End The Bully
Final Project Report

By
hugo.camarasabirabent-martinez , louise.bendjeguelal
noe.auguste , wachirawit.fasquel , alex.chiriac

EPITA
SUP anglophone section

28 may 2025

Summary

General Introduction

1.1 Project Context

Since September, our team has been dedicated to the development of an original video game, building each feature around the detailed specifications document we made at project kickoff. Over these nine months of collaborative work, we have progressed through every stage of the development lifecycle—requirements analysis, system design, iterative implementation, and rigorous testing—always referencing and refining our initial specifications to ensure alignment with the company's vision. This document represents the culmination of our efforts: the final report that accompanies the project's official deliverable. It summarizes the journey from conceptualization to completion, highlights key challenges and achievements, and provides a comprehensive overview of the game's final functionality and architecture.

Our company, inspired by RogueLite Hack n Slash games such as Cult of the Lamb, developed an exciting game titled End The Bully. Set in a dark and captivating universe, the game follows the story of a protagonist who, after being bullied by the son of the world's main antagonist, takes a stand to seek revenge against his tormentor. The game features classic mechanics of the Hack n Slash and RogueLite genres, challenging players to defeat waves of enemies using unique skills while acquiring equipment through chests or rewards from vanquished foes.

End The Bully also includes a cooperative multiplayer mode, allowing multiple players to team up online via a peer-to-peer (P2P) network system. Enemies are driven by AI, ensuring dynamic and engaging encounters throughout the game.

In this report, we will introduce the advancement of our company and its team on the video game.

1.2 In depth presentation of the game

During the brainstorming phase of our project, we decided to prioritize replayability over creating a long, linear adventure. This approach led us to consider several genres, including sandbox games, PvP shooters, RPGs, and RogueLikes. Ultimately, we chose the RogueLite genre because it provides a meaningful challenge and clear objectives for players, enhancing their engagement with the game.

Our primary inspirations are RogueLite Hack n Slash games like Cult of the Lamb, Risk of Rain 2, Rotwood, and The Binding of Isaac. These games stand out due to their sense of progression, where players feel a continuous improvement throughout their journey, while also offering varied experiences on subsequent playthroughs. Additionally, these titles were developed by independent creators and achieved significant success, which is a motivating factor for our team.

Our inspirations also showcase diverse art styles, including pixel art, cartoon visuals, and realism. For our project, pixel art emerged as the ideal choice due to its accessibility and alignment with our resources and design goals.

To enhance replayability and player engagement, we will implement a loot system inspired by Muck and Risk of Rain 2. This will allow players to progressively evolve their characters throughout a playthrough, ultimately becoming powerful enough to defeat the final boss.

Players will have to navigate through increasingly challenging floors filled with stronger enemies. These foes not only test the player's skills but also offer more powerful upgrades as rewards. However, players will have the option to bypass certain obstacles and move directly to the next floor. While this will speeds up progression, it will come with the risk of facing under-prepared encounters with mandatory boss fights, which will block access to subsequent levels.

Bosses will serve as gatekeepers, presenting mandatory challenges for players to advance. These encounters are designed to test the player's mastery of the game's mechanics and the synergies created by their chosen loot and upgrades.

Our goal for End the Bully is to create a game that balances difficulty with rewarding progression. Each playthrough offers a unique experience, encouraging players to experiment with different builds, strategies, and approaches. By combining engaging mechanics, strategic depth, and a dynamic loot system, we aim to deliver a game that feels fresh and exciting with every run. This design philosophy ensures End the Bully will offer both a challenging and satisfying experience,

resonating with fans of the RogueLite genre while carving out its own unique identity.

Work Organization

2.1 Individual Presentations

Noël

My name is Wachirawit Noel Fasquel, and I am currently a first-year student at EPITA, . I am currently responsible for leading the UI development of our team's game project, where I focus on designing intuitive interfaces that enhance the overall user experience.

Over the past year, I have completed several game-related projects that have helped me grow both technically and creatively. These include a digital version of the card game Skyjo, developed using JavaScript, and a GeoGuesser-style game built in Python. Through these experiences, I have deepened my understanding of gameplay logic, interface design, and player interaction, while also strengthening my programming and problem-solving skills.

I am currently working on a new and ambitious project titled EndTheBully, a game that delivers a strong social message through immersive gameplay. This project has allowed me to continue developing my skills in Unity, while also gaining a deeper understanding of game architecture, UI design, and interactive storytelling.

I am proficient in Python and C, and I also have working knowledge of OCaml, JavaScript, and SQL. I have studied and applied each of these languages in real–world projects, continuously pushing myself to improve both my technical abilities and creative thinking.

As a developer, I am committed to meeting deadlines and consistently delivering high-quality work. I am also curious and driven, always eager to explore new technologies and creative possibilities. I have a particular interest in artificial intelligence in games and level design, as I believe innovation in these areas will be key to shaping the future of the gaming industry.

In the short term, I aim to expand my skill set and refine my creative process. In the long term, my ambition is to become a developer who contributes meaningfully to the gaming world—someone who creates memorable, impactful experiences that leave a lasting mark.

Noé

I am Noé Auguste, a founding member of the All Nighters, originally from Annecy. After studying mathematics and physics at Berthollet High School, I joined EPITA, where I met Hugo, Noël, Louise, and Alex. Together, we created our game development group, All Nighters, and released our first game in 2020.

Outside academics, I've always been passionate about sports. I played basketball for 15 years, first in the E.S.E.M.T club, and now casually with friends or at my father's club during holidays. I also practiced rowing in high school, but had to stop due to injury. In parallel, I have experience as a basketball referee and as a coach for a youth team. Currently, I regularly train at the gym in Lyon with Hugo, Louise, and Noël.

Beyond sports, I would describe myself as a creative person. Despite not coming from an artistic family, I've always had the urge to create through drawing, writing, building, or designing games. This led me early on to experiment with game development, making various projects such as 2D and 3D platformers, clickers, and even a multiplayer FPS. While I mostly followed online tutorials, these experiences were both formative and enjoyable.

My interest in video games also extends to theory. I've read theses such as The Impact of Art Style on Video Games by Eric Sarver and Éléments de game design pour le développement d'une attitude réflexive chez le joueur by Thomas Constant. I also spend significant time watching video game content on YouTube, whether it's analysis, design breakdowns, or reviews.

Given this background, I naturally leaned toward game design for End The Bully. Working on this project has been a great opportunity to apply my knowledge and passion in a meaningful way.

Hugo

I am Hugo Camarasa Birabent, a founding member of the All Nighters, originally from Lyon. I studied mathematics and informatics in the Cité Scolaire Internationale of Lyon and once I finished high school I joined EPITA, where I met Noé, Noël, Louise, and Alex. Together, we created our game development group, All Nighters, and released our first game in 2020.

Outside academics, I have always loved doing some sort of physical activity, most of the time sports. When I was younger, I did ice hockey during three years due to my older cousin, who was a professional at the time. I really liked it but in middle school I did not have as much time as before to train and attend to games so I stopped. However, the next year I started playing tennis, which was fitted better with my new schedule. I enjoyed it and at the same time I practiced ping-pong too. However, after some years, I did not like the sport as much as before and at that moment I did a year of basketball before discovering the sport which passionates me, the gym and more specifically, the powerlifting. Its a sport that allows me to surpass myself all the time and has permitted me to meet a lot of people that have now become great friends.

Beyond sports, I would describe myself as a logical but sometimes impatient person. I love spending time solving problems and trying to find a solution to some sort of puzzle or imagining strategies because I find it really stimulating intellectually. However, at the same time, I love playing FPS or shooter games which are really nervous and quick, or speedrunning games, which requires to have a great knowledge of the game but also great mechanical ability to be able to execute every move precisely and quickly. I love having to find a solution to a problem with a short amount of time available and using every tools that I know and have to my disposition.

Even if I have played a lot of different video games, I never really tried to make one myself. Not so long ago, I imagined one and wrote descriptions and stories for it but I could never bring myself to create it because I never thought I was prepared enough.

When we first talked about this project, my implication and motivation made my group choose me as the Chief of Project, but in reality it was more an honorary title as we all worked together and I was not the only one assuming the group leadership.

Louise

My name is Louise, and I am currently a student at EPITA, where I am pursuing my passion for technology and software development. Prior to joining EPITA, I completed my high school education at Sainte-Marie Lyon, where I developed a strong academic foundation and a growing interest in programming.

Before starting this game development project, I had already acquired skills in website creation and Python programming. These experiences gave me a solid grasp of problem-solving and logic-based thinking, which have been invaluable throughout the project. This initiative gave me the opportunity to apply those skills in a more complex and collaborative environment, and to grow significantly in areas I had not explored before—such as game engines, animation systems, and gameplay design using C and Unity.

Outside of my studies, I have a strong interest in fitness and cars. Going to the gym helps me maintain discipline and focus, which are qualities I also bring into my work as a developer. My interest in cars comes from a fascination with mechanics, design, and performance—principles that also influence the way I approach challenges in programming.

This project has not only strengthened my technical competencies but also helped me better understand my own workflow, how I collaborate in a team, and what I aspire to build in future development projects.

2.2 Initial Task repartition

Task	Employee				
	Alex	Noé	Hugo	Louise	Wachirawit
Art Direction / Graphic Design		R		S	S
Multiplayer / Networking	S		R		
Enemies / AI	S		S	R	
UI		S		S	R
Loot / Upgrades		S	R		S
Combat / Player Experience	R	S			S
Marketing / Communication	S		S	R	

R : responsable, S : substitute

1. **Art Direction / Graphic Design**
   The task of Art Direction (AD) and Graphic Design mainly involves drawing and designing the sprites of various characters and objects in the game, as well as the maps and levels. It also includes sprite animation and will work closely with the Marketing/-Communication team.

2. **Multiplayer / Networking**
   The members of the group handling the Multiplayer/Networking task will be responsible for making multiplayer functional and operational. They must ensure secure network connections and optimize the data transmitted through it to make the gameplay as smooth as possible.

3. **Enemies / AI**
   The Enemies/AI segment of the game involves the AI of all enemies or companions in general, the AI of various bosses, as well as the addition of new enemies with their attacks and specific features.

4. **UI**
   The UI (User Interface) task includes creating and refining the HUD, the game menu, settings (including resolution, key mapping, and game language), the inventory, and potential post-processing to improve the graphical quality of the game.

5. **Loot / Upgrades**
   Those responsible for the Loot/Upgrades section will handle the loot system (i.e., how enemies or chests drop items), the upgrades provided by items, and the addition of new weapons.

6. **Combat / Player Experience**
   The Combat/Player Experience section covers the movement of various characters, the combat system (specifically how attacks are performed), the enhancement of combat with ranged attacks, the major addition of Spells, and finally, the procedural generation of enemies and chests.

7. **Marketing / Communication**
   The Marketing and Communication team will primarily focus on the game's website to present and explain the game in more detail. Secondly, they will handle potential advertising for the game through social media or other platforms.

2.3 Changes in task repartition

Throughout the entire project, some issues were notified by the different members and in consequence some reassignments of the tasks had to be done.

One of the main changes was the passing of the "Multiplayer" role from Hugo to Noé, due to the incapacity of the first to do the required work and find a good way to do it.

Hugo first tried to use Photon, an engine used for multiplayer games, but he struggled to find a good way to implement it and to make it work.

As he spent too much time on the task and was getting demoralized, Noé looked into another solution that he had heard of, Mirror.

Noé managed to make it work and hence took the responsability of the "Multiplayer" task.

This allowed Hugo to concentrate on his other task, "Loot and Upgrades".

On another part, as we did the website relatively early on and the only adjustments we could do to it were simply updates on the project advancement, the "Marketing and Communication" task was finished, therefore Louise and Alex had the opportunity concentrated on their other tasks.

Every other task did not encounter any significant problem and the team was able to execute them without issue or need of major changes.

2.4 Collaborative Tools

Throughout the development of our video game project, effective communication and collaboration were essential. To ensure smooth coordination among team members, we relied on several digital tools, each serving a specific purpose and contributing to the project's success.

DISCORD

The primary platform we used for real-time communication was **Discord. We created a dedicated server exclusively for our team, structured to support both general discussions and topic-specific exchanges. Multiple text channels were set up, each corresponding to a particular aspect of the project—for instance, channels for sprite design, development updates, bug tracking, and general planning. This organization allowed us to keep our discussions focused and our ideas easy to revisit.**

In addition to text channels, we created multiple voice channels, including one for each team member—ideal for individual work sessions—and a general meeting voice channel where we held weekly sync-ups and spontaneous brainstorming sessions. Thanks to this structure, we maintained clear, frequent, and fluid communication, which greatly reduced misunderstandings and helped us progress steadily.

NOTION

We also made extensive use of Notion, a powerful platform that enabled us to centralize and structure our documentation. With Notion, we created a collaborative workspace where we documented our ideas, specifications, task lists, timelines, meeting notes, and progress updates. It was especially useful for tracking responsibilities, breaking down larger tasks into smaller actionable items, and sharing updates with clarity.

As the project evolved, Notion became an essential tool not only for organization but also for preparing our oral presentations, storing visual assets, and writing summaries of completed phases. It allowed everyone to stay on the same page—literally and figuratively.

GIT

Finally, to manage our codebase and version control, we used Git, hosted on GitHub. This enabled us to work simultaneously on different features of the game without interfering with each other's work. We adopted a branching strategy to separate development tasks—each team member worked on their own branch before merging into the main branch after review and testing.

Git also helped us track code history, roll back changes if something broke, and maintain an overall stable and clean codebase. Regular commits ensured that our progress was continuously saved and that our work remained synchronized across the team.

In summary, the combination of Discord, Notion, and Git formed a solid foundation for our collaborative workflow. Each tool served a specific purpose—communication, documentation, and version control respectively—and together, they allowed us to stay organized, productive, and aligned from the first day of development to the final delivery.

Team's Advancement / goals

3.1 Initial Objectives

From the beginning, our goal for End The Bully was to create a game that balanced challenge and enjoyment. We wanted to design a Roguelite where each run would feel meaningful, encouraging players to persevere through repeated deaths and gradually improve through both skill and long-term progression. The idea was to have the player start from scratch after each death, while keeping persistent unlocks such as new classes or bonuses through an achievement system.

We intended to make the gameplay dynamic and skill-based, with movement using ZQSD, melee attacks on left click, ranged attacks on right click, and spell usage on the A key. Our vision was a fast-paced, responsive combat experience that demanded attention and rewarded mastery. To support this, we planned to introduce spells that could be unlocked by completing in-game achievements, giving players side objectives and a sense of progression beyond simply finishing the game.

To enhance replayability, we aimed to offer a wide variety of characters and weapons, each with distinct abilities and playstyles. Our design process included the creation of multiple items with synergistic effects, some providing buffs, others debuffs, allowing players to develop different strategies and builds with each run. These items were to be found in procedurally generated rooms filled with enemies and random chests, with keys dropped by defeated enemies acting as a reward mechanism.

Our ambition was to create a gameplay loop where the player explores dungeon floors, defeats enemies, opens chests, collects synergistic equipment, and becomes increasingly powerful in preparation for the final boss. This structure was inspired by games such as The Binding of Isaac, Cult of the Lamb, Risk of Rain 2, and Rotwood, which all combine satisfying combat with strong replay value and evolving player progression.

From an artistic standpoint, we agreed early on to adopt a pixel art style. It was a practical choice that aligned with our skills and allowed for quick asset production, visual coherence, and easy content expansion.

Narratively, we had imagined a light and humorous fantasy setting: the son of the Demon King steals the player's lunchbox, and the player sets off through the demon's dungeon to retrieve it, collecting gear and fighting magical enemies along the way. Our genre was defined as a hack n slash Roguelite, focused on action, variety, and emergent gameplay.

In short, our initial objective was to build a game that would be engaging, rewarding to replay, and filled with meaningful choices. While the final implementation may differ, this was the direction we had in mind when we began the project.

3.2 Task advancement throughout the project

Our first concern was to have content to implement and a clear idea of what the game could be, that's why we spent a lot of time in meetings and brainstormings to find a great idea.

Once we had done it, we defined every task and who would do it and started to work.

The very first thing we tried to do was develop a really simple version of our game without any multiplayer aspect, to ensure that it was entertaining and could be fun to play.

In this version we implemented :

- the player, with its appearance and a first weapon
- Puddle, our first enemy
- a small and plain piece of land where our character could evolve

Once we were assured that this part of the game worked, we thought about the multiplayer and that is the moment where some problems started emerging.

As we had already started, it was quite a challenge to adapt existing code and features to the multiplayer aspect of the game. At the same time, the team was adding and improving features, making it even more difficult to follow, but in the end we managed to make it work.

At this point, everyone started working on their respective tasks and requesting help from others when they had a problem of some kind or when they had to link or use their code with the one from someone else.

At diverse points during the project, we did some sort of check-ups to verify that everyone was doing their work and that no help was needed from anyone, or in contrary case, what help was needed and how everyone could help in solving the problem.

Everyone worked well in their respective aspects and tasks and we are glad to say that a major part of our project has been brought to reality through our work an the time we put in.

Presentation of the final version of the game

4.1 Gameplay / getting trough the levels

In the final version of the game, the gameplay offers an engaging mix of action, progression, and strategic exploration. Players take control of a main character that they navigate using the ZQSD keys, providing fluid movement in four directions. Right from the beginning of each level, the player is immersed in a hostile environment filled with enemies (mobs) that must be defeated in order to move forward. The character is equipped with a basic weapon that deals damage, and combat is a central component of the experience. Players must time their attacks, manage positioning, and react quickly to threats in order to survive the encounters. The mobs present varied behaviors and attack patterns, making each fight feel dynamic and requiring the player to stay alert and adapt their strategies.

The main objective in each level is to reach and defeat the boss, a stronger enemy with higher health, more powerful attacks, and sometimes unique abilities. Beating the boss is essential, as it unlocks access to the next level, maintaining a sense of progression and challenge throughout the game. This structure creates a gameplay loop where players must balance fighting, surviving, and improving their character as they advance.

Along the way, players can explore the level and find hidden or strategically placed chests. Opening these chests rewards the player with items that grant various bonuses—these can include increased attack power, better defense, faster movement speed, or even temporary abilities that enhance gameplay. These rewards encourage players to not only rush through the level, but to take time to explore and weigh the risks of seeking out loot in areas that may be guarded by tougher enemies.

This progression system, where each level becomes more difficult and more rewarding, adds depth and replayability to the game. The combat, exploration, and reward mechanics come together to create a satisfying and immersive gameplay

experience. Overall, the final version of the game offers a balanced mix of action, difficulty scaling, and strategic choices, making each playthrough feel rewarding and unique.

4.2 Multiplayer

End The Bully features a complete peer-to-peer multiplayer system that supports up to four players per session. The game is structured around a host-client model, where one player acts as the host while others connect as clients.

Connection and Lobby

Upon launching the game, players enter a connection menu where they can join a session by entering a host address and their username. Once connected, players are directed to the lobby, where each participant can see the list of connected players and a readiness indicator. The lobby interface updates dynamically as players join, leave, or change their readiness status. Once all players are marked ready, the host can initiate the game.

Character Selection

The game includes a dedicated character selection phase. Each player can browse a list of unique characters using left/right arrows, view the character's name, description, and visual preview, and confirm their selection before the game begins.

Character previews include both melee and ranged weapon visuals, helping players understand the role and style of their chosen character. The selection interface ensures that all players are aware of each other's choices in real time.

Gameplay Synchronization

During gameplay, each player controls a character with full movement, aiming, and combat functionality. The game ensures that the positions, animations and actions of the players are synchronized across all clients with a real-time updates of weapon orientation and usage.

Each player's unique character appearance and weapon set are retained from the selection phase and displayed accurately to others during gameplay.

Scene Transitions

The multiplayer system supports seamless transitions between scenes from lobby to character selection and from character selection to the gameplay.

Throughout these transitions, player data such as name, character selection, and order of connection are preserved and restored automatically. This ensures continuity and avoids identity or role confusion during the session.

4.3 UI

The user interface (UI) of a game plays a crucial role, as it not only sets the atmosphere of the game world but also shapes the player's first impression. In our project, the first element of the UI is the main menu, which features three primary buttons: Play, Options, and Exit. These buttons were created by adding UI components to the game's canvas and were programmed individually to execute their respective functions. The menu also includes a background image, which was implemented by stretching a static image to cover the screen. To enhance visual consistency and immersion, the buttons were customized using specific color palettes and imported fonts that align with the game's overall aesthetic.

The Connection Lobby is the second major component of the UI, structured into three main sections:

1. Hosting and Joining Menu: This section allows players to either host a session or join an existing one. Depending on the selected role, the player receives different permissions, particularly in the next stage of the lobby.

2. Waiting Lobby: After selecting to host or join, players are placed in a waiting area until all participants are ready. Only the host has the authority to initiate the game, reinforcing the host's role as the session leader.

3. Character Selection Menu: In this final part of the lobby, players choose the character they will play during the game. The selection is influenced by gameplay preferences, such as opting for a fast and agile character or a slower but more powerful one.

The third major component of the user interface is the in-game UI, which includes several essential features designed to enhance the player's experience and support gameplay.

The first feature is a player stats panel, which provides an overview of the character's strengths and weaknesses. This panel allows players to quickly assess their current capabilities and adapt their strategies accordingly. It was implemented using a standard UI panel and can be toggled on or off by pressing the "T" key.

The second feature is the inventory system, a critical element of the game given the central role of items in gameplay. The inventory allows players to view the items they have collected and to make informed decisions based on their available resources. It was implemented using a panel and supported by custom scripts

that manage the addition of new items, the stacking of duplicate items, and the removal of unwanted ones.

Another important element is the in-game menu button, which provides players with the ability to exit the game or adjust audio settings. This ensures a more flexible and user-friendly experience by allowing players to manage sound levels and leave the game session at any time.

Personal Advancement

5.1 Noé

1. First Part of the game development

At the beginning of the project, I developed a basic prototype of the game. This first version included a simple platformer scene built using a tilemap, a border to define the playable area, and a camera system that followed the player as they moved through the environment. I also implemented the player's basic animations for idle, running, and jumping states. One of the core features I created at this stage was the melee weapon mechanic: a sword that rotated dynamically to point toward the player's mouse cursor. This allowed the weapon to be aimed with precision, laying the foundation for directional combat. This work can be found in the early scripts located in the "development" folder of the project.

While building this first version, we were simultaneously holding regular meetings to organize our ideas and divide the workload. We also started producing the initial sprites for enemies, environmental objects, and item pickups. During this time, I played a major role in coordinating our small team. I helped ensure that everyone had a clear understanding of what to do and how to begin their individual tasks.

2. Second Part of the game development - Items

After getting the core movement and combat working, I moved on to designing the item pickup system. I implemented a mechanic allowing players to interact with chests placed on the map. When a player entered the trigger zone of a chest and pressed the interaction key (E), an opening animation would play. Once opened, a user interface appeared, offering the player a choice between three random items. These items were drawn from a list and chosen using a simple randomization algorithm. The player could select one of the items via the UI, and the selected item would then appear in the game world or immediately apply its effect, depending on its type.

The first items I created were designed to enhance specific player stats. For instance, the Sugar item increased movement speed, the Mushroom increased maximum health, and the Dumbbell improved melee damage. I also developed a script to handle generic item pickups placed in the scene, so that when the player walked over them, their stats were updated accordingly.

Once this system was functional, I exchanged tasks with Hugo. He took over the design and balancing of new items, while I began working on the multiplayer system. This transition marked a significant shift in my responsibilities, as I moved from isolated gameplay mechanics to broader game architecture and network synchronization.

3. Third Part of the game development - Multiplayer

To implement multiplayer, I created a new scene dedicated to network play and integrated the Mirror library. Mirror allowed us to use a peer-to-peer model while still maintaining a server-client authority structure. I began by setting up the basic NetworkManager, handling player connections, and managing player prefab spawning. I then worked on synchronizing player movement, animations, and stats across clients using SyncVar attributes with hooks to apply visual and functional updates when values changed.

I also managed the scene transition system so that when the game moved from the lobby to the character selection screen and finally into the main game scene, all clients were synchronized correctly. This required learning how to use the OnServerSceneChanged method and implementing delayed replacements of lobby players with their in-game equivalents. This process also involved carrying over certain variables such as character selection and player preferences.

One of the biggest difficulties I faced during this phase was displaying usernames correctly across all clients. Although I had implemented a SyncVar for the player's name, issues with timing and data propagation meant that the names did not reliably appear as expected. After several attempts to resolve this, I opted for a fallback solution where each player is simply labeled as "Player 1", "Player 2", and so on, depending on their connection order. This workaround ensured visual consistency and allowed us to move forward.

4. Fourth Part of the game development - UI for Multiplayer

During multiplayer UI development, I was responsible for designing the status and preview panels that appear during the character selection phase and in-game. This meant working with Unity's UI system to create components that updated in real-time based on player actions. I implemented preview panels showing each player's character, current status, and selection readiness. These elements needed to be synchronized and reactive to input, so I had to develop logic to track local and remote player state changes without introducing lag or UI flickering.

5. Fifth Part of the game development - Issues with multiplayer

Later in development, we encountered issues with authority handling on networked GameObjects. For example, some objects were not responding to commands or SyncVar changes due to ownership misconfiguration. I had to revise the player prefab structure and ensure that components responsible for input and state changes were owned by the correct connection. This included separating visual components from authority-sensitive scripts and using checks like isOwned and isServer to gate interactions appropriately.

As the project evolved, we also added more variety to the characters. Each character had unique traits and needed different visual previews and gameplay values. I worked on the sprite swapping logic and character loading system during the selection phase. This system had to correctly instantiate the preview character, assign the right animations, and update the UI elements dynamically as players navigated through the selection options.

6. Conclusion

We also discussed implementing spells and abilities that players could unlock through achievements. While this feature was not fully completed, I was involved in early design discussions about how to structure ability slots, cooldown systems, and player progression logic.

By the final stage of the project, I had a much clearer understanding of Mirror's systems, including command functions, target RPCs, and the importance of properly synchronizing GameObjects in different scenes. Compared to my earlier attempts with multiplayer in Unity, where I struggled to understand even basic features like SyncVars and NetworkManager, I was now able to build functional multiplayer scenes with working UI, item systems, character selection, and real-time updates.

This project significantly improved my skills in Unity, especially in areas like UI design, network programming, and team coordination. It also reinforced my understanding of C# beyond basic scripting, pushing me to write modular and scalable code. Working in a team also helped me become more flexible, as I often had to adapt to changes, integrate feedback, and shift tasks based on group needs.

End The Bully taught me both the creative and technical sides of game development. From sprite creation to network management, I was involved in nearly every layer of the project. Despite a few incomplete features, the experience was both intense and rewarding, giving me a solid foundation for future game development work.

5.2 Noël

From the earliest brainstorming sessions, we knew that EndTheBully would require close collaboration, thorough planning, and constant iteration. We divided the work into logical components: core gameplay mechanics, UI and menus, audio systems, character design, enemy AI, and more. I chose to focus primarily on the user interface, audio integration, and in-game systems such as the inventory and pause menu. These components may seem secondary compared to character movement or level design, but they are in fact crucial to delivering a polished and immersive experience for players.

This report will outline my contributions to the project, reflect on the challenges encountered, and highlight the knowledge and skills I acquired throughout the development process. It is not only a technical breakdown of what I built but also an analysis of how these elements contributed to the overall player experience. By documenting both my successes and the difficulties I faced, I hope to demonstrate the depth of engagement required in developing a complete game—and the value of every system, no matter how small, in shaping a cohesive and enjoyable final product.

1. Main Menu Design and Functionality

My initial objective was to create a menu that not only matched the tone of our hack 'n' slash game but also provided a smooth and user-friendly introduction for players. I focused on crafting a clean layout, selecting appropriate fonts and colors that convey a sense of adventure while ensuring clear navigation to the main components: Play, Settings, and Quit.

Technically, this required building responsive scripts for each button. I ensured that the Play button could preload the necessary assets and initiate the game without delay, allowing for a seamless transition into the gameplay scene. This task improved my understanding of Unity's scene management system and the importance of optimization to enhance user experience.

2. Settings Button Development

Although still under construction, the Settings button was implemented with a forward-looking architecture. I designed its script structure to be scalable, allowing future integration of features such as volume sliders, resolution options, and control preferences. While currently a placeholder, this component reflects my growing ability to anticipate future needs and plan for modularity and maintainability in my code.

3. Quit Button Implementation

The Quit button, while simple in concept, required attention to usability and safety. I programmed it to execute a clean exit sequence while minimizing the risk of accidental closure. This included defining a precise interaction zone and testing various user behaviors to ensure reliability. It taught me that even the simplest features require careful planning and validation.

4. Play Button Logic and Testing

The Play button was especially important due to its role in transitioning players from the menu to the core gameplay environment. I worked on ensuring that all necessary resources were preloaded before the scene switch, which reduced wait times and prevented in-game hitches. I performed stress tests by simulating rapid and repeated inputs, and refined the transition to maintain immersion and performance.

5. Health Bar Implementation

Most recently, I completed the design and integration of the Health Bar, a critical UI component that provides players with real-time feedback during gameplay. I aligned its visual style with the rest of the UI, ensuring it fits seamlessly with the overall aesthetic. This task helped me practice combining design consistency with functional clarity, a balance I now understand is key in game interfaces.

After completing the main menu and health bar, I shifted focus to the core in-game interface, expanding the user experience beyond menus.

The Settings menu is now fully functional, allowing players to adjust sound settings. This serves as a foundation for more advanced configuration options in the future.

The Stat Menu has been redesigned to display player statistics in a clear, concise layout. I added icons and acronyms to improve readability, and the menu can now be toggled at will—giving players immediate access to information without interrupting gameplay.

I also developed the Inventory UI, allowing players to view and manage their items. This screen is accessible via a button and designed to be expandable for future content.

6. Inventory System Implementation

To support the inventory UI, I created a modular and scalable inventory system composed of three core classes:

1.ItemData – Stores essential information such as item names, sprites, and icons, enabling visual representation in the inventory. ItemData – Stores essential information such as item names, sprites, and icons, enabling visual representation in the inventory.

2. InventoryItem – Manages item quantity, operating like a stack that increases or decreases based on player actions.

3.Inventory – The main inventory manager that stores items in a list and a dictionary for fast access and efficient tracking.

I implemented the following methods:

- Add(ItemData item) – Increases quantity if the item exists; otherwise, adds it as a new entry.

- Remove(ItemData item) – Decreases quantity or removes the item entirely if the count reaches zero.

7. Advancements in UI and Audio Integration

Building on the foundation of the inventory system, I transitioned into refining and expanding the overall user interface and game experience.

To begin, I shifted my focus to the Settings Menu, implementing a sound manager to allow dynamic control over audio levels. This marked a crucial step in improving player immersion and providing greater customization options. In parallel, I continued enhancing the Stat Menu, not only improving its functionality but also redesigning its layout and visual elements to make it more appealing and readable. Icons, spacing, and visual effects were adjusted to create a cleaner, more modern look.

Next, I tackled the design of the Inventory UI. I introduced a dedicated panel and item slots, preparing the system for future item integration. To ensure ease of access, I configured the inventory interface to be toggled with a single button press, maintaining a fluid gameplay experience.

In my effort to improve the game's overall visual coherence, I contributed new sprite assets to the interface. These additions helped to make the UI feel more alive and visually engaging, contributing to the unique identity of EndTheBully.

I then focused on transitions within the game by developing a fade-in / fade-out effect for scene changes. This effect was later replaced with a more practical and thematic approach: a level name overlay that appears when entering a new area, providing clear contextual feedback to the player and enhancing immersion.

8. Audio Design and Sound Integration

Sound design became a major focus of my work during this phase. I implemented a variety of sound effects to bring the game world to life:

- Player actions now include walking sounds, sword swing effects, and an unsheathing sound when drawing the weapon.

- Enemy behavior has also been enhanced with attack sounds to make encounters feel more dynamic and responsive.

- I added background music that plays during gameplay, setting the tone and atmosphere of the game.

This integration of sound elements significantly enriched the sensory experience of EndTheBully, making interactions more satisfying and environments more immersive.

9. Pause Menu and Final Enhancements

Continuing with interface improvements, I developed a Pause Menu that provides players with essential options during gameplay. From this menu, players can:

- Pause or exit the game

- Toggle between windowed and fullscreen mode

- Adjust the volume of both music and sound effects independently.

To add polish and a sense of smooth interaction, I also implemented opening animations for both the Stat Menu and Pause Menu. These subtle visual transitions contribute to the professional feel of the game and help the interface feel cohesive and responsive.

5.3 Hugo

1. Pre-game development and brainstorming period

When I first heard about the project I was really excited about it, because it was the opportunity to do a big project in collaboration with people I liked. That is why at the beginning of the project I was really active during our meetings and brainstorming sessions.

I was always doing suggestions, imagining items or concepts and proposing them to the team to discuss them and see whether they would fit the game or not.

This is how I imagined a lot of items, even if sadly they could not find their way to our final version because of the lack of time and the complexity of their different effects.

Due to my implication and my natural leadership, I was chosen as the project manager, a choice that later revealed itself as not very judicious.

When we assigned the different tasks that we had defined, I willingly chose to do the multiplayer aspect of the game, as I thought it could be an interesting part of the development that I wanted to be a part of. At the time, Noé warned me about the complexity of the task but I did not really think it through and still wanted to do it.

2. First part of game development

As mentioned before, in the beginning I was in charge of the multiplayer aspect of the game.

I first looked for an appropriate engine to use, and I quickly found Photon as an answer.

Following this, I watched some YouTube tutorials concerning it and then tried to implement the same thing in our game, but I faced a problem : the development of the game had already begun and we already had a part of the game implemented and I had to adapt to it to make Photon work.

At this moment it tried finding tutorials about games that already had a base before implementing the multiplayer but that was not a simple task as a lot of those tutorials did not apply to our case or they just did not exist.

At the time I started to lose some of the interest I had in the project due to the difficulty of finding a way to make it work.

Thankfully, Noé came to rescue me and proposed another engine, Mirror. As I did not have the motivation to do the multiplayer part anymore, he took over which allowed me to concentrate on other tasks.

3. Second part of game development

The other task I had was to take care of the items. As we imagined a lot of different items with very various effects, I had to think a lot to make a reusable class that would englobe all the items, whatever their effect was.

This is how I came up with the "Items" abstract class, with enums for the rarities of the items and what stat they would upgrade, the function that is used every time a player opens a chest and also a "Display" and an "Effect" function, that had to be defined in the classes inheriting from the "Items" class.

The first items I implemented and adapted to the class were the "simple items", like the ones that were already existing in the game. Those "simple items" are the ones that only add a certain value or multiply the stats of the player.

At this point of the development, we realised that the interface of the chests had a bug and in consequence I spent some time fixing it, which I finally achieved.

Once this was done, I implemented a function to alter the drop rates of the different items, depending on their rarity.

Then, my main focus was to implement more complex items. I ended up creating three, and one of them implemented the concept of "luck" in the game, changing even more the drop rates of the items.

5.4 Alex

In this game development project, my role focused on building and integrating the player's gameplay and combat systems, which are central to how users interact with and experience the game. These systems include attack mechanics, weapon handling, combat feedback, stat-based differentiation, and animation blending. My objective was to create a responsive, dynamic, and enjoyable combat system that also supports strategic variation through character identity and dual weapon options. The project was developed in Unity, using C# as the primary scripting language.

1. Combat System and Character Design

The game features three main characters: the Elf, the Knight, and the Dino. Each has distinct attributes and combat styles, designed to encourage varied playstyles. A key design feature is that every character uses two weapons: one melee and one ranged. This dual-weapon approach allows players to adapt in real-time to different combat scenarios, switching between ranged precision and close-quarters damage. This also increases player agency, making combat more fluid and strategic.

2. Detailed Character Profiles

The Elf is a character tailored for players who favor speed and agility in their gameplay. Her combat strategy revolves around quick movements and accurate long-range attacks. She uses a dagger for her melee weapon, which delivers fast, low-damage strikes that are particularly effective in hit-and-run engagements. For ranged attacks, she uses a bow that launches arrows aimed via the mouse cursor. These arrows are implemented using Unity's Rigidbody2D physics system, providing a realistic and responsive projectile trajectory. Statistically, the Elf boasts high movement speed, moderate attack power, and low health, which encourages a playstyle reliant on reflexes and evasion. Her animations are handled through Unity's blend trees, allowing for seamless transitions between running, idle, and attack states, making her gameplay feel smooth and natural.

The Knight represents the strength and endurance archetype. He is built to withstand heavy damage and control the battlefield through powerful melee attacks. Equipped with a massive sword as his melee weapon, the Knight delivers sweeping strikes that cover a broad area and apply knockback to enemies using Unity's AddForce method. For ranged combat, he uses a firearm that shoots bullets toward the player's cursor. These bullets are instantiated as prefabs and automatically destroyed after three seconds to prevent memory overflow. His stats emphasize high strength and defense but come at the cost of movement speed, making him a perfect choice for players who prefer a more deliberate, tank-like playstyle. His animation system is managed with separate animator controllers for melee and ranged modes, with weapon switching handled by adjusting animation layers dynamically.

The Dino is a balanced character designed to offer a mix of offense and defense, appealing to players who like to shift between aggressive and tactical gameplay. His melee weapon is a hammer that inflicts area-of-effect damage and heavy knockback. For his ranged attack, he throws rocks that follow a parabolic arc, calculated through custom physics scripting to simulate a lobbed projectile. Stat-wise, the Dino has balanced health and movement speed, but he delivers high melee damage, positioning him as a reliable all-rounder. His gameplay encourages players to manage shared cooldowns between the hammer and rock-throwing abilities, promoting a more thoughtful and strategic combat rhythm.

3. Weapon Switching and Combat Flow

Weapon switching is performed via a key toggle system. When the switch occurs:

- The active weapon GameObject changes.

- UI icons update.

- Character stats and animations adjust dynamically.

This system ensures seamless transitions and responsive gameplay. Melee weapons use hitboxes and trigger detection, while ranged attacks rely on raycasting or directional prefab instantiation.

4. Knockback and Visual Feedback

Knockback was crucial for enhancing combat feedback. When enemies are struck, they are pushed based on the weapon used and the direction of impact.

- The Knight's sword and Dino's hammer have high knockback values.

- The Elf's dagger causes minimal knockback.

- Rigidbody2D forces are used to create dynamic reactions.

To reinforce feedback, visual elements like screen shake, particles, and audio cues were added. These help the player feel the impact and intensity of combat, improving immersion.

5.5 Louise

Over the course of this project, I was responsible for a wide range of tasks related to both the gameplay mechanics and level design. My work focused primarily on enemy behavior, animations, level structure, and boss implementation. These components are essential to the gameplay experience, and I took great care to ensure that every element I developed was coherent, polished, and integrated smoothly into the final version of the game. Below, I describe in detail the main features I implemented, the challenges I encountered, and the solutions I developed.

1. Level Design and Boundaries

Another key contribution of mine was the design of the first and second levels of the game. I was responsible for laying out the level structure, placing terrain elements, and defining navigable and blocked zones. This involved working closely with the team to ensure consistency in visual style and difficulty progression between levels.

It was not easy because I had to create textures, which is something I had never done before. I did my best but I still remain unsatisfied with how the textures of the second level look.

To improve the gameplay experience, I also implemented MapBounds and CameraBounds in both levels. This ensured that players could not leave the intended playable area and that the camera would follow them smoothly while respecting the designed limits of the map. This was particularly important for preserving immersion and preventing bugs related to off-screen enemies or assets.

2. Implementation of the Enemy “Puddle”

One of my major tasks was the complete design and integration of an enemy character named Puddle. This enemy appears early in the game and serves as an introductory challenge to the player. I handled every aspect of Puddle’s creation—from its visual identity to its behavior in the game environment.

1.1. Sprite and Animation Work

The visual design of Puddle required a series of custom-made sprites to represent its different states. On December 31st, 2024, I created the front, left, and right idle sprites to match the aesthetic of the game while making sure the enemy could be easily identified by the player from any direction. Later, I added attack sprites for both left and right directions. These sprites were essential for building a believable and visually responsive enemy.

During a second phase on January 12th, 2025, I focused on the animation work. I designed separate animations for idle, running, and attacking states, ensuring that they played smoothly and transitioned naturally depending on the in-game situation. The animations were then integrated into Puddle’s Animator, and I created the methods runningAnim, iddleAnim, and attackAnim to switch between them based on Puddle’s behavior.

1.2. Enemy AI and Behavior Scripts

To bring Puddle to life in the game world, I implemented several AI scripts. Initially, I wrote a script called AIChasePuddle.cs, which allowed Puddle to detect and follow the player. A key feature of this script was the ability to change the sprite dynamically depending on the player’s position, giving a more immersive and reactive feel to the enemy.

Later on, to improve modularity and make future enemy creation easier, I refactored the AI behavior by creating a base Ennemy class. The Chase() function, previously written specifically for Puddle, was moved into this class to serve as a common method that other enemies could inherit and override if needed.

Originally, attack behavior was handled in a separate script (AIAttackPuddle.cs), but I later removed this script and integrated the attack handling directly within Puddle’s animation and behavior management methods, providing a cleaner and more scalable system.

3. Boss Creation

One of my most significant contributions was the creation of the game's first boss, named Biggle. Biggle appears at the end of the first level and serves as a major milestone for the player. The design process included both visual and mechanical components.

3.1. Visual Design and Animation

I created the complete sprite set for Biggle, including its running and attacking animations. As a boss, Biggle required more detailed and expressive animations than regular enemies. I focused on making the boss's movements feel weighty and threatening, helping to set it apart visually and thematically from the rest of the enemies.

3.2. Ranged Attack Mechanic – Spit Projectile

Biggle features a special ranged attack: it can spit projectiles at the player. This mechanic introduces a new level of challenge, as it forces the player to not only engage in close combat but also to dodge incoming attacks from a distance. I implemented the projectile logic and ensured that it inflicted appropriate damage to the player upon impact. This required integrating damage detection, cooldowns, and an attack pattern system to keep the fight balanced but difficult.

4. Technical Challenges and Problem Solving

Throughout my contributions, I encountered several technical challenges. One major challenge was the coordination between animations and behavior scripts, particularly when it came to synchronizing Puddle's attack triggers with the animation frames. After experimenting with various solutions, I implemented a state-checking system that ensured animations would not interrupt each other and that behavioral actions would only execute when appropriate.

Another challenge was designing a scalable enemy system. Initially, each enemy had unique scripts, but this quickly became unmanageable. Refactoring the codebase to use an inheritance-based enemy structure with a shared base class significantly improved both the readability and extensibility of the code.

5. Git Management and Branch Merging

In addition to my development and design tasks, I was also in charge of managing the merging of branches on our Git repository. This responsibility was crucial to the project's success, as our team worked on separate features in parallel and frequently pushed updates to different branches. Ensuring a smooth integration of these changes required vigilance, consistency, and a good understanding of version control systems.

Throughout the project, I was the one who handled the integration of work across all team members, especially during key development milestones. Each time we had to consolidate new features or prepare a new build, I took care of merging the active branches into the main development branch. This was not a simple task—conflicts often occurred when different team members edited the same files, and resolving these issues required a detailed understanding of everyone's code.

One of the most challenging aspects of this role was dealing with merge conflicts, which became more frequent as the project grew in complexity. Sometimes, changes to the same line of code or overlapping modifications to animations, prefabs, or scene files would result in errors that Git couldn't resolve automatically. I had to manually inspect differences, understand the intent behind each change, and carefully merge them without breaking functionality.

Although this process could be tedious and frustrating at times, especially when last-minute commits introduced unexpected issues, it taught me a lot about collaboration workflows, conflict resolution, and the importance of keeping a clean and organized commit history. I also learned how to use tools like GitKraken and command-line Git utilities more effectively to track changes and avoid potential issues before they happened.

Ultimately, my work on Git management played a crucial behind-the-scenes role in keeping the project running smoothly. It ensured that everyone's progress was reflected in the main project without delays, and it helped avoid larger structural bugs that could have arisen from unmanaged code conflicts. It also allowed us to maintain regular backups, track our development history, and revert changes when necessary. This experience gave me a much deeper appreciation for version control systems, and I feel far more confident handling collaborative coding environments in future projects.

Conclusion

6.1 Personal Reflections

Hugo - What I Would Do Differently

In my opinion, if I had to do it again I would put my attention on the multiplayer before anything else and then build the game around it. I would implement some sort of protection around our main branch in Git too as this has led us to the loss of some time. But more importantly, I would try to imply myself more and more constantly as my implication during was very intermittent and not very constant.

Louise - What I Would Do Differently

If I were to start this project again, there are several things I would do differently. First, I would invest more time early on in designing a clear and flexible architecture for our codebase. Our initial approach, where each enemy had its own unique script, made maintenance harder down the line. Refactoring into a shared base class helped, but this could have been implemented from the start.

Second, I would push for more frequent integration between branches. Some of the most frustrating merge conflicts occurred because we let branches drift too far apart before merging. Regular, smaller merges would have made the process smoother and would have helped us catch compatibility issues earlier.

Finally, I would encourage even more cross-team communication, especially when working on interconnected systems like UI, animation, and AI. Sometimes small changes in one area had unexpected effects elsewhere, and better coordination could have prevented bugs or confusion.

6.2 Skill Acquired

Louise Part

Technical Skills

On the technical side, one of the most valuable areas of growth was my proficiency in Unity and C scripting. I became comfortable with structuring reusable and maintainable scripts, particularly when working with enemy AI and animation systems. Implementing the behavior of enemies like Puddle, and especially designing a boss like Biggle with attack patterns and projectile logic, helped me better understand state machines, animation blending, and interaction between game objects. I also learned how to use Unity's Animator Controller effectively to synchronize animations with behaviors, something I had little experience with before this project.

Additionally, I became much more proficient in using Git, particularly in a collaborative environment. I took charge of branch merging, and while this task was sometimes frustrating—especially when facing repeated merge conflicts—it forced me to develop a disciplined approach to version control. I learned how to identify, understand, and resolve merge conflicts carefully, which will be invaluable in any future team project. I also improved my knowledge of debugging techniques, scene structuring, and level design, particularly through setting up map and camera bounds.

Non-Technical Skills

Equally important were the non-technical skills I developed. Collaborating within a team for nearly a year required strong communication, patience, and adaptability. Using tools like Discord and Notion taught me the importance of keeping our work transparent and organized. These platforms allowed us to avoid confusion and ensure that everyone was aligned with the project's goals.

I also improved my time management and problem-solving mindset. Balancing deadlines, meetings, coding, and testing taught me how to break large tasks into smaller, manageable parts and how to prioritize them based on importance and urgency. Handling stress—especially during the most chaotic development phases—helped me grow personally as well, and I learned not to panic in the face of bugs, crashes, or last-minute changes.

Hugo's Part

During this project, I have perfectioned my understanding and use of the language C. This experience has allowed me to discover an develop other ways of thinking and it has taught me the importance of constance and implication. Even if I had already worked on projects or expositions with friends before, this project was really different as it was during a long period of time and much more complex and I think it has taught me a lot about working in a team, management and organisation.

6.3 Final Conclusion

After nine months of continuous work, this video game project represents the culmination of our collective efforts, creativity, and perseverance. Starting from a structured specification document and a clear set of objectives, we navigated through every phase of the development cycle—from early concept discussions to the delivery of a fully functional multiplayer game. Along the way, we encountered numerous technical, organizational, and interpersonal challenges that tested both our individual abilities and our teamwork, but ultimately contributed to a richer and more realistic learning experience.

This final product reflects not only the application of our technical knowledge, but also our ability to collaborate, adapt, and problem-solve in a real-world project setting. While some features had to be simplified or dropped due to time constraints or unforeseen complexity, the game still meets its core goals and offers a satisfying and engaging user experience. The process allowed each of us to deepen our understanding of game development, project management, and version control, while gaining hands-on experience with tools like Git and Unity.

Most importantly, this project taught us how to work as a team—how to plan, communicate, and deliver something ambitious together. It pushed us beyond academic exercises and into a context that closely mirrors professional environments. We leave this project not only with a working game, but with confidence in our capacity to face new technical and collaborative challenges in the future. This report marks the end of the project, but the skills and lessons we've acquired will carry forward into whatever comes next.