0r!g!ns

Technical Report n°3
Final Report

A1

EPITA 2030 Promo

CERIBAC Noah
BELIARD Antonin
TONNOT Antoine
GLEMET Adam

Document Version: V1.2.0
Game Version: V 1.1.0

Table of Contents

Introduction

This final technical report aims to present the final state of the Or!g!ns team project, Ascendant Trials.
During the last months, our group continued and completed the development of the game. Since the previous defense, the project has become much more concrete.
The levels, sections, and mechanics are now implemented.
This progress was also made possible thanks to a reorganization of the project.
Each member became responsible for one of the remaining sections, with the goal of implementing its levels and mechanics individually, while still being able to ask the others for help when needed. This new organization made the work clearer and allowed everyone to focus on a precise part of the game.
This report will first present general information about the project and the team. Then, it will introduce the game itself, its concept, and its objectives. After that, we will summarize the previous technical reports in order to show how the project evolved during the year. The main part of the report will present the work done by each member of the team. Finally, we will explain the tests, the difficulties encountered, the differences with the initial planning, and our final conclusion about the project.

General Information

I. Team Members

The team was initially composed of 5 members being: CERIBAC Noah, BELIARD Antonin, TONNOT Antoine, GLEMET Adam and finally MAILLOT Marine.

Marine left the group between the defense 1 and 2; she has not been working on the game since then.

Here are the members of the 0r!g!ns group who worked on this report and on this project.

CERIBAC Noah

I have always been passionate about computer science and new technologies, which is why I chose to study at EPITA.

The year-long academic project offered by the school allows me to explore new fields and to work under real professional conditions, within a domain I particularly enjoy.
Despite my previous experience in video game development, notably using Lua, I continue to learn every day through this Python-based project developed with the Ursina engine.

BELIARD Antonin

My name is Antonin Beliard, and I'm a first-year EPITA student. I've been building tech projects for years, including robotics, small web tools, and mobile apps; so I'm comfortable turning requirements into something that actually runs.

TONNOT Antoine

I'm Antoine TONNOT and I am a first-year student in EPITA. I love coding and co-op game. I also regularly play piano. I started coding in Python and then moved to C# a few years ago. I started coding during the 3rd year of high school.

GLEMET Adam

I am Adam Glemet, a first-year EPITA student. Since I discovered web development for my ninth-grade internship, I've always been interested in computer science because I already had a passion for video games, so I took NSI specialty in high school where I could develop alone small one-month project in Python, a website group project in HTML/CSS.

II. Specificities of the game

At the beginning of the year we were asked to complete a CDST, to define the type of game we were going to create and what features we would have to code.

Here is a screenshot of this CDST, showing what has been decided in the beginning of the Year and of this project.

Type of game :				
Action/Adventure	Battle Royale	Beat them all	fighting	Simulation
FPS	MMORPG	MOBA	Party Games	Survival Horror
Platformer	Puzzles	Reflection	Rogue Like	TPS
RPG	RTS	Sandbox	Shoot them up	Racing
Others :				
General features of the game :				
IA :	Wander	Attack	escape	"Path Finder"
Multiplayer :	Cooperative	Battle (2-4)	Massive	
Network :	P2P	Lan	Online	
Graphic features :				
Dimensions :	2D	3D	Others :	
Special features :	Stereoscopy	AR	VR	
graphics :	Personal	Custom	Existing	
Details :				
Sound characteristics :				
Music :	Personal	Custom	Existing	
FX :	Personal	Custom	Existing	
Details :				
Other features :				
Website :	Personal	Custom	Prefabricated	

III. Initial Task Distribution

In the beginning of the year we were also asked to divide our Game Project into several Tasks and to split up those Tasks between the group members.

Here was our Task Distribution at the beginning of the game Project.

Tasks	noah.ceribac	antonin.beliard	antoine.tonnot	adam.glemet	marine.maillot
Basic Logic	R	S	-	-	-
Level Mechanics	-	-	S	R	-
NPC AI	-	-	S	-	R
Loot System	S	R	-	-	-
Tower/Dungeon System	R	-	S	-	-
Sandbox Mode	-	-	-	-	-
Networking	S	R	-	-	-
Website	-	-	-	R	S
Level Design	-	-	R	S	-
Character Design	-	-	-	S	R
Texturing	R	S	-	-	-
Animation	S	-	-	-	R
GUI	-	R	-	S	
SFX/VFX	-	-	R	-	S
Bug Test	R	R	R	R	R

IV. Final Task Distribution

As previously said, MAILLOT Marine left the group in the middle of the project, this meant we had to rethink our game and mainly the features we wanted to add in our game.

So we have created this tasks distribution table and we also changed some tasks goals but we will talk about that later on this report.

Tasks	noah.ceribac	antonin.beliard	antoine.tonnot	adam.glemet
Basic Logic	R	S	-	-
Level Mechanics	S	S	S	R
NPC AI	S	S	R	-
Loot System	S	R	-	-
Tower/Dungeon System	R	-	S	-
Sandbox Mode	-	-	-	-
Networking	S	R	-	-
Website	-	-	-	R
Level Design	S	S	R	S
Character Design	-	-	-	R
Texturing	R	S	-	-
Animation	R	S	-	-
GUI	S	R	-	-
SFX/VFX	-	-	R	-
Bug Test	R	R	R	R

And finally for the Tasks Distribution and Tasks Planning, here is our Task progress table, showing where each task should be depending on the current Defense State.

Task progress table			
Tasks	Report 1	Report 2	Final Report
Basic Logic	100%	-	-
Level Mechanics	25%	50%	100%
NPC AI	25%	50%	100%
Loot System	25%	50%	100%
Tower/Dungeon System	50%	100%	-
Sandbox Mode	-	-	-
Networking	95%	100%	-
Website	50%	100%	-
Level Design	10%	50%	100%
Character Design	75%	100%	-
Texturing	50%	100%	-
GUI	5%	50%	100%
SFX/VFX	0%	50%	100%
Bug Test	100%	100%	100%
Veillez indiquer le pourcentage d'avancement prévu à la date de la soutenance			

The Project

I. The Origins of the Project

The origin of this project first came from the year-long group assignment we had to complete as part of our studies.
Once the group was formed, we had to look for a gameplay idea that could be both interesting and achievable within the project constraints.
At first, we considered creating a game based around blackjack, with a gore and horror atmosphere. The concept included bonus and penalty cards, and the idea was to design a small, simple game playable either solo or in a one-versus-one mode.
The visual and narrative direction was meant to be dark, obscure, and unsettling.
However, the announcement that Godot could not be used, even with a plugin allowing Python programming, forced us to redirect the project toward a simpler and more suitable technical solution. We therefore decided to use Ursina, a game engine based on Panda3D. Although it is less known than PyGame, Ursina offers easier support for the type of game format we wanted to develop.
After this change, we had to find a new game idea. During a voice meeting, Antonin shared an idea inspired by a game of Pico Park he had played with his girlfriend the day before. From this initial concept, we progressively added our own identity to the game, such as a tower system inspired by manhwas and mythological themes brought in by Marine.
Once the game concept was clearly defined, we established the number of levels and the main themes of the adventure: the Underworld, the Abysses, Earth, the Sky, and Space. This structure then allowed us to start designing the levels and the individual mechanics associated with each part of the game.

II. Presentation of the Game Idea

As explained previously, Ascendant Trials is a platformer inspired by Pico Park, but with its own identity added to it through a mythological twist and a tower-based progression system.
The game is divided into five main sections, each one based on a different theme. Each section is itself divided into five levels, which means that the full tower is structured around a total of twenty-five levels. The five themes are the Underworld, the Abysses, Earth, the Sky, and Space.
Each section also introduces mechanics linked to its environment. For example, some levels include low-gravity mechanics, lava-based obstacles, swimming phases, or other elements that force the players to adapt their way of moving and solving puzzles. This allows each part of the tower to feel different while still keeping the same cooperative platformer base.
Originally, the game was supposed to include a boss fight at the end of each themed section. Each boss was planned to match the mythology and atmosphere of its section. However, because of the lack of time and following Marine MAILLOT's departure from the group, we had to rethink this part of the project and simplify some features.
As a result, the weapon system was replaced by a key system, and the boss fights were replaced by doors. Instead of defeating a boss with weapons, the players now have to complete the section, find or obtain the required key, and use it to open the door leading to the next part of the tower. This solution keeps the idea of progression between sections while making the game more realistic to finish within the time we have left.

Previous Report Summary

I. Summary of Technical Report No°1

The first technical report introduced the foundations of the team project and the first real version of Ascendant Trials. At this stage, the project was still at the beginning of its development, so the report was mostly focused on explaining the concept of the game, the organization of the group, and the first technical choices that had been made after the validation of the functional specification document.

The report first presented the general context of the project. Ascendant Trials was designed as a cooperative puzzle-action platformer, inspired by Pico Park, but with a more personal identity. Instead of simply copying the original inspiration, the group wanted to create a game based around a vertical tower divided into several themed sections. Each section was planned to be linked to mythology, especially Greek mythology, and to introduce different mechanics and challenges. The goal was to make a game that would be easy to understand for players, but still interesting through cooperation, timing, communication, and puzzle-solving.

A large part of the first report was also dedicated to the presentation of the team. Each member of the group had a defined role in the project. Marine MAILLOT was project manager and worked mainly on NPC AI, character design, animation, and mythological inspiration. Antonin BELIARD was responsible for networking, the multiplayer lobby server, and part of the infrastructure around the website. Noah CERIBAC worked on the basic game logic, the codebase, the tower system, and textures. Antoine TONNOT was mainly in charge of level design, game mechanics, and sound and visual effects. Adam GLEMET focused on the website and level mechanics. This task distribution helped the team organize the project and gave everyone a clear area to work on.

From a technical point of view, the first report showed that the group had already started building the core of the game. The first codebase was created, with the beginning of the player controller, level loader, block creation system, save system, and basic gameplay logic. These elements were important because they created the structure on which the rest of the project would be built. Even if the game was still far from finished, the foundations were already present and allowed the team to start testing simple levels and mechanics.

The report also introduced the first ideas for level design and game progression. The tower was planned to contain five sections: the Underworld, the Abysses, Earth, the Sky, and Space. Each of these sections was supposed to contain five levels, with a boss fight at the end of each section. The bosses and NPCs were inspired by mythology, such as Cerberus, Charybdis, Scylla, the Minotaur, and other mythological figures.

These ideas gave the game a strong identity and helped define the atmosphere of each part of the tower.The visual and design aspects of the game were also discussed. The team started thinking about the general color palette, character design, and visual identity of the project.
The website was also started, even if it was still in an early state. Its goal was to present the game, the group members, and the resources used for the project.
Overall, Technical Report No. 1 was mainly a preparation report. It helped define what Ascendant Trials was supposed to become, both as a game and as a group project. It presented the first ambitions of the team, the technical basis of the project, and the first organization needed to continue development. Even if many features were still only planned or partially implemented, this report gave the project a clear direction and allowed the team to move from a simple idea to a real development process.

II. Summary of Technical Report No°2

The second technical report presented the progress made by the team after the first defense. Compared to the first report, which was mostly about defining the project and preparing the structure, this second report focused more on implementation and on the concrete evolution of Ascendant Trials. The game was no longer only an idea or a set of planned features. Several important systems had started to work, and we had a clearer view of what still needed to be improved before the final version.

One of the most important parts of the second report was the progress made on the multiplayer system. During the first defense, the networking part was already started, but it was still mostly separated from the game itself. In the second report, this system became much more complete and better integrated. The lobby server was improved so that it could manage rooms, players, readiness, colors, and level progression. Players could create or join a room, choose a color, get ready, and start a shared level together. The server was also structured more clearly, with separated modules for the protocol, room management, player sessions, and status monitoring.

The multiplayer system also became connected to the main Ursina game.

A client layer was added to communicate with the server, and a multiplayer runtime was created to manage the in-game session. The game started to support a host-authoritative model, where the host simulates the gameplay while the server manages the room state and relays information between players. This was an important compromise because it allowed the team to have real online cooperation without making the server too complex for the scope of the project.

The second report also showed progress on the gameplay and codebase. Several bugs were fixed, especially bugs related to collisions, player movement, and block interactions. New block behaviors were added, such as damage blocks and progression-related blocks. The system was also improved so that blocks could have multiple behaviors instead of being limited to only one. This made the level design more flexible and allowed the team to create more varied situations inside the levels.

The player model and textures were also improved during this period. The game started to have a more personal visual identity, with textures for the player and some objects. In multiplayer mode, the players could be differentiated visually, especially through color changes.

This was important because the game is based on cooperation, and players need to quickly understand who is who during gameplay.
Level design also progressed in the second report. The structure of the tower became clearer, and the different themed sections were more defined.
The team continued working on the idea of five sections, each with its own atmosphere and mechanics. The Underworld, the Abysses, Earth, the Sky, and Space were still the main parts of the game, and each section was supposed to bring different challenges. Some mechanics were already planned, such as water-related gameplay, dangerous blocks, moving platforms, and other obstacles linked to the theme of each section.
The website was also improved between the first and second reports. It became more useful as a presentation tool for the project, with sections dedicated to the game, the team, and the different resources used. Even if the website was not the main part of the project, it was still important because it was one of the required deliverables and helped present the work of the group in a clearer way.
However, Technical Report No. 2 also showed that the project still had several limits. Some systems were functional but still needed more testing, especially the multiplayer system. Some gameplay features were not finished yet, and the team still had to check if all planned ideas were realistic with the time left. The report made it clear that the project had advanced a lot, but that the final version would still require simplification, bug fixing, and better integration between all the systems.

Overall, Technical Report No. 2 marked an important transition in the development of Ascendant Trials. The project moved from a mostly planned concept to a concrete playable base. The multiplayer system, the gameplay logic, the textures, the level systems, and the website all progressed significantly. This report also helped the team understand which parts of the project were strong enough to keep and which parts would need to be simplified or reworked before the final defense.

Project Advancement

I. BELIARD Antonin

In this last report, my work was mainly around the online side of the project, the public website, the in-game multiplayer layer, the final UI, packaging, and a lot of integration and polish work around the levels. When I talk about game files or sections, it is because I either made that specific pass myself or had to connect it to multiplayer, visuals, respawns, keys, doors, or the final build.

My work started with the first server and website files. It then moved to the private room server, the in-game multiplayer runtime, the lobby, developer tools, final website copy, Windows packaging, and many late fixes. Near the end I also worked a lot on section visuals, backgrounds, textures, movement modes, death/reset behavior, spawn reliability, UI, and player animation. That late work was important because the final version had to feel like one game, not like separate features placed next to each other.

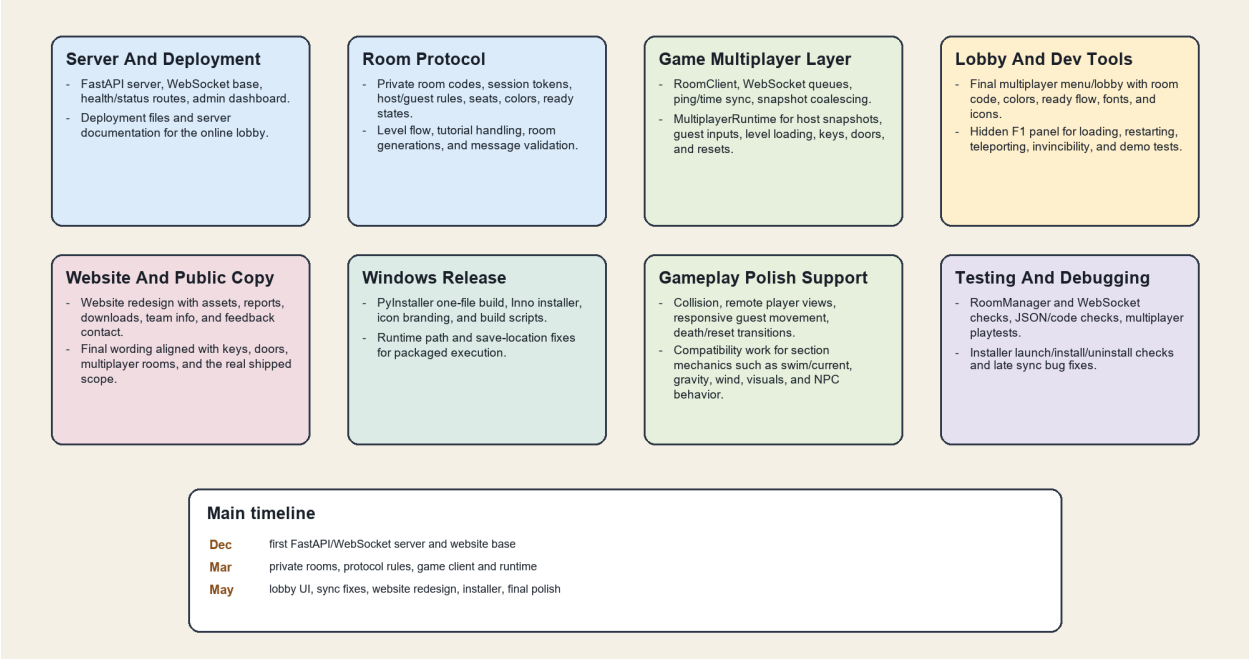


Figure 1 - Main areas of my work. The map separates my networking, UI, website, release, and polish tasks from the main game architecture owned by Noah.

1. Server, Deployment, and Public Website

One of my first large tasks was the server side of the project. In the network repository, I created the first FastAPI server and the first website page. The first version of the server already had the base we reused later: HTTP routes, WebSocket support, health and status endpoints, deployment files, environment configuration, static files, and an admin dashboard.

The dashboard was mostly for development. It gave me a quick way to check whether the deployed server was alive, whether it could answer requests, and whether the process was behaving correctly. It was useful when the server stopped being a local experiment and became an actual online service for the project.

I also handled part of the public website during the year. Earlier in development, I added robots/no-index rules because the website still had placeholder text and unfinished promises. At the end, I redesigned the public site with new assets, icons, fonts, the player visuals, report links, downloads, team information, and a feedback contact. I also updated the content to match the game we could actually present.
That last wording pass was important. The old project plan still talked about some ideas that changed later, such as boss fights, a more weapon-focused progression, or a fully finished level creator. For the final website, I changed the copy so it focused on the real game: private multiplayer rooms, five sections, keys, doors, reports, downloads, and the current playable scope. I wanted the website to support the presentation instead of creating questions about features that were no longer realistic.

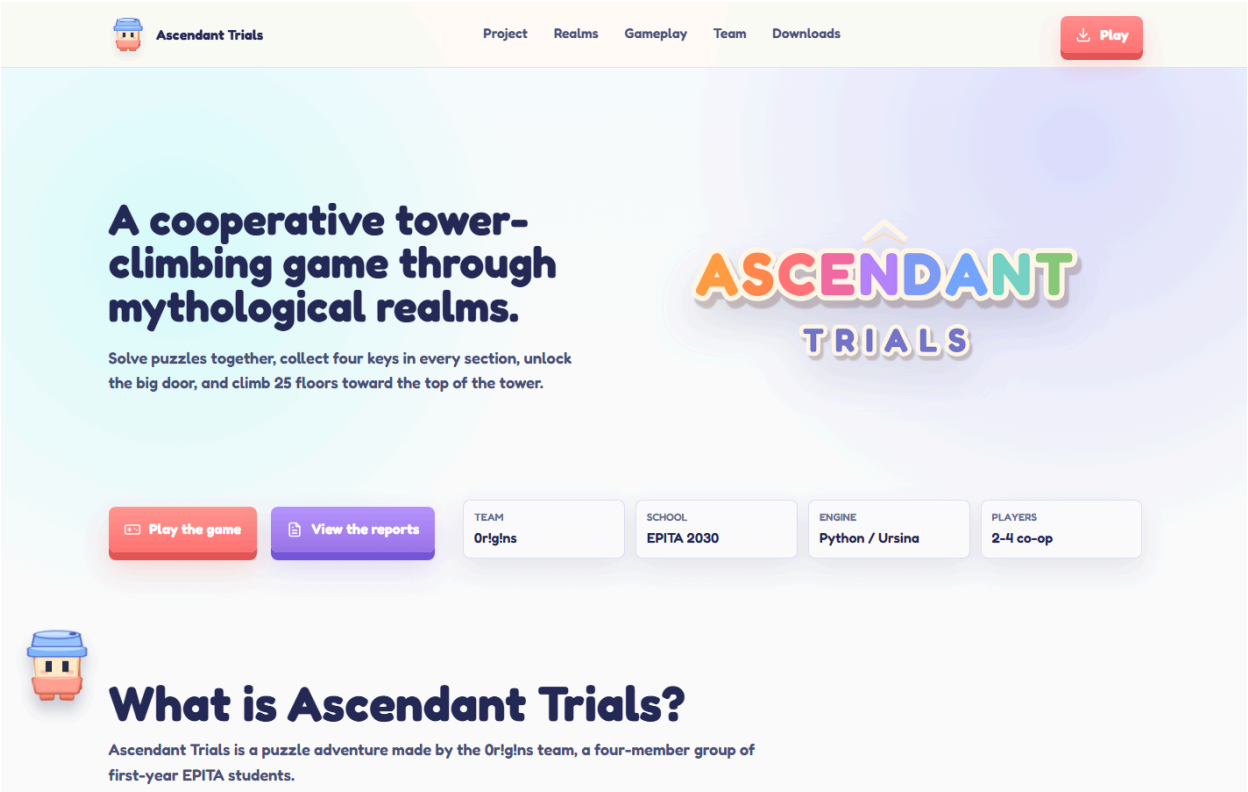


Figure 2 - Public website after the final redesign and copy update.

The website was also part of the delivery work. It gave the jury and external testers a simple place to find the project, reports, downloads, and team information. Even if it is not the game itself, it is still part of how the project is judged, because it is often the first thing someone sees before launching Ascendant Trials.

2. Private Room Server and Multiplayer Protocol

My main networking task was the private room system. The first server only proved that WebSockets worked. Later, I implemented the real room logic with `RoomManager` and `protocol.py`. This added generated room codes, session tokens, host and guest roles, player seats, player colors, ready states, room phases, and level flow.

The server owns the lobby state. It creates the room, lets other players join with a short code, tracks who is ready, starts the game when the host asks, and broadcasts room updates. It also checks the role of every message. Guests can send player input, but they cannot do host-only actions such as starting the game, restarting a level, sending world snapshots, loading a level through the developer panel, kicking players, or completing the level.

I added validation for the playable level range. The tutorial is treated as level `00/01`, while the main game can progress through section 05 level 05. This mattered near the end because multiplayer could not stay blocked at the early demo range. If the final game had five sections, then the server also had to understand those five sections.

I also added room generations. Each major state change increments a generation number: start game, restart level, developer load, or move to the next level. Clients receive the generation in server events and snapshots. If a snapshot from an older generation arrives late, the client can ignore it. This fixed a category of annoying bugs where a room had already changed level but an old gameplay message was still in transit.

The final network model is host-authoritative. The FastAPI server does not simulate the level itself. It manages rooms and relays messages, while the host game client runs the actual gameplay state. A fully authoritative dedicated server would have required moving a lot of Ursina gameplay logic into a separate server simulation, which was not realistic for this project. For a private cooperative game, the host model was the practical solution.

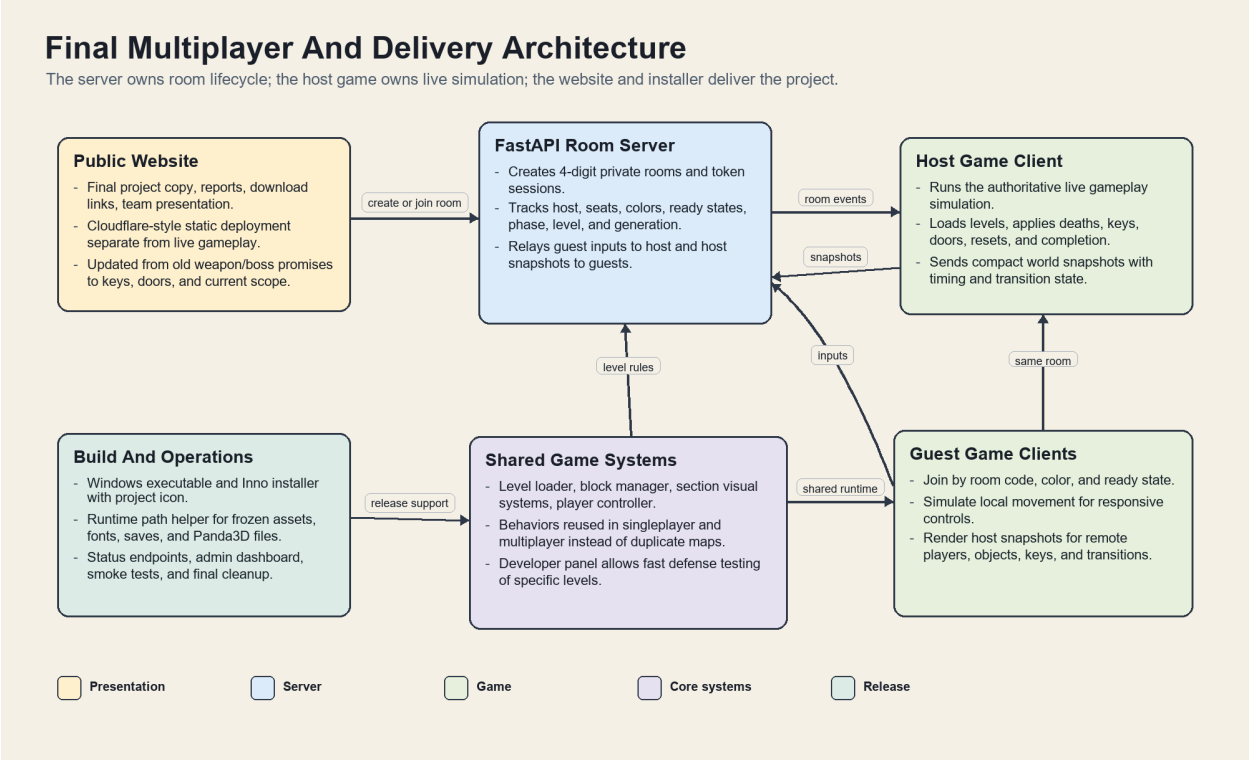


Figure 3 - Final networking overview. The server owns the room flow, while the host game client owns the live gameplay simulation.

3. In-Game Multiplayer, Lobby, UI, and Developer Tools

Once the server was structured, I connected it to the game. I added the multiplayer client and runtime under `scripts/networking/multiplayer/`. The `RoomClient` handles HTTP calls, the WebSocket connection, a background socket thread, incoming and outgoing queues, ping messages, time synchronization, and snapshot coalescing. Snapshot coalescing was useful because movement snapshots are temporary; if the game processes old snapshots, the remote players feel delayed.

The `MultiplayerRuntime` is the layer that connects server events to the Ursina game. It loads the level requested by the server, creates local and remote players, sends guest inputs, applies host snapshots, manages room events, and keeps shared state such as key progress and door transitions. This runtime is also where I had to connect multiplayer to the existing loader, block manager, player controller, section modules, and HUD.
For movement, I used a trusted cooperative model. Guests simulate their own movement locally so controls stay responsive, then send their input and compact state to the host. The host keeps the official gameplay state and sends snapshots back. I did not design this as anti-cheat. It is a private cooperative project, so I focused on responsiveness and stable presentation.

I also worked on the multiplayer lobby and final menu UI. I added the room-code flow, player colors, ready state display, host controls, fonts, icons, layout polish, and later a small performance overlay. The lobby needed to be clear because it is the first multiplayer screen. If players cannot quickly understand how to host, join, pick a color, and start, the rest of the multiplayer system becomes hard to show.

Figure 4 - Final in-game multiplayer lobby with room code, colors, ready flow, and host controls.

The hidden `F1` developer panel was another important tool I added. It can load a specific level, reload or restart, go to the previous or next level, teleport to spawn or endpoint, toggle invincibility, and unlock progression for tests. In multiplayer, host-only level loading and restart go through the server so every connected client follows the same state. This panel saves a lot of time during testing and during the final interview, because we can jump directly to the level or mechanic we need to show.

I also worked on the HUD and progression display when the project moved from the old weapon-fragment vocabulary to the final key and door presentation. Some internal names still use `wpn_endpoint` because that was the earlier code name, but the player-facing system became keys and doors. I updated the UI and multiplayer state around that so the final presentation made sense, while keeping the older internal hooks working.

4. Levels, Backgrounds, Textures, Mechanics, Deaths, and Animations

I did not own every level in the game, but I did contribute a lot to how the final sections look and behave. Some of that work was full level creation, some was visual and texture work, and some was integration so mechanics worked with multiplayer, death resets, spawns, and snapshots.

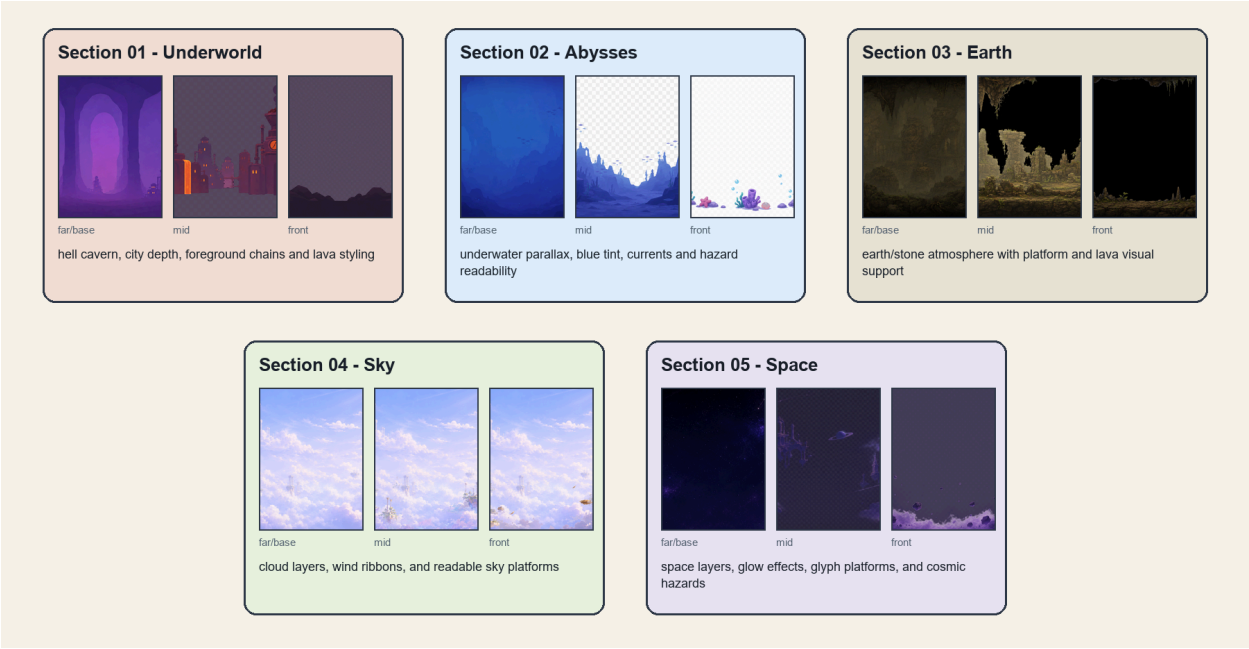


Figure 5 - Background and texture layers used by the section visual systems I worked on during the final content polish.

For section 01, I did not create the original early levels, which were mainly from other teammates. My main Section 01 work was the Underworld visual pass and cleanup. I added a large runtime visual module for the hell theme, with cavern/city/foreground background layers, sparks, wisps, flames, animated windows, lava surfaces, lava flow highlights, bubbles, and generated platform visuals. The goal was to make the first section feel like an actual themed part of the tower instead of only JSON blocks with plain colors.

The Section 01 visual work also covered the platform types used in those levels. Moving platforms, blinking platforms, stable platforms, and lava blocks needed to keep their collision behavior while getting better art. I built visual meshes around the existing blocks instead of replacing the gameplay bodies, so the levels could keep working while looking much better. Blinking platforms also got separate on/off visuals and shimmer/outline states, which made them easier to read during gameplay.

I also cleaned up a lot of temporary Section 01 visual experiments. One commit added many generated assets and demo scripts while I was testing platform and background styles. After I knew what was useful, I removed the leftover demo files and old generated residues. That cleanup matters because a messy repository makes it harder for teammates to know which files are actually part of the game.

For section 02, my contribution was much more direct. I created the five Abysses level JSON files and the underwater pass. Every Section 02 level uses `movement_mode: "swim"`, so the player changes from normal platform movement to free underwater movement. I added support in the player controller for swim mode, swim display scaling, swimming animation, vertical movement, and current forces. I also had to make sure guests in multiplayer could apply local current effects so the controls stayed responsive.

The first Section 02 level introduces swimming and currents. It uses a rightward tutorial current, a downward stream, an exit current, the first key/fragment pickup, and an endpoint. The second level adds a stronger horizontal current, a bubble lift, and a cursed lure object. The third level introduces a whirlpool with an attractor current pulling toward a damage core, plus a tempting lure and an upper exit current. The fourth level uses blinking bite gates as damage hazards, a rushing throat current, and a final attractor pull. The fifth level combines Scylla-style blinking bite hazards with a Charybdis-style whirlpool, updraft current, cursed lure, and final exit current. This section was my biggest level-design contribution because the levels, currents, hazards, pickups, endpoints, movement mode, and background layers were all connected in the same pass.

The Abysses background and texture work used three image layers: far, mid, and front. In the runtime visual module, I added underwater tinting and background movement so the section felt different from the Underworld. I also styled blocks and current/hazard areas so the player could understand what was safe, what was a force zone, and what was dangerous.

For section 03, the level layouts themselves were Antoine's work. My part was the Earth visual pass and the mechanics support around switches and animated platforms. I expanded the Section 03 runtime module with platform switches, toggled platforms, disappearing platforms, and toggled blinking platforms. I also added state capture and apply functions so these mechanics could be synchronized in multiplayer snapshots instead of existing only on the host.

The visual side of Section 03 was also one of my tasks. I added the Earth background layers, stone/platform visuals, darker cave atmosphere, and lava styling for damage floors, pits, and fissures. The lava visuals include moving surfaces, veins, flow meshes, bubbles, and different treatment for deep lava or smaller hazards. This made Section 03 more readable and more connected to its Earth theme without changing Antoine's level design ownership.

For section 04, the history is mixed. My own work was the final Sky pass with wind gust support, five playable Sky level files, visual backgrounds, and the runtime mechanics needed for gusts. The mechanic is called `wind_gust`; it can apply a vector, speed, period, active duration, and offset. That let me create constant gusts, timed gusts, updrafts, tailwinds, headwinds, downdrafts, and crosswinds.

The first Sky level introduces a first updraft and a gentle tailwind with a fall kill zone underneath. The second level uses timed tailwind and headwind lanes, plus an updraft around the key. The third level adds moving damage hazards shaped like bronze feather sweeps, with lower and middle updrafts. The fourth level continues the gust bridges with headwinds, feather sweeps, and a fragment updraft. The fifth level is a wind gauntlet using named gusts like Boreas, Zephyr, Eurus, Notus, and a final updraft, mixed with moving damage sweeps and the endpoint door. I also built the Sky visual module with cloud background layers, wind-ribbon effects, and platform visuals that avoid overwriting doors, pickups, wind zones, or hazards.

For section 05, the original base came from Noah, so I do not want to claim the whole section. My contribution was fixing and integrating the Space section after the merge, then adding the texture/UI/visual pass. I worked on Section 05 level files where needed, especially around gravity buttons, bumpers, low-gravity rules, key pickups, endpoints, and final-door behavior. I also connected those mechanics to the player controller and multiplayer runtime so gravity direction, movement state, and resets would not desynchronize.

The Space visual pass was large. I added space background layers, glow effects, platform cracks, glyph details, star/crescent shapes, wave bands, vertical fades, and platform styling. The section needed to feel different from the previous ones, so I used darker cosmic backgrounds and brighter platform details. On the gameplay side, Section 05 uses gravity buttons to flip movement, low-gravity rules for floating jumps, bumpers to launch the player, and starfall damage zones. My work here was mostly making the section presentable and stable after merge conflicts, not taking credit for every original level idea.

Figure 6 - Later campaign visuals after the final section polish and integration work.

I also worked on the player animation and movement feel. I improved the avatar animation for rising, falling, landing, walking, blinking eyes, swim mode, and gravity flips. The player model is built from parts, so the animation work moves and scales body parts rather than just sliding one square around. I also fixed a problem where the player model could be rebuilt too often, and I worked on `collision_scale` and group bounds so the visible player and the collision body matched better.

Remote players needed the same kind of attention. In multiplayer, a remote player is not just a static block. It needs interpolation, tinting, movement mode, gravity direction, landing animation, swim animation, and a collision proxy that follows the gameplay state. I worked on remote player views and snapshot interpolation so other players did not jitter as much, while still keeping physical interaction usable.
Deaths and respawns were another important part of my final work. I added the iris transition system in `effects_manager.py` and connected it to singleplayer and multiplayer resets. When the player dies, input is disabled, the iris closes around the player, the level attempt is reset while the screen is hidden, pickups and section state are restored, then input comes back. For doors, I added endpoint transition focus and hold radius so the iris can focus on the actual door instead of always using the player position.

Spawns were connected to this work too. Levels define `player_spawn`, and the loader passes that position to the runtime. I also fixed issues around bad spawns by delaying gravity for a frame in the player controller, so the player does not immediately fall or collide incorrectly before the level is fully ready. In multiplayer, reset snapshots also had to include transition state and timing so guests would not keep old visuals, old pickups, or old runtime state after the host reset the level.

5. Packaging, Tests, and Final Limits

Near the end, I worked on the Windows release path. I added PyInstaller support, an Inno Setup installer, an IExpress fallback script, app icon generation, window title branding, and build requirements. I also added runtime path support because packaged Python games do not load files from the same place as the development version.

The packaged game needed a better save location. In development, saving near the project files is acceptable. In a packaged executable, writing inside the temporary extraction folder is not reliable. I changed packaged saves so progress is stored under `%APPDATA%\Ascendant Trials`, which is closer to how a normal Windows game should behave.

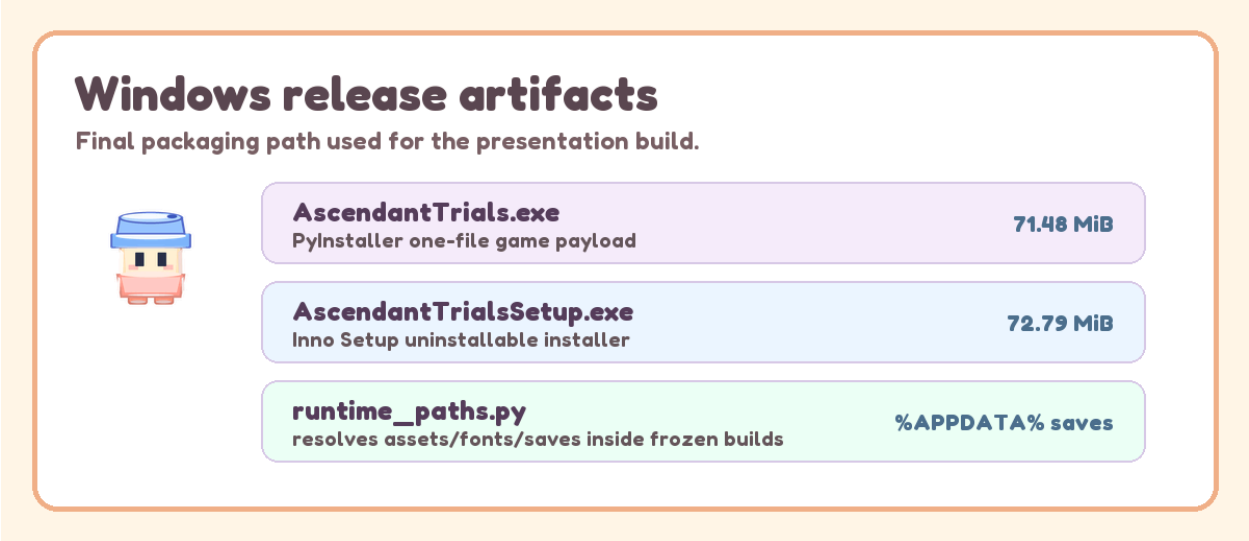


Figure 7 - Windows build and installer work: executable, installer, icon, runtime paths, and save handling.

For testing, I used different checks depending on the feature. On the server side, I tested room creation, joins, host disconnection, colors, ready flow, level loading, restart, tutorial handling, and final level boundaries. On the game side, I used syntax checks, JSON checks, focused launches, manual multiplayer tests, screenshot checks, and repeated reset tests. For the release work, I built, launched, installed, and uninstalled the Windows output.

The final networking system still has limits. Room state is stored in memory, so a server restart loses active rooms. If the host disconnects, the room closes because the host owns the live simulation. The model trusts private cooperative players and does not try to solve anti-cheat. WebSockets over TCP are good enough for our project, but a larger commercial game would probably need a different networking stack.

Overall, my contribution was to make the online, visual integration, and presentation side of Ascendant Trials solid enough for the final defense. I built the room server, connected it to the game, worked on the lobby and developer tools, created the whole Section 02 underwater pass, contributed the visual and mechanic support for other sections, updated the website, packaged the game for Windows, and fixed many of the bugs that only appear once everything is connected. My part was not the whole game architecture or the whole level design. It was the server, multiplayer, UI, release, and final polish work that helped the project become a complete deliverable.

II. CERIBAC Noah

During the first technical reports, I was mainly responsible for the file and folder architecture of the project. I also helped the other members of the group when they had problems implementing features or adapting their work to the existing codebase.
For this final defense, my role changed a little. Antonin took over most of this responsibility, because my schedule did not allow me to manage this part as well as before, especially because of my internship, exams, and other constraints. Because of this, I mostly acted as an assistant on the project. Outside of the additions mentioned below, I mainly gave my opinion, corrected issues, and proposed suggestions for the features and modifications made by the other members of the group.

1. Texture wise

Because the objective of the game changed during development, I had to rework some old textures that I had created before. For example, the five weapon textures became key textures, because the weapons were replaced by keys in the new version of the game.
I also created door textures adapted to each section, so that they could fit better with the theme and visual identity of the different environments.

Figure 1 & 2 - Keys and Doors models

All these textures were made using .json files. To modify them more easily, I used a tool that allows the manipulation of JSON texture files through a graphical interface. This helped me design and adjust the textures without having to edit everything manually in the file.

2. Section 5 implementation

Regarding my personal section, I was responsible for implementing the five levels of Section 5, which is based on space.
For this section, I implemented three unique mechanics.
The first one is an inverted gravity system. When one or several players press the button, the gravity of the level is reversed. The players then end up either on the ceiling or back on the floor, depending on their previous position.

Figure 3 - Gravity Switch

The second mechanic is a low gravity system. Since the section takes place in space, the gravity is weaker than on Earth. Some levels are therefore based on this mechanic, where the player moves and falls more slowly.

The last mechanic is a bumper. It is a small platform that makes the player bounce when they jump on it. This mechanic was useful to create more vertical movement and to make the space levels more dynamic.

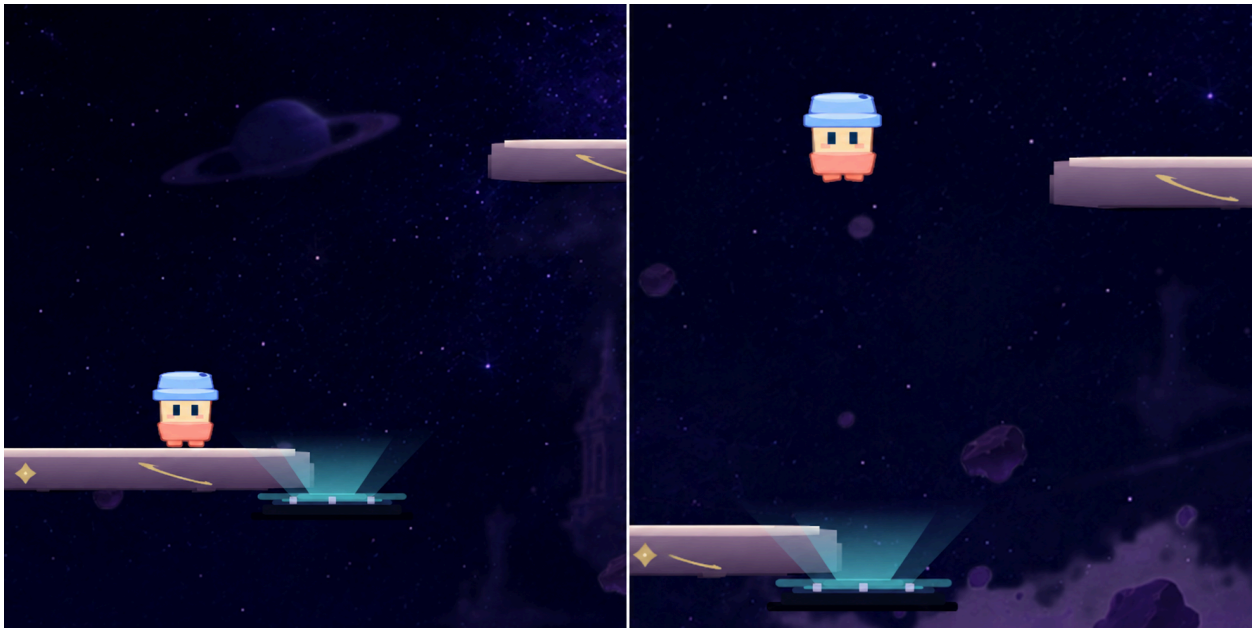


Figure 4 - Bumper

With these three mechanics, I was able to create the five levels of the section. The first three levels each introduce only one mechanic. This allows the player to understand them progressively instead of discovering everything at the same time.
The last two levels then combine these mechanics to make the section more challenging.

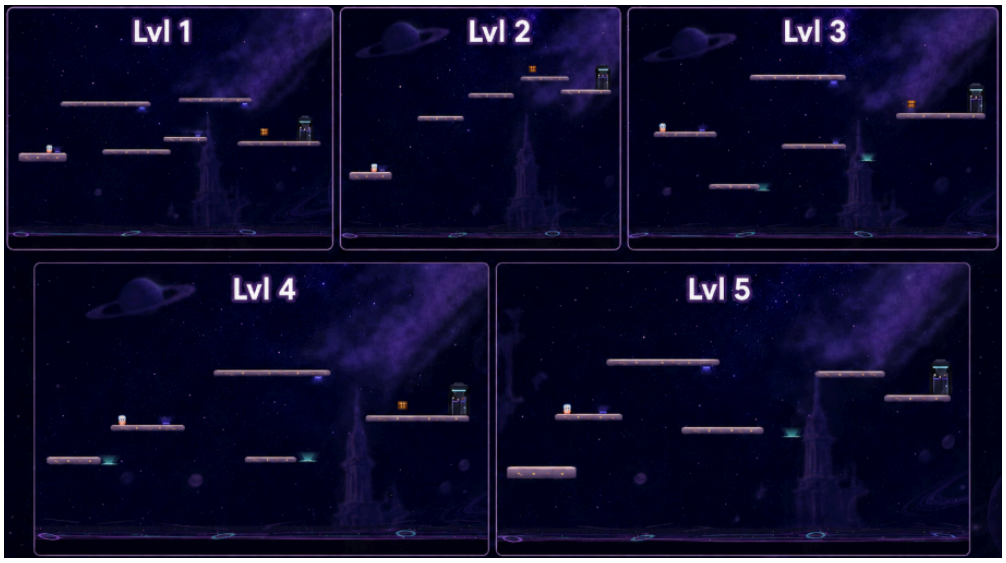


Figure 5 - The five levels of the section five

The levels of Section 5 are not among the longest levels of the game. However, they are some of the levels that require the most technical control from the players. The player must be precise, especially with low gravity, because falling can take a long time and mistakes can quickly become frustrating. For this reason, patience is also important in this section.
I also had to design the textures for the gravity button and the bumper. The button code and texture were later reused in several other levels by the other members of the group.

3. Tutorial level

As requested during the previous defense, I was also responsible for creating and adding a tutorial level. This level can be played both in solo and multiplayer modes.
The goal of this tutorial is to help new players discover the game in a simple and guided way. To do this, I added small panels that explain the basic actions and mechanics step by step.

Figure 6 - Tutorial Level

The tutorial teaches the player how to:

- move
- jump
- collect fragments and follow the progression
- understand the introduction to mechanics
- finish a level using the endpoint

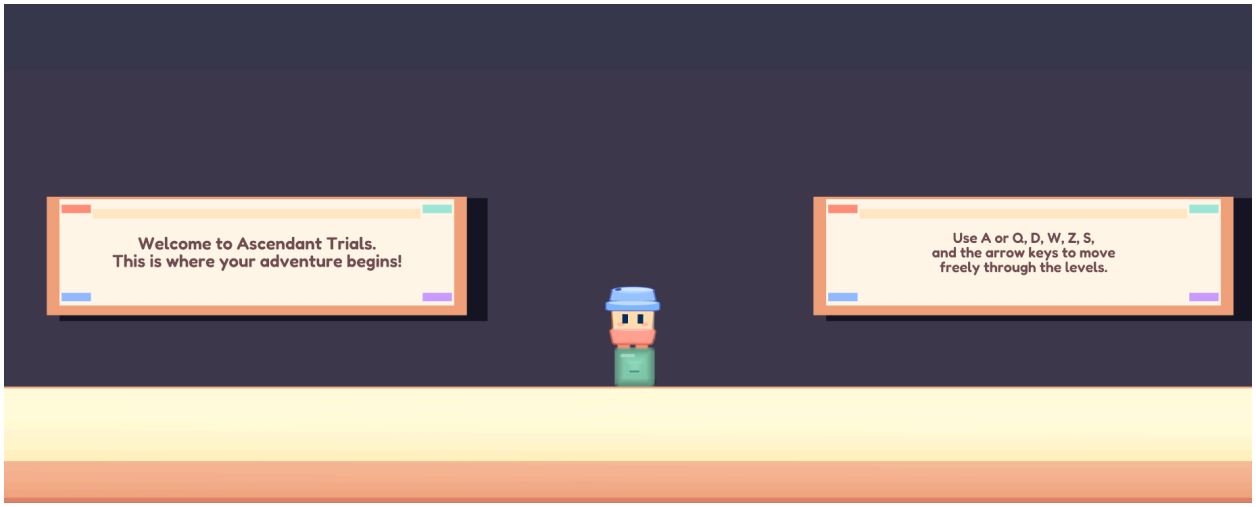


Figure 7 - The signs used to guide the player(s) through the tutorial

Regarding the tutorial menu, I also had to create and code the tutorial button. This was done using the texturing and GUI work previously made by Antonin. The button allows the player to choose between solo mode and multiplayer mode before starting the tutorial.

Figure 8 - The Gui for the Tutorial

4. Installer and Executable

I also worked on an installer for the project. The goal was to make the installation process easier by allowing the game to be installed automatically, uninstalled if needed, and launched through an already compiled executable version.

However, Antonin later developed a more complete and better adapted solution for the final version of the project. For this reason, we decided to use his version instead.

Even if my installer was not kept in the final build, this work was still useful because it allowed us to identify what was needed to make the project easier to install and execute for the final defense.

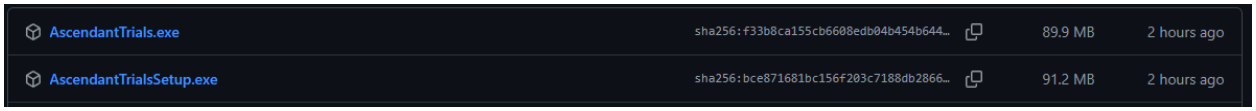


Figure 9 - Screenshot of both executable (the portable version of the game and the installer)

5. Conclusion

For this final part of the project, my role was less focused on managing the codebase than during the previous defenses. Because of my schedule, Antonin took over this responsibility, while I mainly helped by giving feedback, correcting issues, and suggesting improvements.

My main concrete contributions were the rework of several textures, the implementation of the five space levels, the creation of three new mechanics for Section 5, and the addition of a playable tutorial. These elements helped make the game more complete and refined, and easier to understand for new players.

III. TONNOT Antoine

Since the last technical report, my work has mainly focused on continuing the level design of Ascendant Trials, balancing the first section, creating section 3, and creating the music of the game. These tasks were important to improve both the gameplay progression and the atmosphere experienced by the player.

1. Improving previous levels

The first major task was the completion of section 1. As the first complete section of the game, it plays an important role in introducing the player to the main mechanics of Ascendant Trials. Because of this, particular attention was given to the difficulty of the game. This section had to remain accessible for new players while still offering enough challenge to make the gameplay interesting. Several parts of the levels were therefore tested and adjusted to make sure that the first section was not too difficult or frustrating.
During this process, some jumps, platform placements, and timing-based challenges were modified. The objective was not to make the section too easy, but to create a smoother progression for the player. Since this part of the game introduces the basic platforming experience, the player must be able to understand the mechanics progressively before facing harder challenges in later sections. This balancing work helped make the first section more coherent and better adapted to the beginning of the game.

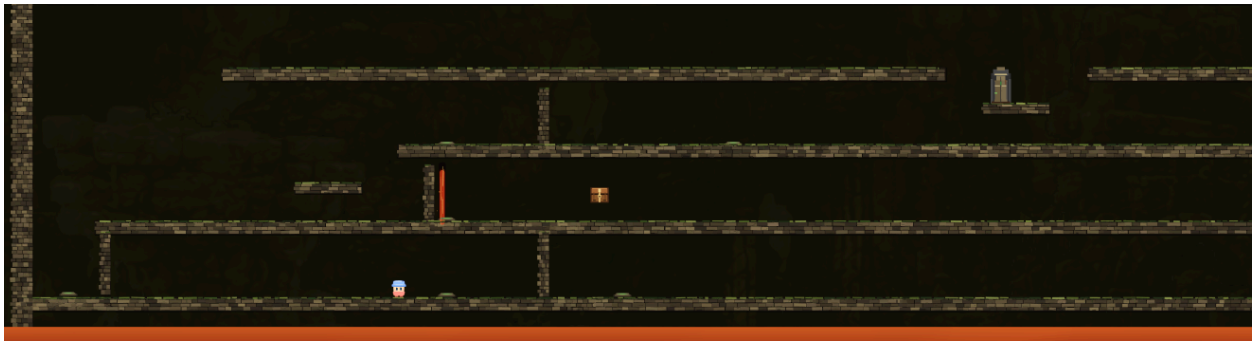
2. Introducing a brand new section

After section 1 was completed, development continued with the creation of section 3. This section was designed with a different identity from the previous levels. Its structure is more focused on maze-like environments, buttons, and environmental events. The goal was to create a section that feels distinct while still remaining coherent with the rest of the game. Instead of relying only on platforming, section 3 introduces more exploration, pathfinding, and interaction with the environment.

The section begins with an introduction level. This level was designed to establish the atmosphere of the section and prepare the player for the new type of challenges ahead. It gives the player time to observe the environment and understand that this part of the game will be more focused on navigation and puzzle-solving.

The first maze in the second level introduces the button mechanic. Buttons are used to interact with the level, for example by opening paths or activating parts of the environment. This level was designed to make the mechanic clear and easy to understand. The player should be able to see the link between pressing a button and the effect it creates in the level. This is important because the first use of a mechanic must teach the player naturally. The challenge should come from understanding the puzzle, not from confusion about how the mechanic works.

A second maze was then created with a higher level of complexity. Since the player has already been introduced to buttons, this level uses the mechanic in a more advanced way. The path is less direct, and the player must think more carefully about the layout and the order of actions required to progress. This level increases the difficulty while keeping the environment readable. The objective was to create a stronger puzzle-platforming experience without making the maze unfair or too confusing.In this level, a new environmental event was introduced: the ground starts shaking, which slows the player down. This earthquake event changes the way the player moves through the level and creates a feeling of tension. By reducing the player's movement speed, the event forces the player to adapt and react differently to the environment. It also introduces the idea that the level itself can become dangerous, not only through platforms and obstacles, but also through dynamic events that affect gameplay directly.



This event is developed further in level 4. The earthquake appears again and slows the player down in the same way, but this time it is connected to a more important moment. Right before the player reaches the door and finishes the level, a large mole appears from the ground. This moment was designed to surprise the player and introduce the mole as a major threat. The player does not fight the mole, but its appearance creates anticipation and gives meaning to the earthquakes introduced earlier.

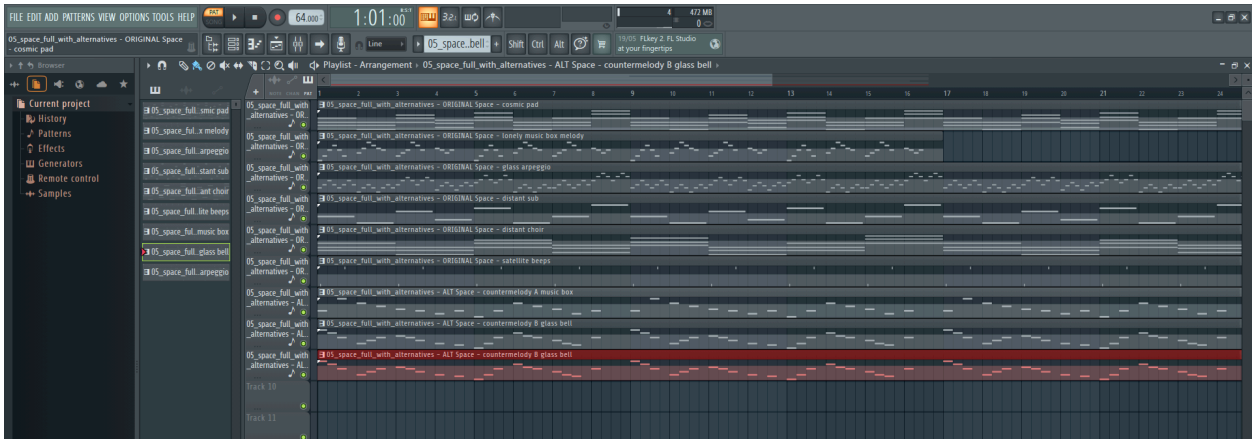
Level 5 uses the mole as the central gameplay element. In this level, the mole appears from the ground several times and tries to hit the player. Each time it appears, it causes a powerful earthquake that slows the player down. This combines the mechanics introduced in the previous levels into a more complete challenge. The player must avoid the mole's attacks while also dealing with reduced mobility caused by the earthquakes. This creates a more intense level where reaction time, positioning, and anticipation become essential.

The progression between levels 3, 4, and 5 was designed to make section 3 feel more coherent. The earthquake is first introduced as an environmental event, then connected to the appearance of the mole, and finally used as part of the mole's attack pattern. This structure allows the player to understand the mechanic step by step before facing its most difficult version. It also gives the section a stronger identity, as the mole becomes a recurring threat rather than a sudden isolated obstacle.

3. The soundtracks of the game

In addition to level design, my work also included the creation of six original music tracks: one for the menu and one for each of the five sections. Music is an important part of the player experience because it helps define the mood of each environment. Each track was therefore composed to match the atmosphere and emotions intended for its section.

The menu music was created to introduce the identity of Ascendant Trials before gameplay begins. It needed to create an appropriate first impression without being too intense. For the Hell section, the music was designed to be very energetic, matching the danger and intensity of the environment. For the Abysses section, the track focuses on tension and pressure, creating a heavier and more stressful atmosphere. The Earth section received a more joyful theme, giving it a livelier and more positive feeling. The Air section uses a more eerie atmosphere, reinforcing a sense of instability and mystery. Finally, the Space section is supported by music designed to feel wider, calmer, and more mysterious, matching the idea of a distant and unfamiliar environment.



4. Conclusion

Overall, my contribution since the last technical report has helped improve both the structure and atmosphere of the game. Section 1 was completed, balanced, and corrected to provide a smoother first experience for the player. Section 3 was created with maze-based levels, button mechanics, earthquake events, and the progressive introduction of the mole as a major threat. In parallel, six music tracks were composed to strengthen the identity of the menu and each section of the game.

IV. GLEMET Adam

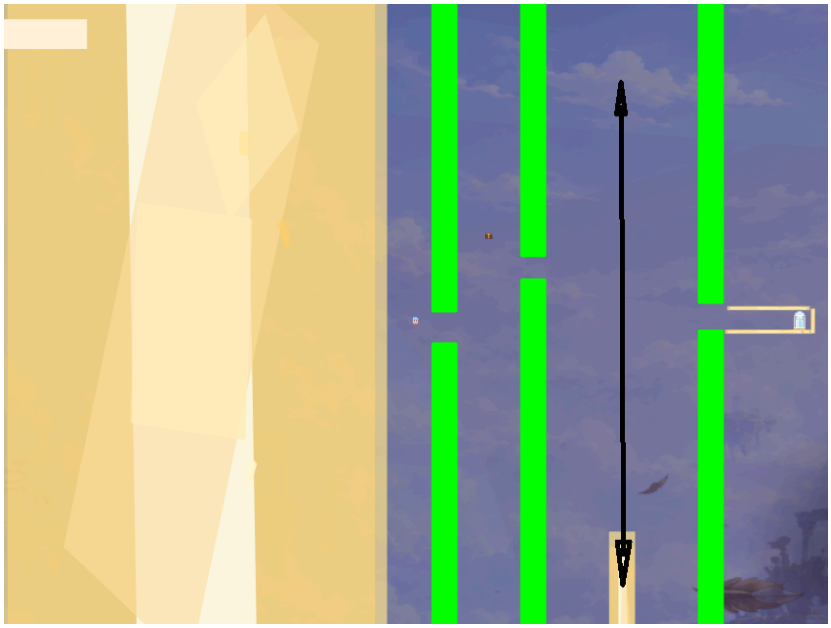
During the period between the two most recent project presentations, I had the opportunity to further develop several levels within Section 4 of the game. More specifically, I successfully designed and implemented three levels, corresponding to Levels 3 through 5, all of which incorporate a flight mechanic as a core gameplay feature.

The first of these levels is partially inspired by the game *Flappy Bird*, although it differs significantly in its execution. Instead of requiring continuous jumping inputs, the player navigates a sustained flight path. The level consists of a sequence of obstacles formed by vertical walls with openings through which the player must pass. Additionally, a sliding section has been integrated to diversify the gameplay and introduce variations in movement dynamics.

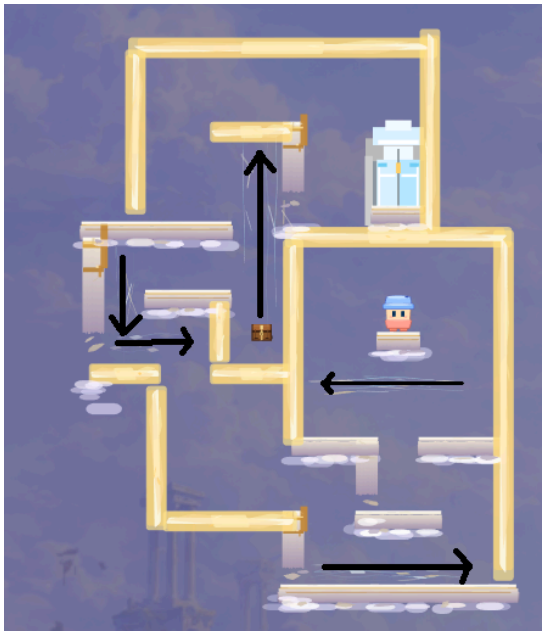
The second level is designed as a confined and intricate maze, aiming to challenge the player's precision and control while flying. This level introduces environmental forces, specifically wind currents, which actively push the player toward hazardous areas. These zones inflict damage and can result in the player's elimination. As such, this level emphasizes the player's ability to manage external forces and maintain control under constrained conditions.

Finally, the third level combines all previously introduced mechanics into a more complex and demanding environment. In addition to flight, obstacles, and wind effects, this level incorporates lightning hazards that appear at specific moments, increasing the difficulty and unpredictability. Furthermore, the level design introduces uncertainty through multiple possible paths, requiring the player to make quick decisions and adapt to evolving situations.

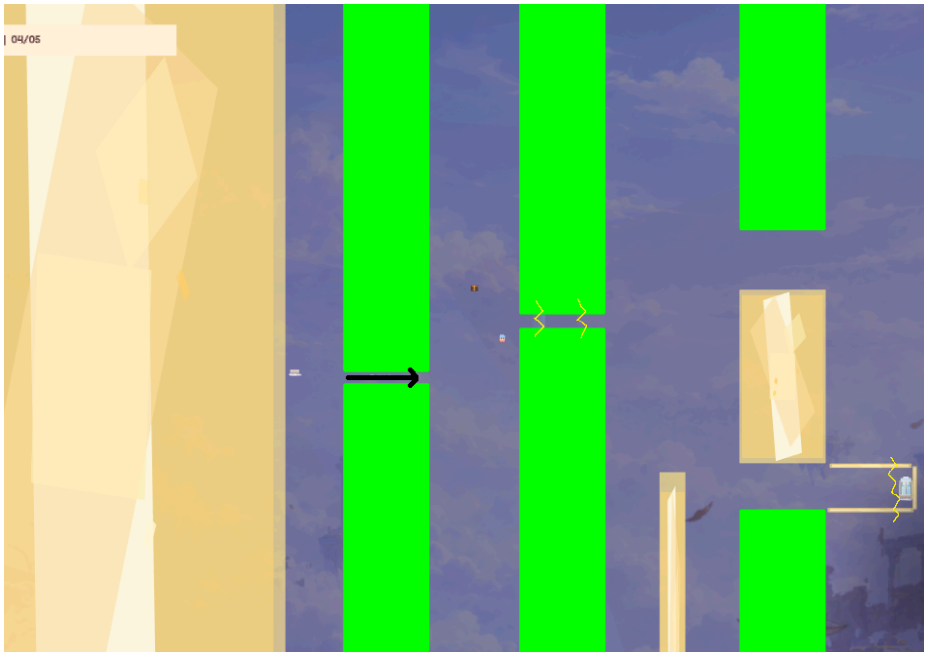
Throughout these levels the blocks that make the player die are designed to resemble sunrays and lightnings in inspiration to Icarus whose wings melted under too much sun



level 04/03



level 04/04



level 04/05 (lightning in yellow)

Final Review

I. Tests and Bug Fixing

Concerning the testing and bug fixing part, most of the tests were done together during voice calls, mainly through the multiplayer mode. This allowed us to test the game in conditions closer to a real play session, especially because Ascendant Trials is based on cooperation between players. By playing together, we were able to quickly notice problems related to synchronization, collisions, level progression, player movement, and interactions between different mechanics.

We also asked some friends and classmates to test the game. This was useful because they were not as familiar with the project as we were, so they could give us a more neutral point of view. They sometimes found bugs or confusing elements that we had not noticed ourselves, simply because we were already used to the game and its logic.

For bug fixing, we created a dedicated forum on the project Discord server. This forum was used to report every bug found during tests. Each report usually described the problem, when it happened, and sometimes which level or mechanic was concerned. This made it easier for the members responsible for the affected parts of the project to identify and correct the errors.

Overall, this organization helped us centralize bug reports and avoid losing information between voice calls or messages. It also allowed us to progressively improve the stability of the game and make the final version cleaner and more playable.

II. Difficulties Encountered

Concerning the difficulties we encountered during the development of the project, one of the most important ones was, as mentioned previously, the departure of Marine MAILLOT from the group. Her departure forced us to reorganize part of the project, especially because she was responsible for several important tasks such as NPC AI, character design, animation, and part of the mythological identity of the game. Because of this, we had to rethink some features and simplify a few parts of the original project to make sure that the final version would still be achievable.

Another difficulty we faced was related to the use of Git. Since not every member of the group had the same experience with version control, some mistakes happened during development. For example, some progress was pushed directly onto the main branch, sometimes overwriting work that was already functional. This created problems in the codebase and forced us to do some rollbacks and manual corrections, which were not always easy to manage. These issues made us realize the importance of using branches correctly and being more careful before merging or pushing important changes.

We also encountered difficulties with the networking system. Our multiplayer system works through an intermediate server, which is useful to manage rooms and communication between players. However, this architecture also creates some delay, which can become noticeable during gameplay. Since Ascendant Trials is a platformer based on timing, movement, and cooperation, even a small delay can affect the player experience. Because of this, we had to adapt some parts of the game and test the multiplayer mode many times to reduce problems as much as possible.

Finally, one of the most constant difficulties during this last defense period was the harsh and dry climate of the northern Île-de-France region, in other words, Paris and its suburbs. Even if this may sound less technical than the other issues, the temperature did affect the team during work sessions. It also affected the equipment and technical infrastructure used to run and test the game. Long development sessions became more tiring, and some computers had more difficulty running the project comfortably during hotter periods.

Overall, these difficulties forced us to adapt our organization and our ambitions. Some features had to be simplified, some technical choices had to be adjusted, and the team had to work more carefully to keep the project stable. Even if these problems slowed us down at different moments, they also helped us understand better how important organization, communication, and testing are in a group project.

Conclusion

To conclude, this year-long project brought us a lot, both technically and personally. Through Ascendant Trials, we discovered more deeply the world of video game development and all the work required to create a playable and coherent game.
We also learned a lot about networking, especially in the context of multiplayer games. Managing rooms, player synchronization, delays, and shared progression showed us how complex online gameplay can be.
This project also helped us improve our planning, communication, and teamwork. We had to divide tasks, adapt to problems, and reorganize ourselves when needed.
In addition, we gained more experience with tools such as Git and GitHub. Even if some mistakes were made, they helped us understand the importance of good version control practices.
Finally, this project showed us that some coding habits we had before were not adapted to bigger projects. A larger codebase requires cleaner organization, clearer responsibilities, and more maintainable code.
Overall, Ascendant Trials was a challenging but very useful experience.