Project Download – **Art Assets** – **Repository** – Video Demonstration

Beat'em Squad

NARRATIVE ECONOMY AND GAME FEEL IN VIDEOGAMES

UNIVERSITAT JAUME I

David Arona Bueno

Advisor: Vicent Cholvi Juan

This project is submitted for the bachelor's degree of
Video Game Design and Development

INDEX

1. TECHNICAL PROPOSAL

In this section I'll be detailing the design and development of the project itself, from how did it took form, to why. The schedule followed, the tools used and the expected results will also be discussed here.

1.1 INTRODUCTION

Throughout the degree I have done many games for the subjects. Although the games met the goals requested, they lacked interest in the more interactive side. I want to take this opportunity to make a game that is fun and feels more or less complete. Short, but polished and replayable.

Another area I'd like to explore is character design, not through the use of heavy dialogue and exposition, but through how they move, how they sound and how they behave. The ways to play to each character as well as their relation with the game's systems will give the player information on their personalities and traits.

Most of the workload of the project will be on making the animations and coding the infrastructure, so it can support the multiple situations derived from the characters' repertoire of abilities and its relationships with the environment. Both making the game feel responsive and making the code easy to modify and modular are the highest priorities in the technical section.

1.2 RELATED SUBJECTS

- VJ1222 - VIDEO GAME CONCEPTUAL DESIGN

The basic design techniques will be heavily used throughout all the projects, but even more valuable to my idea of the final product are the game analysis and discussion presented in this subject.

- VJ1227 - GAME ENGINES

As I'll be using Unity for the project, the knowledge of the different tools provided by this subject will be important.

- VJ1236 - SOUND PRODUCTION TECHNIQUES

Sound design will probably be important to one of my main objectives (game feel) but more than anything I value the discussion and tricks of the trade learned here.

- VJ1218 - HYPER MEDIA NARRATIVE AND VIDEO GAMES ANALYSIS

Narrative economy conveyed through art and gameplay mechanics is quintessential for my goals.

- VJ1215 - ALGORITHMS AND DATA STRUCTURES

Efficiency and modular code will be things that I'll prioritize in the technical side of the project, both things I learned the importance of in this subject.

1.3 MAJOR GOALS

- NARRATIVE VALUE THROUGH ART AND INTERACTION
- IMPROVE AND LEARN ABOUT GAME FEEL AND ITS KINESTHETIC VALUE
- Offer replayability through the characters' varied movesets and overall different playstyles.
- Taking special care of the animation department to better develop the cast of characters.
- Offer a polished, playable and enjoyable product, made entirely with hand-made resources.

1.4 PROJECT SCHEDULE

Divisions are made by tasks and respective work hours. Each section's final product should result in a substantial approach to the project's goals. Each section assumes that the main tasks of previous sections have been completed.

From the Core Game Planning onwards **{1.4.2}**, total hours estimated correspond to the planning of 6 hours a day, excluding weekends.

Underlined text for tasks vital for the acquisition of objectives noted before.

Cursive text for tasks that, while additive to the overall quality of the product, aren't essential. These tasks don't have estimated durations, as their accomplishment depends fully on the other tasks and thus are pretty hard to predict.

1.4.1 PRE-DEVELOPMENT DESIGN

Mainly design workload done before starting the project itself. Character base sprites and certain animations are also done in this phase.

Table 1 Pre-Development Design Planning

Pre-development Design	
Tasks	**Estimated Duration in hours**
Playable Characters Design	10 hours
Playable Character Concept Art	10 hours
Main Core Mechanics Design	10 hours
Total	**30 hours**

1.4.2 CORE GAME

When this phase finishes, the game has to be in a working state. There won't be enemies or background, but the movement mechanics and attack controls will be fully functioning.

Table 2 Core Game Planning

Pre-development Design	
Tasks	**Estimated Duration in hours**
Playable Characters Design and Implementation	10 hours
Core Mechanics Design and Implementation	85 hours
Sprite Sheets for the 4 Playable Characters	140 hours
Stage, UI and NPC mock-ups	5 hours
Stage, UI and NPC Design	depending
Extended Sprite Sheets	depending
Total	**240 hours**

1.4.3 ENEMIES

In this phase I'll focus mainly on the enemies that the player will confront, especially in the A.I. department of them.

Table 3 Enemies Planning

Enemies	
Tasks	**Estimated Duration in hours**
Enemy Design and Implementation	50 hours
Enemy Artificial Intelligence	70 hours
Boss Design and Implementation	depending
Total	**120 hours**

1.4.4 ENVIRONMENT AND OTHER NPCs

During this phase, I'll work on elements of the Game World, like the environment, background art and bystanders.

Table 4 Environment and other NPCs Planning

Environment and other NPCs	
Tasks	**Estimated Duration in hours**
Civilian Design and Implementation	10 hours
Civilian System Implementation	40 hours
Stage Design and Implementation	70 hours
Encounter Design	depending
Total	**120 hours**

1.4.5 OVERALL IMPROVEMENTS

In this phase I'll try to finish tasks that have been left out of the other phases. Here I'll also try to make general improvements to the core game by tweaking and playtesting.

Table 5 Overall Improvements Planning

Environment and other NPCs	
Tasks	**Estimated Duration in hours**
Unfinished previous tasks	60 hours
UI Design and Implementation	20 hours
Kinesthetic Improvements	40 hours
Playtesting	60 hours
Effects System	depending
Cinematic Introduction	depending
Tutorial Level	depending
Total	**180 hours**

1.5 TOOLS

This section details the tools used in the project and the benefits on their use to further project's goals.

1.5.1 PROGRAMMING

- **Unity**: The game engine that will be used for the project.
- **MonoDevelop**: Supports C# and is standard in Unity.

1.5.2 Game Art

- **Paint.net**: Freeware painting software used for spriting pixel art.
- **Texture Packer**: Compresses individual PNGs to spritesheets usable in Unity.
- **Gifmaker.me**: Platform used to test animations before taking the next step.

1. 6 EXPECTED RESULTS

- Higher understanding of the full process of making a videogame, from the inception and the tinkering of ideas, to the construction of the game's mechanics to the quality control part.
- Better and more fluent capacity to use the tools noted.
- A core game structure that is easily expandable and modifiable.
- Discover faster ways to draw and animate a complex spritesheet.
- Learn how to make a game that feels responsive and gives adequate player feedback
- Narrative flow through scenes, attacks and expected player actions.

2. DESIGN

In this section I'll be explaining how and why I designed every part of the game. I'll begin by summarizing the main concepts in a user-friendly manner **{2.1} {2.2}** and then I'll expand on those concepts **{2.3}**. Two fundamental parts of the design of my game, player characters and game feel, are expanded upon in the next two sections as they're too extensive **{3} {4}.**

2.1 INTRODUCTION

Beat'em Squad is a scrolling 2D action game for the PC that thrusts the player into the role of the zany superhuman agents of the Beat'em Squad in a fight to regain the control of the streets and uphold justice.

2.2 DESCRIPTION

In Beat'em Squad you'll play the part of a group of misfit characters part of a higher organization that upholds law and order in the world. An emergency alert comes from District B10 and the squad moves out to action. As you arrive in the already almost deserted area, a group of thug-like robots approach with unfriendly intentions. Smash, cut, blast or electrocute your way through hordes of distinct enemies and save as many civilians as you can.

As Captain Summers, the nervous head of operations with a heart of gold, punch and shield yourself from enemy attacks. From his tactical mind, Summers strong but slow attacks rewards foresight and planning. Block enemy attacks within a certain timeframe and blow the enemy away.

As The Wonder, an enthusiastic new recruit with a past as a vigilante, zap your way through the battleground with your dazzling speed electrical attacks. Specialized in speed and area of effect attacks, though lacking in damage The Wonder excels in crowd control.

As X-42, an ex-convict with a thirst for blood, annihilate every enemy with your superior firepower and range. X-42 can barely contain the explosive power within him, so you'll have to stay aggressive to not overcharge and turn his overwhelming power against him.

As Ronin, a man with the heart of a samurai, cut through swats of enemies with your highly trained sword-wielding skills. Ronin changes movesets depending on which way he's looking at, so only the most experienced players can unlock his whole repertoire's potential.

2.3 GAME MECHANICS AND CONTROLS

2.3.1 USER DEPENDENT MECHANICS

In this section I'll explain the main game mechanics that the player can execute, and its default mapped controls:

CHARACTER MOVEMENT

The character moves around in the 3D space.

- **Move Controls ("WASD" Keys in keyboard)**

Use to move around the map, double press and hold the button of any direction to start dashing in that direction. While dashing, your character will move faster, but once you stop pressing the button, the character goes through an animation of recovery before returning to Idle State.

- **Jumping ("Spacebar" Key in keyboard)**

Use to move your character upwards, which you can use to move around the map minimizing danger. This puts your character in Jumping State. Press the Jumping button shortly to a short jump. Hold the button to jump higher and longer. Press the Basic Attack while in Jumping State to perform a Jump Attack.

ATTACK MOVESET

Your character inflicts damage on the enemy. Each attack will vary on damage, speed, range and effect. To organize the multitude of attacks, I'll classify them by four different parameters: by type (signaling the button you need to press to perform the attack), by input (signaling how you press the button to perform the attack), by effect (signaling how to attack effects enemies and the world space) and by weapon (determines effects and K.O. status). These classifications are not mutually exclusive (see *Figure 1*):

By Type

The attacks by type are standardized in every character (with minor exceptions). How do you chain these is the main core of the gameplay loop.

- **Basic Attack ("O" Key in keyboard):**

Perform the basic attacks of each character. Use in succession combined with Kinergy Attacks or Basic Attacks to form different combos. This type of attack can damage the Enemy, but cannot K.O. them. Usually, there are two completely different Basic Attacks by character (BA being a press of the maped basic attack button):

- **Kinergy Attack ("P" Key in keyboard)**

Perform special Kinergy attacks distinct in each character that use up the Kinergy meter. Use in succesion combined with Basic Attacks to form different combos. This type can Damage and K.O. the Enemy. Usually, there are up three completely different attacks by character (KA being a press of the mapped Kinergy attack button):

By Input

It depends on the timing of the button presses.

- **Standard Attacks**

These attacks only require a simple press of the designated button.

- **Charge Attacks**

Some attacks (either Basic or Kinergy) can be charged to be more powerful or have overall different properties by holding the button in the right time. If you release the button before the charge time is complete, the attack will be weaker. In specific attacks, holding the button can otherwise make a continuous attack to keep going.

- **Barrage Attacks**

Some attacks (either Basic or Kinergy) require you to repeatedly the button to keep attacking. Mash the button enough and you'll unleash a devastating finisher.

By Effect

Aside from damage, timing, range and width, some attacks have particular effects out of the ordinary. Almost all of the attacks of the game can be put down under one of this categories or a combination of these.

- **Standard**

No particular effect to note.

- **Projectile**

The attack makes a projectile appear with its own set of properties, like position relative to the character, travel speed, damage or time until disappearance.

- **No Damage**

The attack has no damage attribute, usually used in combination with other BY EFFECT categories, but not necessarily.

- **Hit Condition**

The attack has different properties or continuity whether you hit an enemy with it or not.

- **Moving Attack**

The attack applies certain speed to the character using it (in the X-axis). This does NOT depend on user input, the distance traveled by the character is predetermined.

- **Moving while Attacking**

The movement of the character (in the XZ plane) is unlocked, so you can move around with the Move Controls while the attack is ongoing.

- **Parry**

The attack changes its properties if you attack the character using it while the attack is ongoing.

- **Insta-kill**

These attacks instantaneously finish off the Enemy, not mattering its amount of Health Bar left.

By Weapon

- **Physical**

Standard type, pertains to punches, kicks or blunt weapons.

- **Slashing**

Edged weapons, such as a sword.

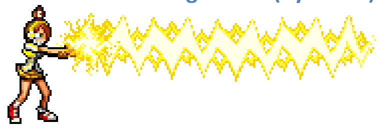
- **Electrical**

By electrical attacks, they leave an electrical charge in the enemy every time they hit.

- **Others**

Explosion, bullets, and other one-offs.

Figure 1 – As an example, this attack classifies as Kinergy (by Type), Standard (by Input), Hit Condition and Moving Attack (by Effect) and Electrical (by Weapon)



UI AND OTHERS

Actions influenced by UI elements appearing on screen (prompts).

- **Change Character (“Y” Key in keyboard)**

Only usable with a full Kinergy meter. Use the Move Controls to choose the character to change in.

- **Enable Pause Menu (“Escape” key in keyboard)**

Time stops and the Pause Menu appears, from there you can return to the game or go to the Character Select Screen.

- **Mouse Controls**

Used to navigate menus such as the Character Select Screen and the Pause Menu.

2.3.2 NON USER-DEPENDENT MECHANICS

COMBO BASED REPERTOIRE OF ATTACKS

The attacks available to each character are usable through a chain of button presses, that is to say, the game uses a classic combo system in which, the further you go into the combo chain, the stronger and crazier the attacks get (this rewards player's aggressiveness and skill). Basically, we can talk about two and a half stages, in which of them you have the option to use a Basic Attack (BA for short from now on) or a Kinergy Attack (KA for short). A KA always finishes the combo chain, while BA lets you continue to the next "stage". (stage transitions are signaled in game, see ***Figure 3***)

1st Stage, no attacks used, you can use both a BA and a KA

2nd Stage, BA used, you can use both a BA and a KA

3rd Stage, BA used again, you can use a KA.

- BA: First Basic Attack

- BA → BA: Second Basic Attack

- KA: First Kinergy Attack

- BA → KA: Second Kinergy Attack

- BA → BA → KA: Third Kinergy Attack

Figure 2 - Small graphic presentation of how the combo system works

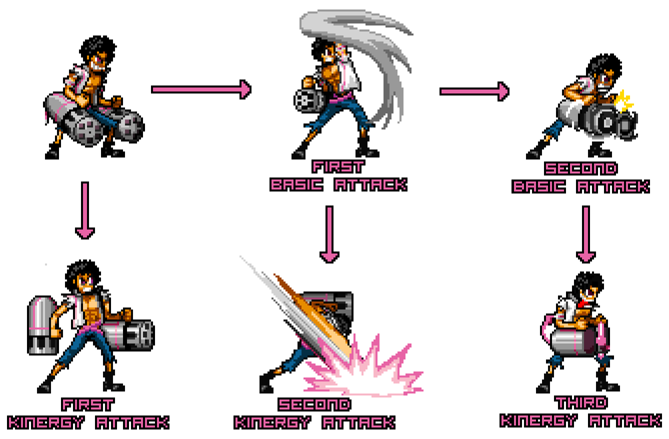
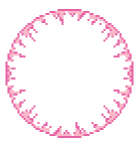


Figure 3 - This SFX signals the player to continue the combo



ENEMIES K.O. STATE

When the Enemy's Health Bar drops to zero, instead of just disappearing an animation will start playing. This means the Enemy is in K.O. State, which is to say that you have to finish him off with a strong attack. Basic Attacks don't finish Enemies in K.O. State, but Kinergy Attacks do. Depending on the attack category by weapon **{2.3.1}**, the animation changes when you finish an enemy. This is not only important in a aesthetical level, but in a mechanic one, as some of the animations can also finish off more Enemies. This is the only exception to the fact that only Kinergy Attacks can finish off Enemies.

BYSTANDERS RESCUE

While playing, you'll encounter some NPCs that aren't Enemies. These are called Bystanders, and it's your duty to save them. To do so, simply move to their position before they're hit with an enemy attack.

KINERGY SYSTEM

Kinergy is the main resource in the game aside from Health. Kinergy Attacks use up a determined amount (each attack amount varies) and Character Change uses it all up. Every character has way different ways to gain Kinergy, which helps differentiate them and define their play style (the specifics of this will be talked about later).

2.4 GAME FLOW

In this section I'll talk about the Game Flow from the moment you start the game to the start of the first enemy encounter. There's a bit more to be talked beyond that but I consider this segment more interesting and representative to analyze.

The game starts in a quasi-empty and badly damaged train, in full speed to its destination. In the background, behind the broken glass of the train's windows, we see what seems to be a city. Immediately, we are introduced to its four main protagonists, sitting on the train next to each other. I want to remark the importance of this scene, especially in contrast with the rest of game, in the portrayal of the world and its characters before continuing with the more technical aspects (see this scene in ***Figure 4***).

This scene serves three narrative purposes:

- **Sets up the tone of the game and its world**

First there is the contrast between the fastness of the background and the stillness of the wagon sets up the scene as the peace before the storm, and the nature of the travelling motif helps build anticipation for the main game. The destroyed, almost in ruins train signifies a world in turmoil, but the vivid palette used deflates the bleakness of the ambient. Between the debris, the four main characters stand out with their heavy outline and saturated colors occupying great part of the left of the frame. Lastly, the indifference of the character to their surrounding cements that this isn't an unusual situation, at least to them.

- **Serves as introduction to the characters**

As said before, the characters stand out as a ray of hope in this desolated environment. Though I'll extensively explain the specific design choices later, one thing I want to note is the use of white in all their palettes and similar patterns of lines in the vests, that unifies them as a group opposed to just a random set of individuals. This will be the first of three (and a variable fourth) introductions to the character's quirks and traits. This is the first time the player sees how the characters look, but some vital parts of their design are omitted for later, to transfer importance to future "introductions" **{4}**. Finally, the order of sitting signifies their relationships with each other.

- **Serves as contrast that can only exist outside of gameplay.**

One priority of this preemptive scene is to show how the characters act when they're out of combat, to offer a little bit more of depth and don't make them such one-note personalities. Though it`s only a simple animation at the beginning of the game, its place within the first of seconds and its recurrence (you'll return to this scene if you lose the game of if you quit) makes it stand out.

In this scene, which I'll call the Character Select Scene in further references, you'll have the option to QUIT, TRAIN or DEPLOY, buttons signaling this options appearing in the lower part of the screen. QUIT will outright close the running program and exit the game, TRAIN or DEPLOY will prompt you to choose a character before sending you into the Training Room Screen or the Stage 1 Screen.

As said, if you press the DEPLOY or TRAIN, a text prompt with an arrow telling to SELECT YOUR HERO will appear. If you hover the cursor over the four characters, two things will appear:

- An animated stat chart above the character, signaling the character's proficiency in that stat with values from 1 to 3. The stats showed are POWER, SPEED, TECHNIQUE, SMARTS and RANGE. The totals values are close to the same, but we find two different distributions, used for the easier and more difficult characters respectively: 3-2-2-1-1 and 3-3-1-1-1
- An animation on the selected character, showing their dominant color and name.

Though I'll discuss font choices later, it's important to note the wording used to convey the characters' role in the history. DEPLOY signals to them not only being in sort of a mission, but also that it is somewhat of an official one and SELECT YOUR HERO makes obvious their nature.

If you picked TRAIN, after you click on the character of your choice, you'll be eased in to factory-like place with a ragdoll in the middle. Here you can practice your attacks without worry. There's a counter that shows how much damage you deal, as well as a button to limit your Kinergy use. Certain attacks do not work as intended if you hit the ragdoll, as they require a functioning enemy.

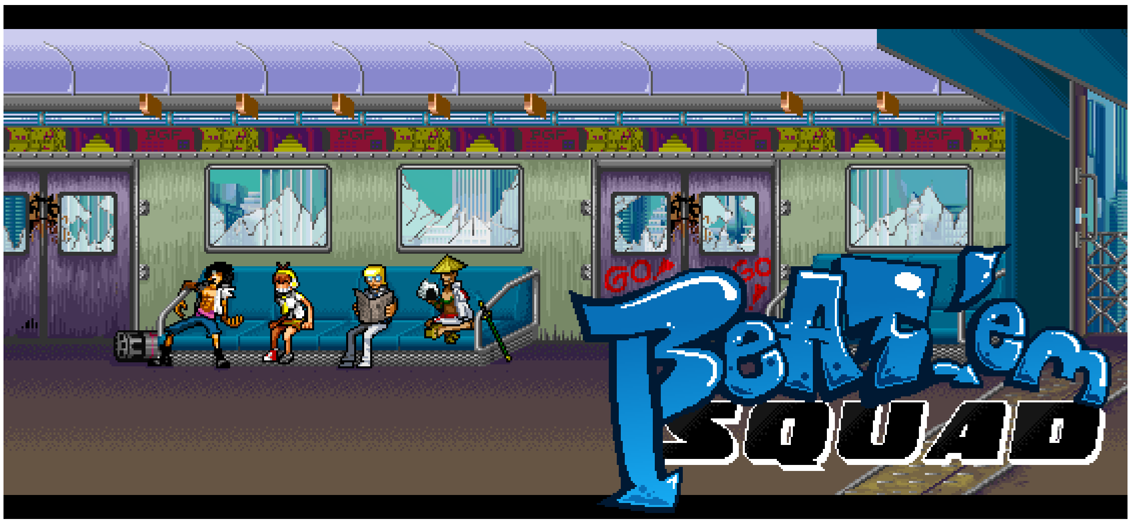
If you picked DEPLOY, a fade with the color of your character choice will transport you into the first stage of the game.

In both cases, you'll be treated to an introduction animation of the character selected before you take control of the character, which I'll detail later {4}. When you take control you'll fast realize that to advance you'll have to go to the left. The game alternates between free roam in the stage and encounters with enemies in which the camera locks and your area of movement is limited to the one showed in screen. I'll expand on the enemy encounter flow later.

Anytime you want while playing, you can press the Pause Button to enter the Pause Menu, which freezes time and lets you choose within two options: RESUME, to continue playing and QUIT, to return to the Character Select Screen. Also, if your Health Bar empties, a Game Over Screen will appear and you'll be transported immediately to the Character Select Screen.

When you clear all the encounters in a stage, after going to the upmost right of said stage, you'll be transported to a new stage, and your Health Bar will be replenished and your Kinergy Bar emptied. After you finish all the stages, you'll be transported to the Character Select Screen.

Figure 4 - Main Menu Screen



2.5 GAME WORLD

In this section, I'll be explaining the various parts that form the game's aesthetic as individual points of discussion, as well as how do they fit into the larger whole. Main characters' role is missing in this section as it's a big part of the project's goals and merits more discussion that'll be talked about later {4}.

2.5.1 BACKGROUND

Set in our world's near future, a strange meteor collided on Earth, causing major natural disasters and sending a wave of unknown radiation across the globe. As the chaos settled, people discovered the effects of the radiation, now known as Kinergy: natural abilities growth as well as the ability to conjure construct and occasionally, other special abilities.

In an attempt to regain order, governments of the world agreed to quarantine kilometers around the point of impact (that would later start to be repopulated) and form a global organization of law-enforcers composed of high performing officers from a variety of fields and countries. This organization (the PGF) would operate above countries' laws.

Years pass and as people continue to use their newly found abilities; the organization grows insufficient to the sheer magnitude of the crime waves hitting the world. To counter this, the PGF lowers its entrance requirements and tries to bolster their ranks with a world-wide recruitment process. The effects of this risky maneuver are yet to be seen, as scientists in every spectrum of the moral compass grow closer to the truth of the meteor's impact…

I want to remark that all of this background is never directly mentioned ingame as choosing to represent it with only environment and character's details is one of the intended goals.

2.5.2 THE PLACE

The place in which the major part of the gameplay action is situated is District B10, one of the outmost districts that surround the previously mentioned point of impact. Riddled with crime and corruption, District B10 has been forgotten by almost all the authorities but our protagonists. People still live here and most of them as much victims of the cycle of violence as part of them. Recently, certain strange robot humanoids attacks have been occurring.

District B10 visuals rely mostly on cold colors in the background, with an ever-looming sea of skyscrapers, and hot, humid colors in the forefront, in the harsh. All around the environment, there's interactive elements and little animations that make the place, such as a man that looks through the door's eyehole when you pass by or rats that scamper away if you get to close to them. Though violent and bleak, the cartoony style of the drawings and colors take away from the harshness of the environment. In Stage 1 and 3 you'll be out in the streets, while in stage 2 you'll be in some sort of storage room with flickering lights.

2.5.3 ENEMIES

The Tulip Band, a group of robotic troublemakers that have made the District B10 their home turf. Their leadership and origins are a mystery but their violent behavior shows that their intentions are surely not benevolent. As the Beat'em Squad, you'll have to stop them before they take full control of the district. The units themselves seem to be powered by Kinergy radiation but how it has come to this is a mystery.

There are three different types of enemies, but their predominant use of the color purple and variations, as well as that their main robot wireframe is basically the same (the clothes and their posture changes drastically) unifies them as a group. They also have more muted colors than the protagonists, cementing them as the opposition.

Model 66 – Slu66er

The Slu66er model is the basic and easiest to beat of the bunch, using a bat enhanced by Kinergy as his main weapon. His attacks are slow and heavily choreographed, giving the player the time to react accordingly. They really aren't a menace alone but can be quite a nuisance in a group, especially paired with another types (see *Figure 5*).

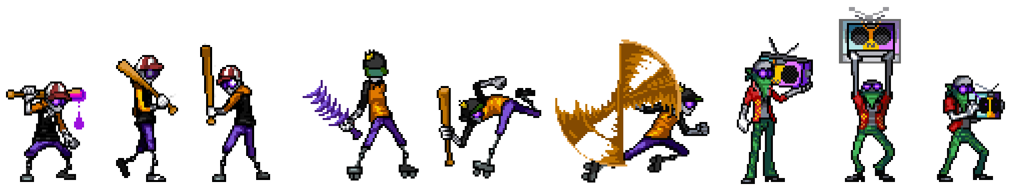
Model 11 – High Ro11er

Similar to the Slu66er but with more mobility and more acrobatic attacks due to their rollerblades. They are truly dangerous by themselves and is recommended to either take them by afar with a range heavy character or approach them and don't let them a moment's rest. Their attacks are way harder to predict, but their aggressiveness combined with their speed makes their trajectory easy to guess (see *Figure 5*).

Model 88 – 8oom 8oxer

Easily the livelier of the bunch, these robots specialize in ranged combat and buffing the other mooks. They're in most of the cases the highest priority target (see *Figure 5*).

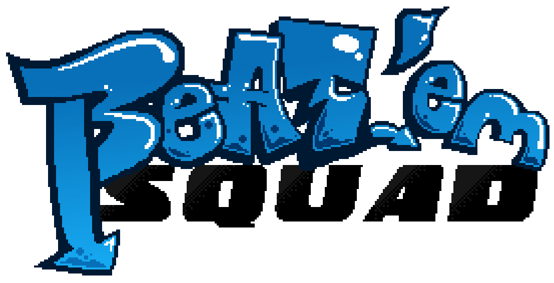
Figure 5 - From left to right, Slu66er, High Ro11er and 8oom 8oxer



2.5.4 TYPOGRAPHY AND UI ELEMENTS

The game uses mainly two different main font types; one of them is the Police Font (not original but heavily edited from the free Police Cruiser font) and the Graffiti Font (completely original). Together the form the core of the game world's idea, so that is mainly how the logo of the game, present on the bottom left corner of the Character Select Screen, is created. Health Bars are a simple color gradient, and Kinergy Bars are customized to the character's color of choice. Other miscellanea are usually also connected to the pertaining character's theme and use of colors (for example the change character arrows).

Figure 6 - Logo of the game, picturing the Graffiti Font (up) and the Police Cruiser Font (down)



3. GAME FEEL

3.1 INTRODUCTION

Game feel is a newly coined term (also known as game juice or kinesthesia) for an intangible feel when manipulating a digital agent **[1]**. It's mostly subconscious, a combination of sights, sounds, and instant response to action. Game feel can make or break a game, and as time passes, more and more developers are taking note of this concept. Good game feel can make just the act of moving with your character a pleasure, while bad game feel can make the most epic of confrontations feel dull. In this project I tried to identify what makes a game feel good, and the techniques to apply that to my type of game. First I'll separate the pieces of game feel into more easy to understand concepts and then I'll explain the techniques I used in my game and which of each pieces I used in each one.

3.2 PIECES OF GAME FEEL

INPUT

How the player can express their intent to the system.

RESPONSE

How the system processes, modifies, and responds to player input in real time.

CONTEXT

How constraints give spatial meaning to motion.

POLISH

The interactive impression of physicality created by the harmony of animation, sounds, and effects with input-driven motion.

METAPHOR

The ingredient that lends emotional meaning to motion and provides familiarity to mitigate learning frustration.

RULES

Application and tweaking of arbitrary variables that gives additional challenge and higher level meaning to motion and control.

3.3 TECHNIQUES USED

TIME STOP

When you hit an enemy, time suddenly stops for an instant, being the time stopped dependent on the amount of damage of the connecting attack. This gives the bigger impacts more gravitas than the lower ones, and gives the player a moment to appreciate what's happening and to anticipate the effects of his attacks.

CAMERA RUMBLE

Similarly to the time stop, the camera flails wildly when a big hit connects. The shake amount is also dependent on the damage of the attack. The screen also shakes when a character falls to the ground, and attack hits the ground… This technique gives immediate visual feedback of the impact happening.

COMBO SYSTEM

How does the combo system work was explained earlier **{2.3.2}**, but, while I said that one attack can only be chained when the previous attack finishes, that is a small white lie. Every attack actually has a certain moment in which you can transition to the next (signaled by a circle effect), before it even finishes, so the combos and the overall flow of the game feels more fluid.

ENEMY RESPONSES

As enemies are usually in the receiving side, their amount of animations for being hit by different types of attacks is way higher than that of the player characters. That adds variety in animations to the combat loop and additionally a cool visual flair when the animations are shown in sequence (it really feels that you are bombarding the enemy with lots of different hits).

K.O. SYSTEM

When an enemy's health drops to zero, the enemy will enter K.O. state in which you'll have to use a Kinergy Attack to finish it **{2.3.2}**. This gives the player a moment of peace, as well as providing them with the opportunity to finish it that they found too difficult to perform in the midst of the battle.

DASH MOVEMENT

Rapidly press two times the move controls to dash very fast during few milliseconds, and then reduces your velocity until it's slightly higher than the usual moving speed. When you release the moving button, your speed decreases to 0 while a braking animation plays. Relating the rapid double press to an instantaneous burst of speed gives the player feedback directly related to the input, and taking a short time to fully stop gives the character a place in the world.

JUMPING AND FALLING

When you jump there's a small crouch animation for the character to propel himself in the air, and when you fall down to the ground, there's a short recovery animation too. While jumping the more you hold the button down, the more you'll stay in the air. That is made by changing gravity itself, and gives the jump a floaty feel that is usually preferred in games.

4. NARRATIVE VALUE THROUGH GAME MECHANICS AND ART

4.1 INTRODUCTION

As said before, one of my main goals is to convey narrative through the unique tools only a videogame can offer. In every frame of animation, mechanic or piece of information I tried to represent at least a hint of the character's traits and characteristics. In this section, we'll be looking only at a handful of these units of narrative which I think are representative of the game's whole.

4.2 ANIMATION

One of the major ways I'll be using to express the characters and the environment will be through classic frame-by-frame 2D pixel art animation. The style used for the 2D is sort of a middle way between minimalistic sprites (such as found in games like Celeste or Undertale) [2] [3] and highly-detailed ones (King of Fighters or Marvel vs Capcom 2) [4] [5]. As is, the sprites have detail enough to show the character's and environment main features and some details, but small enough so the animation process can be fluent and possible to make in a reasonable amount of time.

In the animation process, I'll be heavily taking a page from the 12 principles of animation [6] and trying to improve along the way. At the end of the document I'll link to a database of all the sprites used for the reader's reference.

4.3 CHARACTER DESIGN WORKSHOP

Here I'll resume the characteristics and main design choices made for every character in the game. It will be summarized in the next categories:

Description:

Small summary of its background and personality traits. Help to comprehend the character and are reflected in animation and design choices, discussed after.

Core Themes and Design Goals:

The main narrative units I wanted to convey with each characters. One of my goals discussed earlier is to make the feel different and cohesive within each other, and I did it through the lenses of the design goals defined here.

Keywords:

Words that best describe the character and its traits.

Color Palette and Motifs:

Colors used and visual motifs that persist in the character’s spritesheet. I wanted to give each of the characters one dominant color that is also the color of his special Kinergy powers. This color is present in the clothes and other paraphernalia, but it always take front center when you make a special attack. Heavily relating a color to each character helps immensely in things like UI design, as you can easily make a parallel to that character if you use said color. You’ll see the color white is present on all of the character’s color palettes, in the jackets present in all the characters (although in different shapes of clothing). That is important to give them cohesion as a group of characters instead of unrelated individuals.

Shape Theory and Silhouette (Idle Animation):

A hard look at the Idle animation (when the character’s isn’t doing anything), discussing its importance and analyzing it through the lenses of the shape theory and the silhouette technique.

Gameplay:

Explaining the gameplay loop of that character and how it differs from the others.

Attacks:

Brief rundown of the character’s repertoire of attacks.

4.3.1 CAPTAIN SUMMERS

Description:

The head of operations of the Beat'em Squad, Captain Summers is a nervous wreck that barely holds himself together most of the time, so containing the antics of his comrades usually proves too much for him. Although it may seem to be a scaredy man that doesn't belong in his line of work, his heart is in the right place, and in the most crucial of times, he is the one you can count on the most. He prefers defensive maneuvers before attacking, and though it may seem he does so in order to shield him from the battle, it's truly because he is a too much of a kind soul to hurt anyone. He enlisted in the PGF **{2.5.1}** in a genuine attempt to better himself and protect the weak and the defenseless of the world. The second-to-most veteran of the squad, he was in a different group before together with Ronin {4.3.4}, but their leader suddenly left… Now appointed the leader of the Squad in charge of the District B10 **{2.5.2}**, he tries his best to hold the group together. With his Kinergy powers, he can form armor around himself and augment his physical abilities.

Keywords:

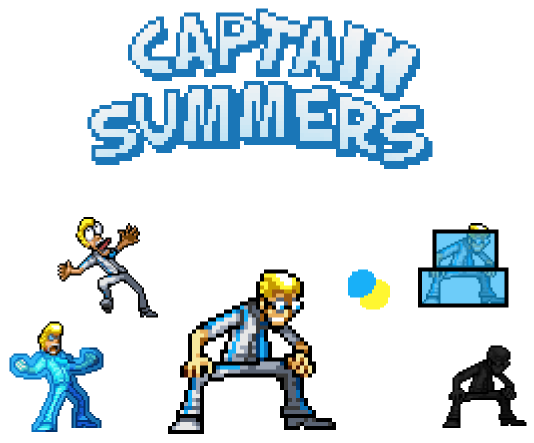
Smart, Coward, Determined, Klutz, Brave

Core Themes and Design Goals:

The main goal with this character is to show its duality in its approach to conflict. Though he is a natural coward, his determination to do good and save people is second to none. The way he moves himself most of the time is clunky, disorganized and graceless, emphasizing the cowardly and nervous side from a comical standpoint, but as he keeps holding on and gets to the stronger attacks of his repertoire, you can see a change of attitude. Lacking the confidence, sheer power or technique of his comrades, Summers excels in analysis of the situation and outsmarting his opponent.

A bookworm, cowardly klutz that will protect the innocent and overcome his enemies.

Figure 7 - Captain Summers, samples, shape theory, color palette and silhouette



Color Palette and Motifs:

Blue is his dominant color and the color of his Kinergy attacks. Yellow and white are the secondary ones. He is the only the one that wears the full PGF uniform, jacket, pants and tie, so white is more prominent in him than the others. This trio of colors gives him a knightly look, that while appropriate to his lawful good personality and appreciation of order and rules, also contrast with his usual cowardly demeanor. His dangling tie, gigantic toupee and glasses are also points of interest in him, animating them using great exaggeration to convey that graceless feel. When using his Kinergy powers, sort of a spiral motif appears in every attack before the Kinergy armor covers his body. For the stronger attacks, his armor takes a crystalline, diamond-like look to emphasize its hardness.

Shape Theory and Silhouette:

In this idle, he's trying to compose himself by staying in a square-like, very static pose, but his nervous nail munching gives away all sense of stability. His toupee is certainly the most prominent in his silhouette, and he is generally lankier than the other characters.

Gameplay:

His gameplay consists on alternating his low damage but fast basic attacks with his slow but heavy hitting Kinergy attacks. Together with The Wonder, he's one of the characters with a lower skill floor, but his unique block and parry mechanics give him depth, forcing the player to increase their awareness of the enemy attack (mirroring the analytical way of fighting of the character). Also by parrying enemy attacks (blocking just in the right time) your Kinergy Bar will increase dramatically.

ATTACKS:

- ***1st Basic Attack (Basic, Barrage, Standard, Physical):***

As it's probably the first attack you'll see, it's important that it makes clear that he doesn't likes violence, as he covers his face and looks to the other side while flailing its arm in the enemy's direction. This attack does very little damage, but stuns the enemy enough to easily chain into other attacks.

- ***2nd Basic Attack (Basic, Charged, Moving while Attacking, Hit Condition, Physical):***

This attack it's pretty unique: Summers throws a punch, if it hits an enemy, its fast and combos nicely, but if it fails, Summers will stumble around until it falls down to the ground dealing heavy damage in a substantial range of effect. You can hold the attack button to continue stumbling and change his direction slightly with the move controls.

- ***1st Kinergy Attack (Kinergy, Charged, Parry, No Damage, Physical):***

The only attack in the game that doesn't deal damage by itself, (outside of the projectile-based attacks) Summers puts up his guard and protects himself from enemy attacks. If you block the enemy attack in a certain timeframe (from putting up the guard to the small sparkle in the knuckles), you'll parry it and deal minimal damage in a wide area. As long as you hold the button down, the guard will stay.

- ***2nd Kinergy Attack (Kinergy, Charged, Parry, Moving Attack, Physical):***

Summers covers his entire body with armor and tackles the enemy. This attack, while slow and very lacking in range, deals a lot of damage and it's the main source of damage of the character. If you release the button too early, the tackle will be pretty weak. This attack also works a blocking maneuver as you are immune while executing, and, though it is pretty difficult, if an enemy hits you just when putting up the armor, the attack will transform into an even more powerful one.

- ***3rd Kinergy Attack (Kinergy, Charged, Moving Attack, Insta-kill, Physical):***

Summers steels himself and puts all of his power into his right fist, unleashing a devastating jab with so much strength that it drags the rest of the body along the screen. The only attack with three charge phases, it can be used as a basic combo finisher or as an insta-kill with great forward range.

4.3.2 THE WONDER

Description:

Coming from a poor family background, Alice always had a knack for helping people in need. Influenced by the black and white morality of superhero TV cartoons, at around her teenage days she started to fight the real fight in the streets as The Wonder, a masked vigilante admired by the community. Her kindness in tandem with her happy-go-lucky personality made her a darling of the masses. When the PGF launched its recruitment campaign, The Wonder jumped on the chance to try to do the greater good. With her karate skills and her electrical powers, The Wonder is always the first to get in to the face of enemy forces and overwhelm them with speed and unexpected attacks. Her Kinergy powers are a rare kind that lets her manipulate and turn into electricity.

Keywords:

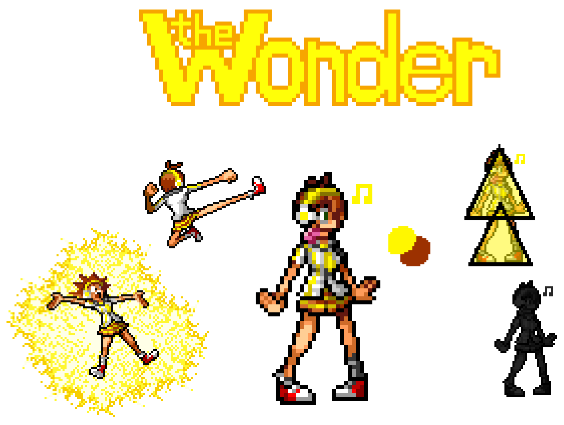
Energetic, Agile, Carefree, Crazy, Dumb

Core Themes and Design Goals:

The main goal with this character is to show its carefree nature and out there antics. The joy of playing this character is in zapping all around the battlefield, unleashing a flurry of kicks and punches in an enemy before appearing in the other side of the map instantaneously. With her cartoony attacks and fast moving speeds, I want to instill her sense of pure, hyperactive fun in the player. Her mobility and area of effect attacks makes her the ideal character to start with.

An hyperactive storm of unaltered craziness that jumps around the battleground at blitzing speeds.

Figure 8 - The Wonder, samples, shape theory, color palette and silhouette



Color Palette and Motifs:

Yellow is his dominant color and the color of his Kinergy attacks. Brown and white are the secondary ones giving her a warm color palette that messes nicely with the pure, friendly vibe that we're going for with her. When she moves transformed in electricity, she takes the shape of a cartoony Z-shape lightning. All her animations have this spastic type of movement, the frames changing shape at a blinking eye's speed. Her design is the most colorful one of the bunch (with red in the feet, and pink when she sticks out her tongue) as I based slightly in the palette appearing in classic out of service analogic TVs.

Shape Theory and Silhouette:

Her idle can be seen as two triangles one above the other, giving her that sense of instability and danger. Through her strange poses and way to move, she is easily differentiable of the other characters. Other noticeable elements of her design are the two hair strands that form like a television antenna and the headphones that cover them. The hair strands actually change shape in attack animations, mimicking the attack made.

Gameplay:

Her style of play mainly consists in moving and jumping around the battlefield all the time. As her attack do not deals as much damage as other characters, you'll have to try to hit multiple enemies with her more powerful, as to not waste Kinergy. Her attacks usually have huge area of effects, so hitting multiple enemies is made way easier than with other characters. Her way of gaining Kinergy also rewards hitting multiple enemies at once.

ATTACKS:

➢ 1st Basic Attack (Basic, Standard, Physical):

Just a simple kick that follows a short hop, pretty fast and spammable.

➢ 2nd Basic Attack (Basic, Barrage, Physical):

After a short buildup, The Wonder unleashes a flurry of karate chops, dealing moderate damage and finishing with a devastating finisher. The recoil time of this attack if you stop mashing the button is pretty noticeable, so it's convenient to try to get to the finisher.

➢ 1st Kinergy Attack (Kinergy, Charged, Electric)

Her bread and butter crowd control attack, this is possibly the best attack in the game against groups of enemies, as it's very fast and has a pretty big area of effect. You can charge this attack to cause way more damage, but the recoil animation is very long, so take care to hit the enemies with it when you use it.

➢ 2nd Kinergy Attack (Kinergy, Charged, Hit Condition, Moving Attack, Physical)

The Wonder shoots a ray of electricity with her fingers and moves herself to the position of the first enemy you hit with it, unleashing a powerful dive kick on the unsuspecting foe. This move serves as a strong reposition tool that leaves you in a favorable spot just behind the enemy.

- **3rd Kinergy Attack (Kinergy, Insta-kill, Electric)**

Long buildup to an explosion of electricity with decent width that insta-kill the enemies it hits. Contrasting with the high speeds of the other attacks, this is the big and slow, heavy-hitting move of the bunch.

4.3.3 X-42

Description:

X-42 was just a baby when the strange meteor that gave everyone Kinergy powers hit the earth. The only survivor of his whole town, which was relatively near the point of impact, the rescue teams found the baby literally bursting with explosive Kinergy. X-42 was brought to a scientific facility to study and contain his extraordinary powers. There he was experimented on along with others until a big incident happened in his late childhood. The teenage X-42 was said to be guilty of an explosion that wrecked the entire facility, and he was brought in a high-security prison. The PGF would occasionally release him and use his tremendous strength when there was no other option for completing an important mission. During the recruitment program, X-42 was permitted to join the organization oficially. X-42 has an insatiable thirst to use their powers on their enemies, but known to few, he also enjoys using his powers for good. X-42 can emit a strange, liquid substance made of pure Kinergy that explodes on contact. If this substance isn’t constantly expelled from his body, he himself could volatilize.

Keywords:

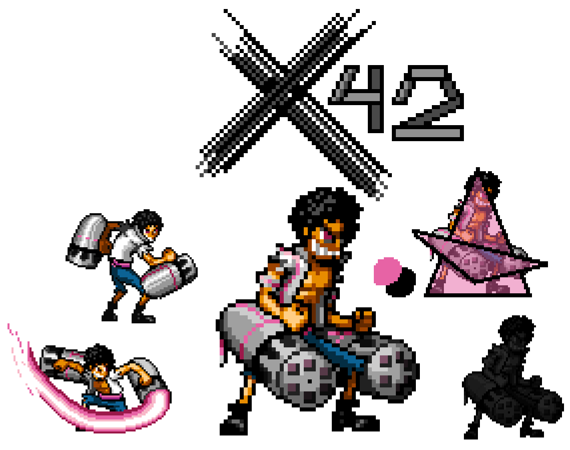
Brute, Heavy, Power, Explosive, Aggressive

Core Themes and Design Goals:

The main goal with this character is to show its overwhelming power and strength. X-42 attacks are slow, but its effects reverberate along the screen. The X-42 brings with him two enormous guns named Blasters, which functions both as his main ranged and close-combat weapons. How the Blasters work, and expressing how much they weigh, is a priority in the portrayal of this character. His attacks are slow, but with a constant motion. With his portrayal and moveset, I wanted to give the sensation of a big, weighty, unstoppable force in movement.

An unstoppable force of nature that blows away everything that dares oppose him.

Figure 9 - X-42, samples, shape theory, color palette and silhouette



Color Palette and Motifs:

Pink is his dominant color and the color of his Kinergy attacks. Black and white are the secondary ones, being the use of black his differing origins and motives from the rest of the cast. When he expels or accumulates too much energy at once, the Blasters and himself start overheating and shaking uncontrollably, which contrast with the slowness and forcefulness of his attacks.

Shape Theory and Silhouette:

In this shape of the Idle animation, for this character I prioritized in making sure the player gets a hint of the Blasters shape and functions, instead of giving a clearer look at some of his other features. His silhouette is probably the most distinguishable of the bunch, because of his bigger body frame, in conjunction with the enormous Blasters, his loose jacket and his afro.

Gameplay:

X-42's Kinergy Bar fills rapidly by itself but once it's full, its Health Bar diminishes as its powers overcharge and he starts to suffer damage. This forces the player to always stay on the offensive, using Kinergy Attacks to waste energy so it doesn't overcharges. X-42 is one of the more difficult characters to play because of that, and also because his attacks are slow (though once they start going, the damage they pile up is tremendous) and you'll have to be careful with your placement to hit with your long-range attacks. He is the character with the highest horizontal range of them all, and the biggest repertoire of projectile attacks. Try to stay outside of enemy's range while you bombard them constantly with your long range attacks.

ATTACKS:

- **1st Basic Attack (Basic, Physical):**

A swing of the Blasters that although it has a long windup, it deals moderate damage for a basic attack.

- **2nd Basic Attack (Basic, Physical):**

After a short buildup, X-42 makes the Blasters spin at vertiginous speeds and slams the enemy within their vortex. An alternative to the more convenient close range combo (see below) that deals a lot of damage and stuns the enemies for a while.

- **1st Kinergy Attack (Kinergy, Charged, Bullets)**

X-42's main long-range attack, in which the Blasters shoot a long volley of Kinergy Bullets. As you hold the attack button, the number of bullets fired increases in quantity and speed (dealing more damage), but if you don't release until the end, the Blasters will overheat and the recoil of the attack will push you backwards.

- **2nd Kinergy Attack (Kinergy, Hit Condition, Insta-Kill, No Damage, Physical)**

This is a one of a kind attack, as if you press the Kinergy Button again after this attack, it will execute another totally different attack (normally it would start from the 1st Kinergy Attack again) **{2.3.2}**. The first attack is an explosive slam to the ground that serves as the main close range combo and doubles as a way to easily waste Kinergy and avoid overcharging.

The second attack is a short range, insta-kill move that grabs and destroys the enemy. Be careful if you use this around more than one enemy, as it leaves you vulnerable to attacks.

- **3rd Kinergy Attack (Kinergy, Insta-kill, Nuke)**

Long buildup to shooting a massive nuke that insta-kills every enemy in a wide range. Although it’s probably the most powerful attack of the game, it’s also the slowest, so use it only when you are totally sure that will hit multiple enemies with it.

4.3.4 RONIN

Description:

Ronin is the most veteran member of the squad. Previously a police officer, Ronin joined the PGF way before the global recruitment process. Obsessed with the old ways of the samurai, Ronin upholds order and peace in the world vehemently following the Bushido code. In his previous squad, he felt betrayed by his captain, so he adopted the moniker of Ronin. Although with his strange Japanese habits, he can be seen as an eccentric or even as a violent lunatic, he follows orders to a T and will always stay on the side of the law. With his highly-trained Kinergy powers, he conjures constructs with the shape of a sword. He also has a Japanese katana, and uses both the real and the Kinergy sword alternatively and with great skill.

Keywords:

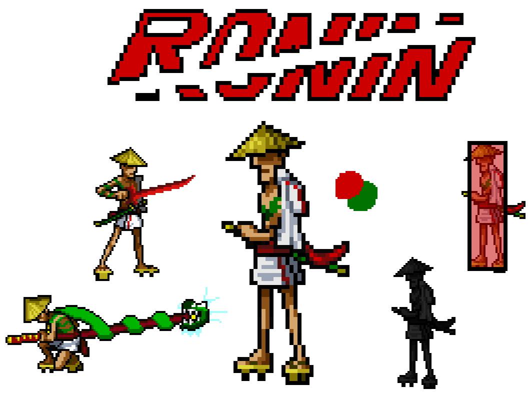
Calm, Skillful, Eccentric, Weeb, Reserved

Core Themes and Design Goals:

Technique and speed are all that define this character. Depending on which side is he looking, his Kinergy Attacks are totally different, so he has almost double the repertoire of attacks of the characters, although they are generally weaker. The curve of learning is way steeper in this character than in the other, which mirrors his disciplined way of fighting. While the attacks made while looking to the right (using the Kinergy made sword) are more fast and fierce, the attacks made while looking to the left (using the real katana) are slow but with greater range.

The lone samurai can topple every challenge with his superior sword-skills.

Figure 10 - Ronin, samples, shape theory, color palette and silhouette



Color Palette and Motifs:

As to reflect his different set of attacks depending on the way he's looking, his primary color changes if he's looking to the right (red) or to the left (green). If he's looking to the right, his secondary colors are green and white, and looking to the left, red and white. Some of his attacks have a japanese flair to them, displaying Japanese kanji characters, sakura flowers falling or even eastern dragons.

Shape Theory and Silhouette:

His idle suggests a straight line, representing him as a collected individual unfazed against the wind that opposes it. He is taller and stiffer than the others and does not make unnecessary movements. His attacks go from a calm buildup to an instant moment of impact, finally regaining the calmness in the recovery. Ronin's unusual samurai's attire gives away his silhouette.

Gameplay:

His moveset is tailor-made so he can chain long strings of attacks. This higher count of different attacks benefits his combo-based gameplay, as there is a different attack for every situation. Ronin changes sheer power for versatility, so only high skilled will be able to use him to his highest potential. Also, some of his attack change where he's looking, letting the players use attack from both sides without even moving.

ATTACKS:

- **1st Basic Attack (Basic, Standard, Physical):**

A short backwards hit with the pommel of the sword's sheath that deals minimal damage. Serves to set up other attacks.

- **2nd Basic Attack (Basic, Hit Condition, Physical):**

Ronin does a short hop forward, and if it hits an enemy, he takes impulse and makes a somersault around him, ending in his back. This move is essential in Ronin's moveset. Together with the other Basic Attack you can perform an infinite combo if you time your button presses right. It also serves to set up a combo finisher with the final Kinergy Attacks.

- **1st Kinergy Attack –Left- (Kinergy, Standard, Slashing)**

Ronin does an instantaneous flurry of slashes at both of his sides. This is the main crowd control attack of his moveset and it also deals moderate damage.

- **1st Kinergy Attack –Right- (Kinergy, Standard, Slashing)**

A simple backwards slash with his Kinergy sword that deals a high amount of damage.

- **2nd Kinergy Attack –Left- (Kinergy, Hit Condition, Projectile, Moving Attack, Slashing)**

A long thrust forward with the katana that covers a lot of ground and deals delayed damage to the enemies it hits.

- **2nd Kinergy Attack –Right- (Kinergy, Barrage, Moving while Attacking, Slashing)**

Ronin starts spinning and forms a tornado of slashes that you move with the Move Controls. Useful for crowd control and escaping difficult situations.

- **3rd Kinergy Attack –Left- (Kinergy, Standard, Slashing)**

Ronin fuses his Kinergy powers with his katana to unleash a powerful slash that covers the entire screen. This is the only attack of the game that hits all the enemies no matter where they are.

- **3rd Kinergy Attack –Right- (Kinergy, Barrage, Slashing)**

Ronin starts dashing around the screen, cutting all the enemies in the midst of his trajectory.

5. DEVELOPMENT

I'll be dividing this section into two big sub-sections that also correspond with the tasks earlier mentioned. To explain how it works, the individual concepts of each sub-section will be explained individually and then I'll explain how they connect with each other. At the side of the pages there'll be examples of the code to help understanding, but it is preferred to keep the repository near to better understand certain parts of the code. If a certain development concept is also a class I'll mark it with ***cursive*** text and expand upon three categories:

Overall Function:

Summary of the class most important role/s

Relevant Variables:

The variables that are most important to the understanding of the process or the more complex ones (like variables of custom enums or structs).

Interesting Methods:

Methods most important to the understanding of the process or the ones that in my opinion required more work.

5.1 CREATION OF A FUNCTIONAL PROTOTYPE

In the first functional prototype (that corresponds with the Core Game section mentioned earlier) **{1.4.2}** I focused in building a structure for the character's to function in, including character movements and all variety of attacks. I'll not list everything here, but I'll try to convey how the main mechanics function.

5.1.1 CHARACTER ACTIONS

We'll use an structure akin to the MVC (Model View Controller) model, where the data will flow as Input→ Model → View. How the code flows from the press of a button to the game showing the action the button is mapped to (for example, to pressing "D", to the character moving forward in game) will be then showed (see *Figure 11*):

REAL INPUT

Meaning the real, physical actions that the player does outside of the game, like pressing a button in the keyboard or tilting the joystick in the gamepad.

DEVICE TRACKER

Overall Function:

An abstract class to customize different devices, (Keybord Tracker, Gamepad Tracker, Touch Tracker etc…) passes the information to the Input Manager.

KEYBOARD TRACKER

Overall Function:

Inherits from Device Tracker, parses the button presses on keyboard and passes it to Input Manager.

Interesting Methods:

```
public override void Refresh()
```

Creates two arrays for buttons (KeyCode[]) and axes (a custom struct array AxisKey[] that on keyboard consists of two KeyCodes, positive and negative)

```
void Update ()
```

Checks for inputs, if inputs are detected, populates InputData (contains the data of the buttons pressed and the way the axis is tilted) and passes it on to Input Manager.

INPUT MANAGER

Overall Function:

Here, we receive InputData and pass it onto the Controller and some UI Elements. InputData is defined here. In the inspector, we can decide how many buttons and axis we'll be using and how do we map them to the keyboard.

CONTROLLER

Overall Function:

Template for future controllers passes the ActionData onto the Animation Manager.

PLAYERCONTROLLER

Overall Function:

With information received by both the player's inputs (as talked before) and the effects of the environment (for example, if an enemy hits the player character), manages all actions and reactions that the player character can perform. Possibly the core class of the entire game. This is a virtual class, so the individual characters controllers inherit from this class to reflect their previously mentioned special quirks (for example, in the child class of the character Ronin, RoninController, its mechanic of switching attacks depending on the direction he is facing towards is reflected on certain methods). **{4.3.4}**

Relevant Variables:

```
public FacingDirection facing
```

A custom FacingDirection enum that stores which direction the PC is facing. Useful for knowing when to flip the character's sprites and hitboxes.

`protected Attacks attack;`

A custom Attacks enum that stores the attack the character is making, if any. Useful for knowing if the PC is actually attacking and to know how to chain combo attacks (for example if it's making the first Basic Attack and you press the Kinergy Attack button, you'll perform the second Kinergy Attack)

`protected ActionData actions`

A custom struct ActionData that stores all the data with value to the Animation Manager, that is to say, all the data that will make the PC change animations.

Interesting Methods:

`public override void ReadInput (InputData data)`

Reads the buttons pressed and acts accordingly. Here we set up the character's movement and attacks. Also here we control if the attack requires a special type of input (see hold or mash attacks).

`public virtual void LateUpdate()`

The main loop, here we control if the character has changed directions, the character's movement and we send the required data (stored in ActionData) to the Animation Manager.

```
public virtual bool receivedHit (int damageReceived, float stunTime, b
ool front, Transform transform)
```

If the character receives damage, its health diminishes and the proper animation plays (as this is immediate and doesn't depend on the character's actions, the animation data is sent by a direct call instead of using ActionData)

`#region` – Wait Events

All the IEnumerators used to control cooldowns. An example would be if the character is hit and gets some invincibility frames, the Ienumerator inmuneTimer is called as a Coroutine and it waits a predetermined amount of time until the character can be hit again.

`#region` – Animation Events

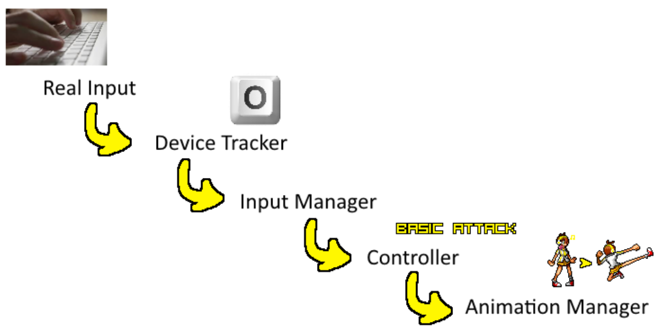
All the methods that are called directly when a certain frame plays on screen. An example would be if the PC uses an attack of the type Moving Attack, the frame calls the method StartMovingAttack and the character moves forward in a speed passed by in a parameter)

ANIMATION MANAGER

Overall Function:

It receives ActionData, interprets it, and changes the pertinent variables on the Animator component of the character.

Figure 11 - Flow from button press to character executing action



5.1.2 FRAME PERFECT HITBOX SYSTEM

First a little bit of explanation of the Hitbox concept: prominent in fighting and action games, a hit-detection system consists in mapping two different types of colliders that are mapped to different attacks. The colliders can be Hitboxes, that represent the damaging area of an attack or Hurtboxes that represent the area in which a character can be hit. In the characters' animations, hitboxes and hurtboxes vary in quantity, form and position. This system is way less resource-intensive than simply making the colliders apply to the shape of the model itself and it's as effective or even more so in certain cases (as the developer has total control over them, is possible to tweak them to make more responsive and permissive hit detection).

Due to the nature of the fast moving gameplay, one of my requirements was to make a frame-perfect hit detection system. To do so, I decided on three different options to approach, and evaluated its pros and cons.

Table 6 Hitbox Systems Pros and Cons

Options	How does it change?	Pros	Cons
1- One hitbox and hurtbox by character	Changes shape and position every frame by using Animation Events	Easy to visualize in inspector and less resource-intensive	Difficult to tweak individual frames, very difficult to control more than one box at a time
2- Multiple hitboxes and hurtboxes per animation	Changes to another set of boxes every time it changes animation	Easy to implement	The less responsive and customizable one
3 - Multiple hitboxes and hurtboxes by frame	Changes to another set of hitboxes and hurtboxes from one frame to another	The most responsive and easily customizable, as you can change shape and position of the hitboxes and hurtboxes in every individual frame in the inspector	Probably the most resource-intensive and the biggest workload

I finally decided on option 3, as I needed that responsiveness due to the fastness of the combat. Now to explain how it works, I'll first explain the different pieces that make it work and then I'll explain how they connect with each other.

CHARACTER GAMEOBJECT

It has every component that makes the character, but in this subsection I'll focus in the Animator component and the HitboxManager script (technically, a class for every character that inherits from Hitbox Manager, for example SummersHM for the character Captain Summers). Contains

ANIMATIONS GAMEOBJECTS

Gameobjects associated with each and every one of the characters' animations. Contains

FRAME GAMEOBJECTS

Gameobjects for all the frames of an animation. It's enabled if that certain frame of animation is happening and disabled if it isn't. Contains

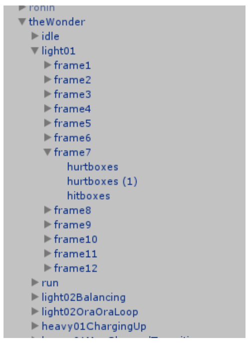
HITBOX GAMEOBJECTS

Contains the area that damages other characters, as well as characteristics pertaining to that area like how many damage it does, how much time does it stun enemy, the type of the damage, etc... There can be none, one or multiple hitboxes.

HURTBOX GAMEOBJECTS

Contains the area that can be damaged by other characters. There can be none, one or multiple hurtboxes.

Figure 12 - Sample of how the GameObjects relate to each other



```
▼ theWonder
  ▶ idle
  ▼ light01
    ▶ frame1
    ▶ frame2
    ▶ frame3
    ▶ frame4
    ▶ frame5
    ▶ frame6
    ▼ frame7
        hurtboxes
        hurtboxes (1)
        hitboxes
    ▶ frame8
    ▶ frame9
    ▶ frame10
    ▶ frame11
    ▶ frame12
  ▶ run
  ▶ light02Balancing
  ▶ light02OraOraLoop
  ▶ heavy01ChargingUp
```

HITBOX MANAGER

Overall Function:

Maintains an organized collection of all the hitboxes and hurtboxes pertaining to a character, enables and disables them based on the current frame of animation. As noted before, each character has a customized version that inherits from this class, but for ease of explanation, I'll treat them as one.

Relevant Variables:

```
protected BoxCollider[][][] colliders;
```

Contains and organized array of all the colliders (hitboxes and hurtboxes) of a certain character.

`public enum animationsSummers`

Contains an enum of every animation of a certain character (animationsSummers, animationsWondie, animationsX42…)

`protected int frameCounter`

Keeps track of the frame in which the animation currently is

Interesting Methods:

`void Start ()`

Through a breadth-first search, populates the colliders array with Colliders from every Frame Gameobject

`public void setColliders (animationsSummers animation)`

Called by the Animation Events, takes into account the animation playing and the frame counter to enable the correct colliders.

`public void clearColliders()`

Called by the Animation Events, disables the current colliders if an animation is finished or if something wrong happens.

ANIMATOR COMPONENT AND ANIMATION EVENTS

The Animator Component contains all the Animation Clips pertaining to the character holding the component and in these you can use Animation Events, which allow you to call functions in the object’s script at specified points in the timeline.

Now that every component has been individually discussed, let’s see how to connect into eachother:

1. The Hitbox Manager stores all of the characters colliders (hitboxes and hurtboxes), looking to the child gameobjects of the character gameobject itself, in a properly organized colliders array.
2. A character animation plays
3. The first frame of the animation call via Animation Event the setColliders method, indicating which animation is currently playing
4. The Hitbox Manager executes the method, starts the framecounter by 0 and receives the animation, with that data, it searches the correct colliders in the colliders array and enables them.
5. The second frame of the animation calls via Animation Event the setColliders method, indicating which animation is currently playing.

6. The Hitbox Manager executes the method, disables the previous frame's colliders, actualizes the framecounter and it searches the colliders in the colliders array and enables them.

7. Repeats these steps, when the animation loops or the animation changes, clearColliders is called and the frame counter starts again from 0.

5.1.3 PROTOTYPE FUNCTIONALITY

At the end of this part, the characters are able to move around the environment, all of their animations have been implemented, and they can attack and be attacked by other characters.

5.2 COMPLETE VERSION

5.2.1 EFFECT SYSTEM

As I talked before **{3}**, effects serve immensely to convey actions, impact, and to give life to the characters and the environment. My main purpose with this system was to separate the effects from the characters themselves, so they can function better as elements of the world. To explain how it works, separate concepts will be discussed individually and then how do they work in tandem.

EFFECT PREFABS AND EFFECT PROPERTIES

Every effect in the game has its own GameObject with the animation attached to it and a script that tells them when to destroy themselves (based on Animation Events). There are certain special ones, like projectiles, that have hitboxes and certain parameters (like speed) attached to them or physical bodies that are affected by the game's physics before they are destroyed.

EFFECT MANAGER

Overall Function:

Stores and spawns the effect prefabs when needed.

Relevant Variables:

`public enum Effects`

A public enum that lists all the effects that can be used

All the `public GameObject` "name of the effect"

Public GameObjects that store the prefabs themselves, ready to be spawned ingame.

Interesting Methods:

`public void SpawnEffect (Effects effectName, Transform parent)`

Receives the name of the effect (to link the enum with the prefab) and the transform component of the GameObject containing the class that called this method. Instantiates the correspondent prefab in a certain position relative to its parent Transform. For the most used positions (spawn in the center of the parent's transform or spawn in a random position within the parent's transform) additional methods are used.

The events happen in this order:

1. A certain animation happens, in a certain frame, an effect is needed
2. An Animation Events calls a function of the Controller class of that animation's character, with the name of the effect required as a parameter
3. This function calls the SpawnEffect function of the EffectManager class, with the effect's name and its character's transform as parameters
4. The SpawnEffect function instantiates the prefab of the effect required in the position proper of that effect and relative to the character.
5. The effect's animation plays out and it destroys itself

5.2.2 ENEMY'S A.I.

For this game's enemy A.I., I wanted it principally to be predictable but fun to play against it. Making it too simple can make the gameplay loop boring and make it complex can be much worse, as the player cannot learn how to play against an unpredictable opponent, making the learning curve inexistent. So i tried to strive for a happy medium, the enemy approaches you and attacks in the same manner, but the attacks it performs vary, keeping the players' expectant of their next moves.

To minimize unexpected situations, the enemy A.I. works by states, having only a few or one action available at each state and changing between them when certain conditions are met. The enemy will always have the player character as the target, and will endlessly pursue him. One important condition that will decide the enemy's action is if the player character is within range of any of its attacks or not. These states are:

INTRO

The introduction animation plays out. Can transition to WAITFORATTACK, MOVING or STUNNED depending on if it is in range of the player character or has been hit by an attack.

WAITFORATTACK

Waits until it decides to attack when it's in range. Time waited in this state is determined randomly between two parameters (that are decided by the developers and determine how aggressive the enemy is the closer they are). If the player character is no longer in range, the enemy transitions to the MOVING state, if it continues to be within range, it transitions to the ATTACK state.

MOVING

The enemy approaches its target until it's in range. When it is, it transitions to the WAITFORATTACK state.

STUNNED

The enemy has received damage from the player's attacks. A certain time passes and the enemy transitions to another state depending on if it's in range of the player's character or not.

MALFUNCTIONING

The enemy is ready to be K.O.ed. The enemy can't do anything in this state and can only transition to the DEAD state. **{2.3.2} {3}**

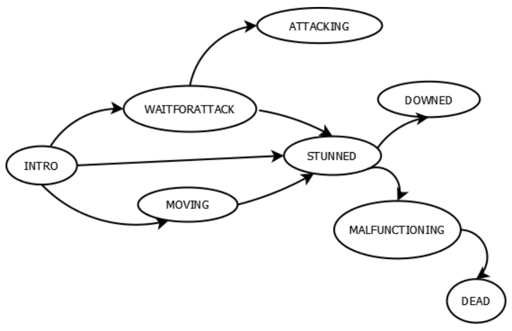
DEAD

The enemy is destroyed.

ATTACKING

The enemy is attacking. The enemy can only get to this state through the WAITFORATTACK state, but the change can be immediate. When the enemy finishes its attack, it can return to WAITFORATTACK or transition to MOVING, depending if it's in range or not.

Figure 13 - How the different Enemy states relate to each other



5.2.3 BACKGROUND EVENTS

PARALLAX EFFECT

To give the impression of an 3d environment with 2d sprites, I use the Parallax Effect technique with 4 layers, which consists in moving these layers at different speeds when the character progresses through the level, which gives off the impression of depth.

STAGE PROPS

Stage props are certain elements of the background that change when the player character moves through certain space or every certain timeframe.

5.2.4 UI AND SCENE MANAGEMENT

In this sub-section, I'll explain how many scenes there are, their purpose and how they are connected. Posteriorly, we'll be seeing how the change between scenes works and how it relates to the actions of the player. Finally, we'll be seeing how you transition in and out of an enemy encounter and what does it entail.

SCENES:

MAIN MENU

The first screen you see in-game and the main menu you'll return every time you die or complete the game. In a previous section this screen was detailed extensively, but it's important to note that from this scene you can transition to the STAGE 1 scene or to the TRAINING ROOM scene.

STAGE 1/2/3

The scenes in which you can freely roam and control a character. Through the Pause Menu, you can transition to the MAIN MENU and if you finish a stage, you'll transition to the next one.

TRAINING ROOM

A simple scene in which you can practice your attacks. Through the Pause Menu, you can transition to the MAIN MENU.

The classes involved in the switching between scenes are:

LOAD NEW SCENE

Overall Function:

It's the class that overseers the game from the start. It loads the appropiate scenes when certain conditions are met and controls the character change and character selection.

Relevant Variables:

```
public string currentCharacter
```

States the player character that you are playing with. Vital for character selection and character changing.

Interesting Methods:

```
public void LoadScene (string sceneName)
```

Function called when the character finishes a stage or when the player selects a character in the Main Menu. The scene with the name equal to the parameter is the scene that it's loaded.

```
void OnSceneLoaded(Scene scene, LoadSceneMode mode)
```

Inmediately starts when a scene is loaded. Sets variables and disables/enables UI depending on the scene that is loaded. If the scene loaded is a stage calls Character Select.

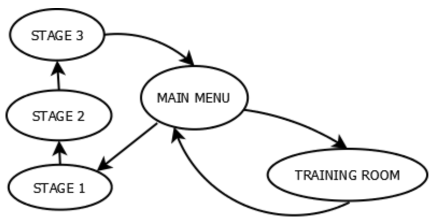
```
void CharacterSelect(Vector3 spawnPoint, int currentHP)
```

Called from OnSceneLoaded when the scene is a Stage or the Training Room, sets the player character variables and starting position. The player character chosen depends on the variable currentCharacter.

From starting the game to getting into the first stage, the code flows like this:

1. Game starts, OnSceneLoaded is called, makes its own gameobject permanent (with DontDestroyOnLoad, the object persists throghout scenes). Disables the UI elements that appear ingame.
2. Selects DEPLOY and chooses Captain Summers, LoadNewScene variable currentCharacter changes to “Captain Summers”
3. Function LoadScene with parameter “stage1” is called, loads the next game with a fade.
4. We are now in stage 1, as LoadNewScene persists between scenes, OnSceneLoaded is called instantly
5. OnSceneLoaded enables all the UI elements that appear ingame and calls CharacterSelect with the starting position of stage 1
6. CharacterSelect instantiates the player character prefab that matches with the currentCharacter variable.

Figure 14 - How the different scenes relate to each other



5.2.5 ENCOUNTER MANAGEMENT

ENCOUNTER MANAGER

Overall Function:

Stores data for all the encounters in the game, displays them in order. The number of each enemy that appear in the encounter, as well as when and where do they appear are defined in the inspector.

Relevant Variables:

```
public Vector3[] spawnPoints
```

Contains the positions in which the enemies will spawn, in order of spawn.

```
public GameObject[] spawnObjects
```

Contains the type and number of enemies that will spawn, in order of spawn.

```
public int enemyCounter = 0
```

Counts how many enemies you have defeated in a certain encounter. When it is zero, the encounter finishes.

Interesting Methods:

```
public void encounterActive(int stageNumber, int encounterNumber)
```

Depending on the stage you are and where you are in said stage, sets up one encounter or another.

```
void encounter”stageNumber”of”encounterNumber”()
```

Depending on the enemies remaining (using enemyCounter) spawns enemies with the coroutine and waitForSpawn. When the enemy counter reaches 0, the encounter finishes.

```
public IEnumerator waitForSpawn (int spawnNumber, float time)
```

Spawns an enemy according to the spawnNumber index in the spawnObjects array as soon as the time specified in the variable time finishes.

The order of events is as follows:

1. The player character triggers the encounter by walking forwards in the stage.
2. The encounterActive method of the Encounter Manager class is called with the parameters stageNumber equal to one and the encounterNumber equal to two.
3. enemyCounter now has the value of 7, method encounter1of2() is called.
4. As the enemyCounter has a value of 7, an enemy appears instantly and two more follow after one second.
5. After defeating one of them, another one appears after half a second. This happens once more when enemyCounter is down to 5 to maintain three enemies on screen.
6. After defeating the three (enemyCounter is now two), one enemy appears instantly and the last one follows ten seconds after.
7. After defeating these two, the encounter is finally over and you are able to continue.

6. RESULTS

The project, the design of its elements and how where they carried out were presented in previous sections.

The executable can be downloaded in:

https://drive.google.com/open?id=1GLcZiHljxN2NJwQ1vAhwLCj30cD5CeJF

https://mega.nz/#F!vC53SaCB!-qBKuXY-d4lPUmDPFfXuZA

In it the result of the previous chapters can be observed. The full download of the folder is necessary for the project to work. Some of the project's major goals are way more appreciable with the game in motion, so it's heavily recommended to keep it open while reading this document.

The art assets organized in folders can be downloaded in:

https://drive.google.com/open?id=1bqlRBBcGlM7ZlvgVOVew_lo8eUucMSXr

The GitHub repository can be found in:

https://github.com/DavidArona/Beat-emSquad

This project has been possible to the basics learned in many of the subjects taught in the degree Video Game Design and Development. Apart from the project results, the making of project has immensely helped me in learning how to manage properly my work hours within a work schedule.

These are the project goals presented earlier:

- ★ NARRATIVE VALUE THROUGH ART AND INTERACTION
- ★ IMPROVE AND LEARN ABOUT GAME FEEL AND ITS KINESTHETIC VALUE
- ★ Offer replayability through the characters' varied movesets and overall different playstyles.
- ★ Taking special care of the animation department to better develop the cast of characters.
- ★ Offer a polished, playable and enjoyable product, made entirely with hand-made resources.

All the main goals have been achieved and overall, I feel more than satisfied with the game results.

7. QUALITY CONTROL

In this section we'll be looking at the tests made for the game to operate as expected. It will be separated in two sub-sections, the user experience, divided in verification and validation, and testing on different devices.

7.1 VERIFICATION AND VALIDATION

The verification process focuses in the review of each of the development phases and the validation process focuses on making sure the project has met its goals. In both cases, the work was made mainly by the developer and acquaintances.

7.1.1 VERIFICATION

This process has been performed along with the project itself. In each step, this process ensures that the project doesn't stray from its goals and objectives, and maintain focus on what it's needed at the time.

7.1.2 VALIDATION

The people given access to the game's final form where asked to give feedback of two types:

- Searching for bugs or other features working not as intended
- Making suggestions to change certain parts.

Suggestions and errors have been since noted and taken care of.

7.2 EFFICIENCY

The key points of possible performance drop-offs are as follow:

- Starting the game.
- Getting from one stage to another.
- Getting a high number of effects in screen.
- Game Over screen to Main Menu Screen.

These performance tests have been carried out at the moment of this writing, and the project has proved to function properly in a diverse set of machines of different capabilities.

8. CONCLUSIONS

Taking into account my first objectives and goals, from my opinion, the project has been a success, even surpassing in some ways what I originally intended. I learned a lot about Game Feel and about how to convey meaning through interactive mediums.

For me, personally, it has also been a huge triumph. I learned how to design better, how to draw better and animate better and how to code and organize better. In the future I'll apply the lessons learned here. I don't know yet if I'll continue working on this particular project, but it's certainly a possibility.

8.1 FEATURES PIPELINE

Features that almost made the cut into the final project but didn't, and why they were scrapped.

8.1.1 BYSTANDERS

This feature was practically done at the time of writing this document, but the lack of polish made me decide against putting it in the final version. It consists in making additional friendly NPCs that roam around the battlefield and the player has to save them before they get hit by the enemies.

8.1.2 BOSS FIGHT

The part of design and implementation wouldn't be a problem, but making a new spritesheet from scratch, would be at least as costly as spriting another whole character so I decided against it to focus more in polish and quality control.

8.1.3 MULTIPLAYER

The possibility of multiplayer, both offline and online, was heavily in my mind from the start of the game, but as it would cause a complete rework of the enemies A.I., it was postponed.

OUTSIDE REFERENCES

[1] Gamasutra – Game Feel: The Secret Ingredient

https://www.gamasutra.com/view/feature/130734/game_feel_the_secret_ingredient.php

[2] Celeste

http://www.celestegame.com/

[3] Undertale

https://undertale.com/

[4] King of Fighters' 98

https://es.wikipedia.org/wiki/The_King_of_Fighters_%2798

[5] Marvel Vs Capcom 2

https://es.wikipedia.org/wiki/Marvel_vs._Capcom_2:_New_Age_of_Heroes

[6] Frank Thomas, Ollie Johnston (1981). The Illusion of Life: Disney Animation