limitations
State Machine
Sets limitations
miro

Figure 4: Character workflow diagram

This diagram displays everything that the Character contains and how it interacts with the Input Handler and State Machine.

The others scripts I work on where small custom physics scripts to avoid using Unity's physics since I wanted to create the game in a way that wasn't to engine dependant. The physics system mostly uses custom made collision boxes to check where the character is and what's around it: the enemy character, the screen limits and floor. Each character has three kinds of collision boxes: a Pushbox that handles collisions with the floor, walls and other characters; Hitboxes that determines the effective range of each attack and Hurtboxes that determines where the character can be hurt.

I keep track of my programming progress in a document called "Journal". I created this document to prevent me from getting lost and forget what purpose each script had since I was not going to come back to the project until finishing my thesis.

Core Elements

In the thesis I wrote about the many walls new players face when introduced to the genre, specially long combos, complex systems and a lot of different inputs. At this point the core design elements for the game were set:

- **Game has limited movement options:** This is to keep the game grounded and less chaotic since multiple movement options can be hard to keep track of.
- **Game has three attack buttons: Light, Heavy and Special button.** This serves two purposes, the first one is to make the game easier to play since less buttons means there are less things actions to remember. The second one is to create simple to understand combos, the main combo in this game is: Light attack into a Heavy attack and finishing with Special attack.
- **Game has no motion inputs and limited button combinations**: The special attack removes the need for motion inputs, although in the thesis was argued that motion inputs do not have to be an entry-wall I decided against it since this game didn't need to have them since the character can only have three special attacks. The only button combination in the game is the Light button with the Heavy button which perform a grab attack which the opponent cannot block against.
- **Game has a single base character with four elemental variations:** The main reasoning behind this is because having a single character reduces the amount of art assets that needed to be created and I felt that having variations is the least I could do to avoid making the game boring by only having one character. It also works well in introducing players to different play-styles.

From here onward I repurposed the BATasks document to track down my current progress, write down future milestones and more importantly keep track of game bugs and feedback from my supervisors and play-testers.

Once I decided on this I had to remake the Character sprite to something more finished, while still having in mind that this was a sole developer project. The final model is a simple humanoid character with two eyes. And since walking a jumping features where already tested with the previous model I added to the new one light attack animations, one for each state (Standing, Crouching and Jumping states) and blocking and getting hit reactions.

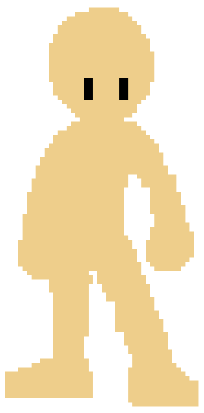
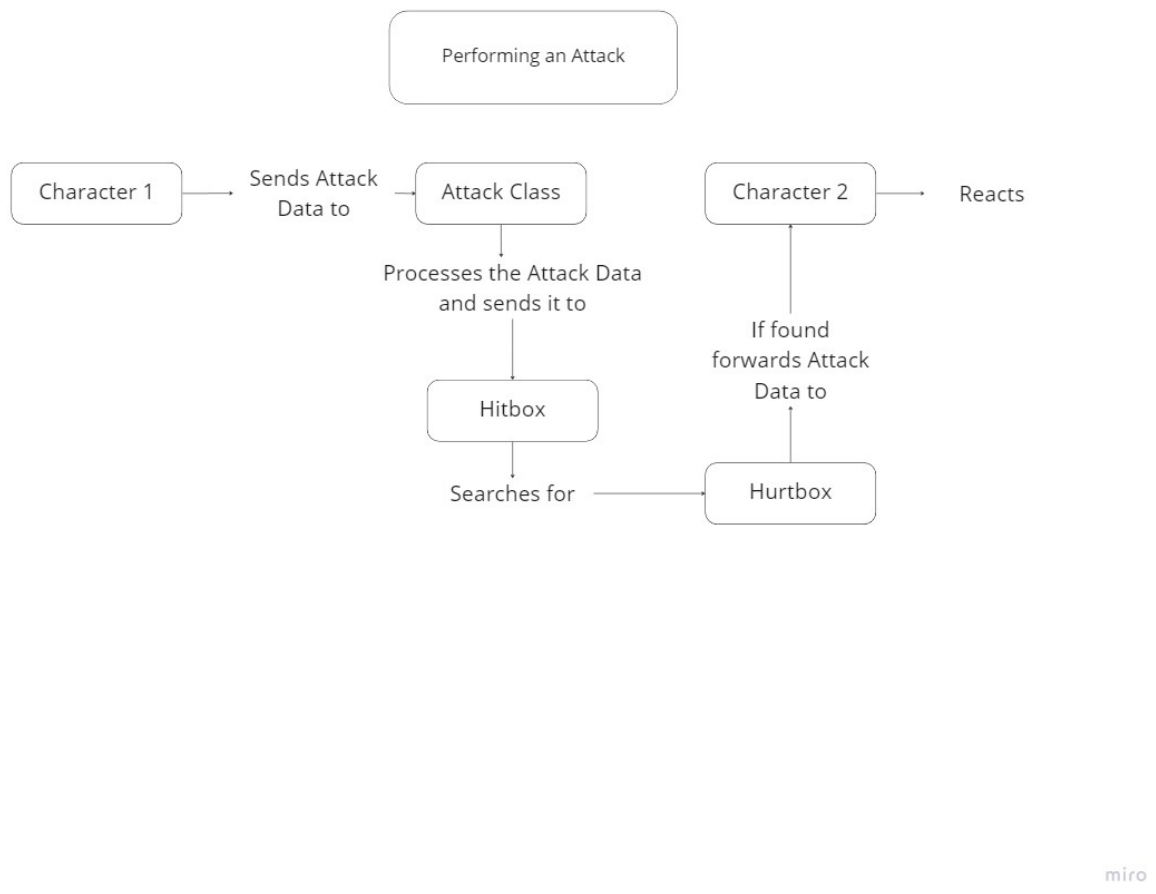


Figure 5: Final version of the in-game character, upscaled

Then I implemented the attack logic and reactions using a class called Attack class which handles all the logic behind any attack, this includes playing the attack animation, handling Hitboxes and Hurtboxes and telling the character getting attack how to react if they are hit.

The following diagram illustrates the logic behind any attack.



Next, I worked with the Game UI displaying a timer and the remaining life of both players and Main Menu system for builds. I had to put a limit at the game's FPS to 60 FPS to make the game run the same across different PCs since animation speeds have to always be the same for any fighting game.

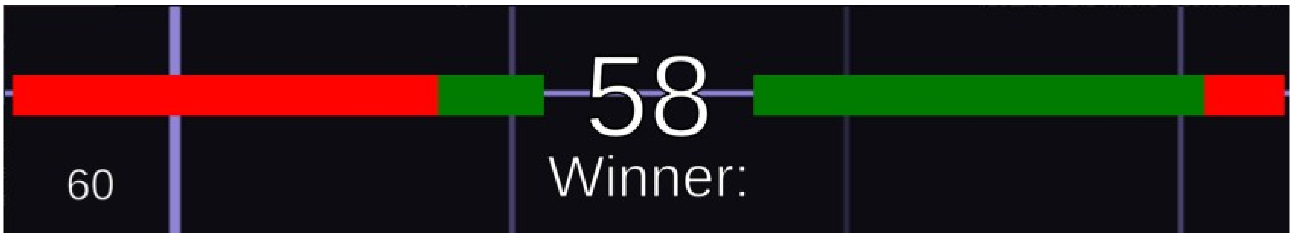


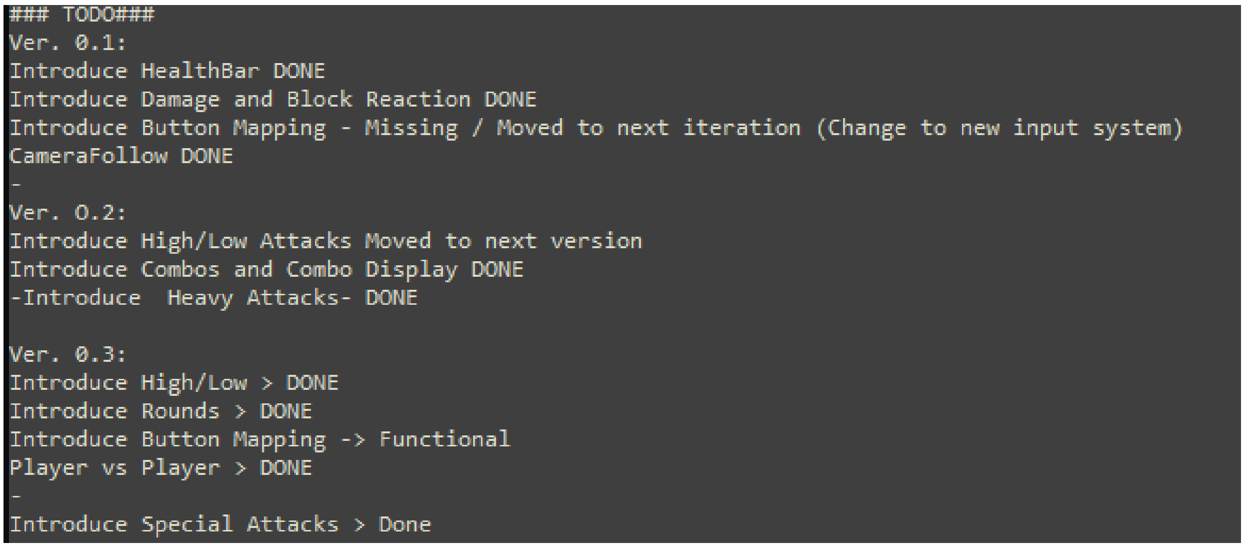
Figure 6: In-game screenshot of game UI elements

The last missing system was a camera system that had to fulfill two conditions: it always needed to be centered between both players and it had to prevent players from walking away from the camera edges. The second condition was fulfill by placing an invisible check behind every character that would notify the Character class if they were ever outside the camera limits, making the Character simulate that they were walking towards a wall.

Iterations

With all the systems in place all that was left was doing weekly iterations where the workflow would be the following:

1. **Updating BATasks doc**: Review feedback and write down bugs from the previous versions.
2. **Bug fixing and implementing feedback:** Marking solved issues with as fixed or implemented.
3. **Working on next version milestones:** Expanding on the core elements and adding missing features.
4. **Preparing a new version**: Creating a build and have people play test it.
5. **Updating BATasks doc:** Creating and reviewing upcoming milestones.



```
### TODO###
Ver. 0.1:
Introduce HealthBar DONE
Introduce Damage and Block Reaction DONE
Introduce Button Mapping - Missing / Moved to next iteration (Change to new input system)
CameraFollow DONE
-
Ver. O.2:
Introduce High/Low Attacks Moved to next version
Introduce Combos and Combo Display DONE
-Introduce  Heavy Attacks- DONE

Ver. 0.3:
Introduce High/Low > DONE
Introduce Rounds > DONE
Introduce Button Mapping -> Functional
Player vs Player > DONE
-
Introduce Special Attacks > Done
```

Figure 7: Screenshot of BATasks.txt

The following elements were develop during the iteration process:

- **Character class further reactions to attacks:** Now briefly stops the getting hit animation for a while for more visual clarity, as well as getting pushed away from the corner instead of their opponent if they are hitting an opponent since the opponent should not get pushed into the wall. Some special attacks make the character move towards certain directions.
- **Creation of an AI for CPU controller opponent:** This AI would perform certain actions depending on the distance between it and the player.

- **Migration from tradition Unity Input System to the new Unity Input System:** This change came to be when trying to add controller support and multiplayer support to game but the system itself is too complicated that controller support was scratched since it was becoming too time consuming. The migration itself was really simple thanks to the Command pattern.
- **Creation of combos and implementation of Heavy and Special Attacks:** The Attack class now checks if an attack connected and if the player is trying to do new attack, if the conditions are met then the character will cancel the previous attack mid animation to start the new one. This is also reflected in the game UI as it displays how many hits were chained together.
- **Creation of Score Manager and character select:** The Score Manager class keeps track of the how many rounds a player has won and carries other information between game scenes like which element the player has selected in the character select screen.

Finishing the game

Due to the iterative nature of the project finishing the game was mostly polishing the game since the game itself was already finished and functional. The last changes made to the game where making the elemental variants feel unique, adding music and sound effects to the game and giving the game a more aesthetic look with consistent game art.

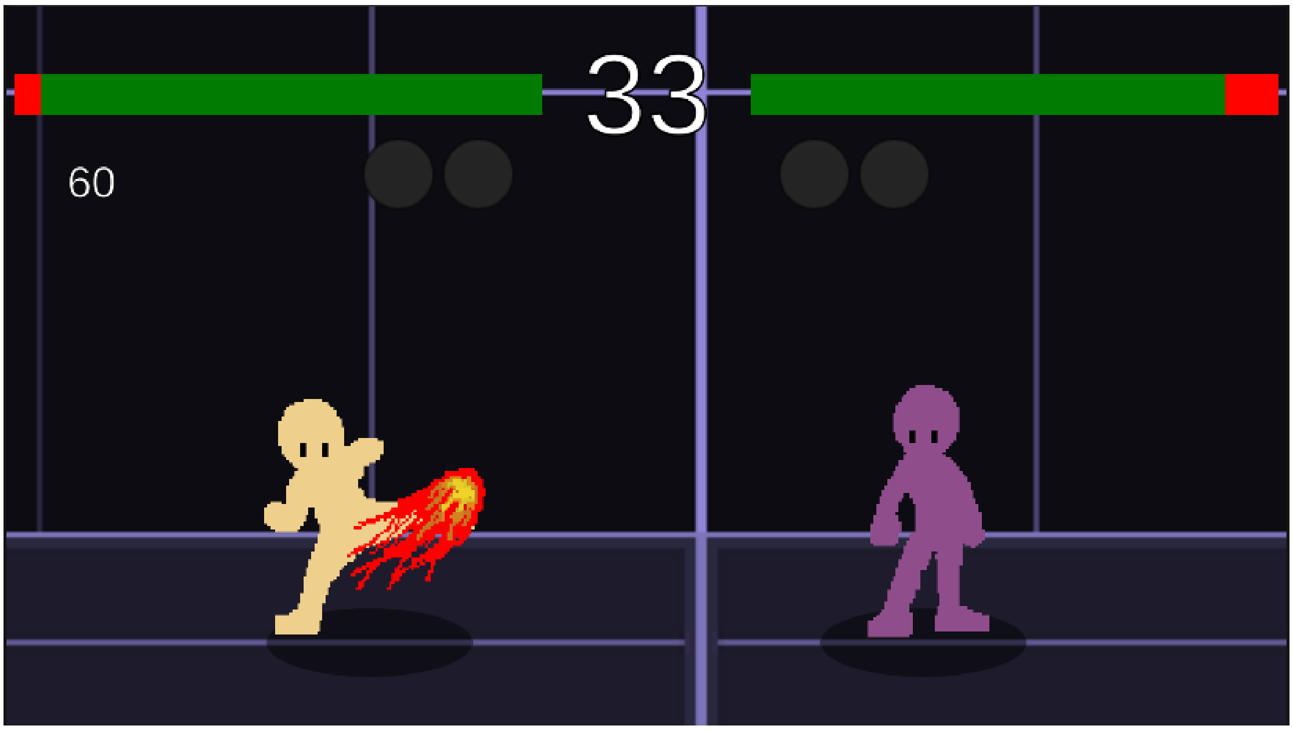


Figure 8: In-game screenshot of Skill Issue

Conclusion

I am really grateful I could work on a Fighting game for my bachelor thesis and I am extremely proud of what I have accomplished here. Honestly, I am a bit surprised I managed to get a functional game without any kind of problems or having to crunch. Having clear goals from the beginning and a very successful workflow did wonders for this project. There were very few problems during the development, the only relevant one was trying to have controller support and button binding but the Unity new Input System was so complex I needed a lot of extra time and preparation for it. I would have loved to implement at least a bit of the scrapped ideas but I decided that the game could stand alone without those ideas so instead I focused on polishing existing systems rather than creating new ones that might or might not work. As for the future of the project, Skill Issue would probably not receive any updates after the submission of this project but I would carry on the ideas, concept and some of the scripts and assets of the game to a new personal project.

Software Used

Unity Engine

Git Hub Client

Visual Studio Community

Aseprite

Clip Studio PAINT

Audacity

University of Applied Sciences Europe
Iserlohn · Berlin · Hamburg

EIGENSTÄNDIGKEITSERKLÄRUNG / STATEMENT OF AUTHORSHIP

Name | Family Name	Vorname | First Name

Matrikelnummer | Student ID Number	Titel der Examsarbeit | Title of Thesis

Ich versichere durch meine Unterschrift, dass ich die hier vorgelegte Arbeit selbstständig verfasst habe. Ich habe mich dazu keiner anderen als der im Anhang verzeichneten Quellen und Hilfsmittel, insbesondere keiner nicht genannten Onlinequellen, bedient. Alles aus den benutzten Quellen wörtlich oder sinngemäß übernommen Teile (gleich ob Textstellen, bildliche Darstellungen usw.) sind als solche einzeln kenntlich gemacht.

Die vorliegende Arbeit ist bislang keiner anderen Prüfungsbeh.rde vorgelegt worden. Sie war weder in gleicher noch in ähnlicher Weise Bestandteil einer Prüfungsleistung im bisherigen Studienverlauf und ist auch noch nicht publiziert.
Die als Druckschrift eingereichte Fassung der Arbeit ist in allen Teilen identisch mit der zeitgleich auf einem elektronischen Speichermedium eingereichten Fassung.

With my signature, I confirm to be the sole author of the thesis presented. Where the work of others has been consulted, this is duly acknowledged in the thesis' bibliography. All verbatim or referential use of the sources named in the bibliography has been specifically indicated in the text.

The thesis at hand has not been presented to another examination board. It has not been part of an assignment over my course of studies and has not been published. The paper version of this thesis is identical to the digital version handed in.

Datum, Ort | Date, Place	Unterschrift | Signature