Project Report

Project name: Timeless Empire

Project team: **Quentin Redt-Zimmer** – Time manager

Yiru Xiong – Project manager

Rory Bueno - Technicians

Mathis Constant - Technicians

Luka Lesueur – Communications

Website: *https://timeless-empire.fr/*

Version: V1 - 25/05/2026

Table of contents

Incorporation of the specifications Page 5

1 - General Presentation of the Project Page 5

2 - Project Objectives Page 5

3 - Main Expected Features Page 6

4 - Technical Constraints Page 7

5 - Target Audience Page 7

6 - Project Validation Criteria Page 7

Conclusion Page 8

I - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - LUKA LESUEUR

1 - Web infrastructure, visual identity, and communication strategy Page 9

1.1 - Artistic direction compliance Page 9

1.2 - Professionalization deployment Page 10

1.3 - Communication flow management Page 11

2 - Wiki implementation and game engine architecture Page 12

2.1 - The web wiki system Page 12

2.2 - The turn and card engine Page 13

2.3 - Object-oriented programming (OOP) and business logging Page 13

3 - Major achievement : autonomous decision-making Artificial Intelligence Page 14

3.1 - Baseline prototype and score-based evaluation Page 14

3.2 - Code organization Page 15

4 - Reflective analysis : joys, sorrows, and technical insights Page 16

4.1 - The sorrows Page 16

4.2 - The joys Page 17

Conclusion Page 18

II - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - YIRU XIONG

1 - Visual identity of the Different Eras Page 19

2 - Character design Page 22

3 - Card design and Gameplay Assets Page 23

4 - User interface and Menu design Page 25
5 - Team leadership and Project organization Page 26
Conclusion Page 29

III - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - QUENTIN REDT--ZIMMER

1 - Reorganization of the project architecture Page 30
2 - Development of the Multiplayer Mode Page 31
3 - Transition to a Public Online Multiplayer System
Page 31
4 - Visual Rework of the Map and Graphic Elements Page 32
5 - Improvement of the User Interface Page 33
Conclusion Page 34

IV - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - RORY BUENO

1 - Objectives of the multiplayer system Page 36
2 - Client-Server Architecture Page 37
3 - Action Synchronization System Page 38
4 - Server management and player connections Page 39
5 - Integration with Gameplay System Page 40
6 - Debugging, Testing, and Optimization Page 41
7 - Skills Developed During the Project Page 42
Conclusion Page 43

V - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - MATHIS CONSTANT

1. Improvement of the Solo Mode Page 44
2. Dynamic Game Initialization and Technical Adaptation Page 46
3. Reorganization of the Project Architecture Page 47
4. Preparation for Executable Packaging and Deployment Page 49
5. Conclusion and Skills Developed During the Project Page 50

VI - INSTALLATION MANUAL Page 52

VII - USER MANUAL Page 53

1 - Introduction and game philosophy Page 53

2 - User interface and general navigation Page 53

3 - The multiplayer lobby system Page 54

4 - Core game and session flow Page 54

5 - Detailed functional mechanics Page 55

5.1 - Resource dependency system Page 55

5.2 - Historical eras Page 55

5.3 - Automated combat resolution Page 56

6 - Victory conditions Page 56

7 - Building Cost/Reward Page 57

Review of the Functional Specifications

1 - General Presentation of the Project

As part of our group project, we developed *Timeless Empire*, a multiplayer strategy video game entirely programmed in Python using the Pygame library.
The main objective of the project was to create a strategic and entertaining multiplayer experience while respecting the technical constraints established at the beginning of the development process.

The game is based on an empire management system in which several players compete through various mechanics such as resource management, technological development, diplomacy, trading, and combat.

2 - Project Objectives

The main objectives defined in the functional specification document were the following:

- develop a complete game entirely coded in Python;
- create a strategic and competitive multiplayer experience;
- design a simple and accessible user interface;
- allow players to manage their empire and resources efficiently;
- implement a progression system through different historical ages;
- ensure stability and synchronization during multiplayer sessions.

The project also aimed to encourage players to develop strategic thinking skills while providing an enjoyable gaming experience.

3 - Main Expected Features

Several essential functionalities were defined in the original specifications.

- Turn-Based System

The game is based on a turn-based system. Each player has a limited amount of time to perform actions such as building structures, attacking opponents, or trading resources.

- Resource Management System

Players must manage different resources such as wood, food, and gold in order to expand their empire and construct buildings.

Trading and Diplomacy System

Players can create alliances, trade resources, or declare war on one another, encouraging social interaction and strategic decision-making.

- Skill Tree System

A technological progression system allows players to unlock new buildings, units, and upgrades as they advance through the game.

- Combat System

Battles between players are automatic and depend on troop strength as well as a random factor in certain situations.

4 - Technical Constraints

The project had to respect several important technical constraints:

- exclusive use of the Python programming language;
- use of the Pygame library without any external game engine;
- compatibility with online multiplayer gameplay;
- responsive and user-friendly interface;
- stable performance during long game sessions.

The game also needed to remain accessible on standard desktop computers without requiring advanced hardware specifications.

5 - Target Audience

The game is primarily intended for players aged sixteen and above who enjoy strategy and management games.
The objective was to provide an experience that is both accessible and sufficiently complex to stimulate strategic thinking and player interaction.

6 - Project Validation Criteria

The project would be considered successful if the following conditions were met:

- a complete game session with three to five players could be played;
- the main gameplay features were fully operational;
- the game remained stable without critical errors;
- new players could easily understand the interface and rules;
- only minor bugs remained without affecting gameplay.

Several testing phases were planned throughout development in order to identify issues, improve stability, and optimize the user experience.

Conclusion

The functional specification document served as the foundation for the development of *Timeless Empire*.

It allowed the team to clearly define the project objectives, organize task distribution among team members, prioritize important features, and guide the development process until the creation of a playable prototype.

I - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - LUKA

1 - WEB INFRASTRUCTURE, VISUAL IDENTITY, AND COMMUNICATION STRATEGY

The launch of the Timeless Empire project required the immediate establishment of a technological showcase and a credible communication channel for our fictional studio. My role within the group was twofold: I was responsible for the creation and design of the project's website, as well as the overall management of our communication strategy.

Drawing on the theoretical and practical skills acquired during my NSI (Digital and Computer Sciences) specialization in my final year of high school, I developed the site's architecture using HTML and CSS. Although I did not consider myself a professional developer in these languages, I had already put these skills into practice by creating personal websites for the volunteer association I work with. However, the website for Timeless Empire placed me in a more demanding context, requiring a higher level of polish and reliability.

My work focused on three major axes during this first phase:

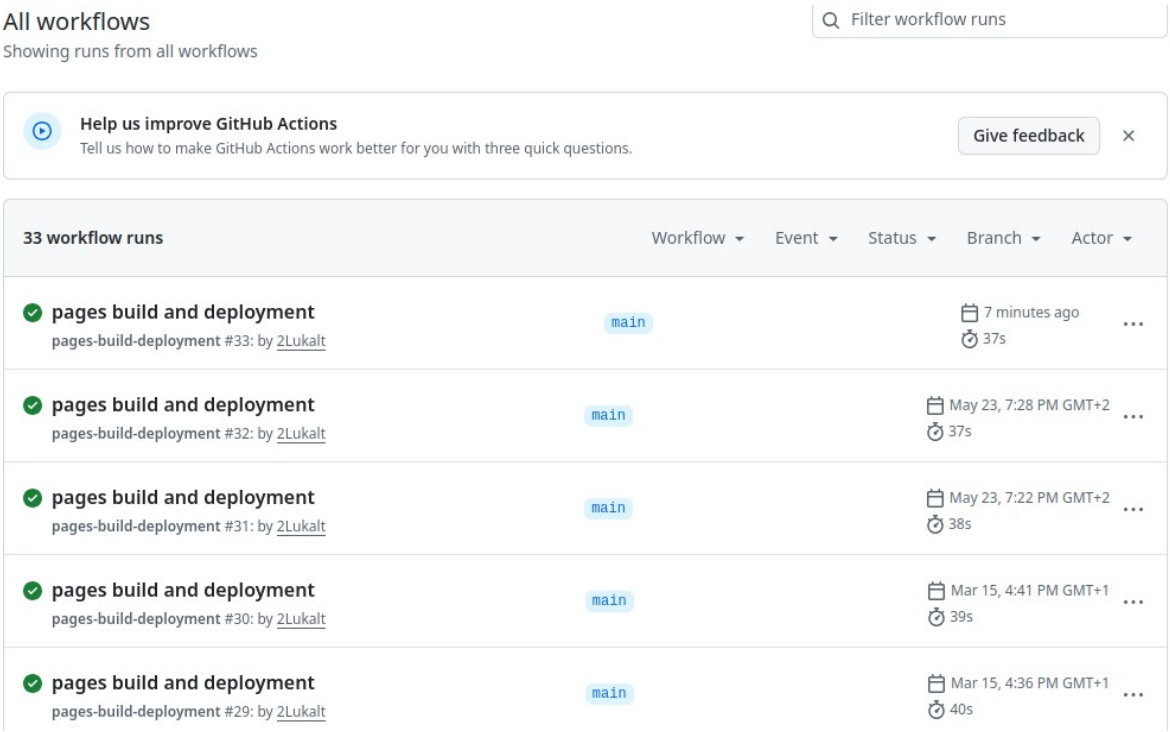
1.1 - ARTISTIC DIRECTION COMPLIANCE

My main technical task was to ensure that the site aligned perfectly with the project's artistic direction. It was not merely about displaying information; it involved a careful selection of color palettes, button shapes, and fonts to ensure the website felt personal and unique. The goal was to avoid the "generic" look often found in standard templates, ensuring that the platform reflected the specific atmosphere of our game. To achieve this, I relied heavily on online documentation for advanced CSS, while also revisiting code structures from my high school projects to establish a reliable base.

1.2 - PROFESSIONALIZATION AND DEPLOYMENT

Following a strategic discussion with the other members during a coordination call on Discord, we decided to take a significant step forward to professionalize our web presence. I took the initiative to purchase the domain name "timeless-empire.fr" through the Swiss registrar Infomaniak. This ensures that our site is accessible to the general public at any time, adding a layer of credibility to our work. For hosting, I chose GitHub Pages. This choice was strategic: beyond simply hosting the site, GitHub allows all team members to consult or modify the source code whenever they wish, facilitating collaborative work.

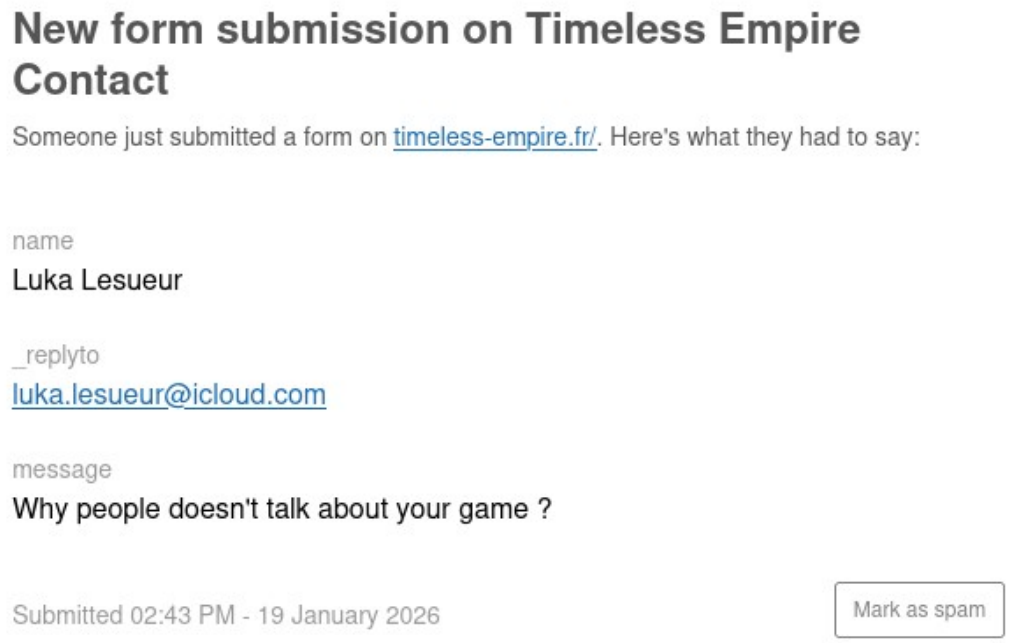
GitHub :



1.3 - COMMUNICATION FLOW MANAGEMENT

As seen on the "Contact" page, the domain name allowed us to create a professional email address: **contact@timeless-empire.fr**. This gives players a direct channel to report bugs or share their feedback. Currently, this interaction is managed through a form connected to the Formspree platform. In parallel, I took charge of communication to keep the site dynamic (patch notes, announcements).

Formspree :



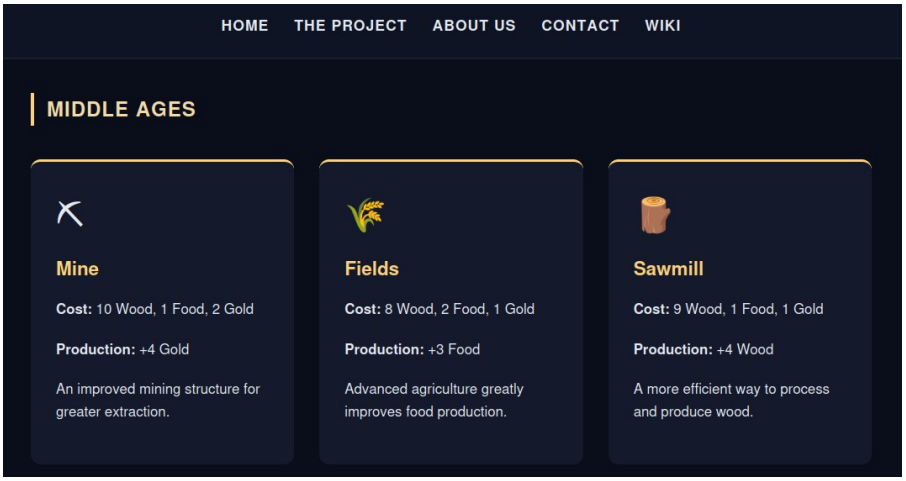
2 - WIKI IMPLEMENTATION AND GAME ENGINE SYSTEM ARCHITECTURE

The second cycle of the project marked a more technical turn, requiring me to shift from pure web communication to algorithm programming and data modeling in Python. My work was then divided between creating the Wiki page of the website and developing the turn and card system in the file tours.py.

2.1 - THE WEB WIKI SYSTEM

I added a “Wiki” page to the site to clearly present all the buildings available in Timeless Empire and explain their costs, production, and role through the different eras of the game. This page helps players better understand the progression system and makes the project feel more complete and immersive.To build this page, I reused the existing website style (header, navigation bar, hero section, and footer) so that it would integrate naturally with the rest of the project. The content is organized by historical eras: Prehistoric Age, Middle Ages, Modern Times, and Contemporary Era. Each building is displayed using the CSS class “.feature-card”, which ensures a clean and consistent layout across the page.I also added icons and descriptive text for every building to make the information easier to read and more visually engaging. Finally, hover effects were included (the card slightly lifts, the shadow changes, and the icon grows larger) to give the interface a more dynamic and modern appearance.

Wiki Page :



2.2 - THE TURN AND CARD ENGINE

In parallel, I developed the turn system in the file tours.py. This part forms one of the foundations of the gameplay. The goal was to ensure that at each turn, the player is confronted with a choice between three randomly generated cards, with the chosen effect applying immediately. It is important to note that while this card system was fully prototyped, we chose to remove it from the final build to optimize gameplay fluidity and respect our deadlines. This pivot forced us to re-evaluate our initial objectives, proving our ability to adapt our technical choices to the reality of the production constraints.

2.3 - OBJECT-ORIENTED PROGRAMMING (OOP) AND BUSINESS LOGING

In the code, this logic relies on the Player class, which represents a player and contains their information (name, resources in wood, food, gold, and owned buildings). This class manages actions through methods to modify its state (receiving resources, building, or destroying a building during a malus). The execution relies on global constants (number of turns per period, structure costs, base income). I coded the application of card effects . I also added a trapped building system that triggers a delayed destruction. Finally, the TurnManager class handles the turn order of players, turn progression, and automatic resource distribution at the end of a round, connecting all the mechanics together.

3 - MAJOR ACHIEVEMENT : AUTONOMOUS DECISION-MAKING ARTIFICIAL INTELLIGENCE

For this third part, I focused on creating a baseline prototype for the game's Artificial Intelligence, implemented through the "DecisionBotTimelessEmpire" class. The goal was to establish a functional foundation capable of evaluating tour choices by calculating and comparing scores.

3.1 - BASELINE PROTOTYPE AND SCORE-BASED EVALUATION

To avoid random behavior, I designed an evaluation function named “evaluer_choix_tour" that analyzes the current state of the automated player (wood, food, and or) and compares exactly two options: a resource card (carte_ressource) and a building card (carte_batiment).
The decision-making process executes the following exact scoring rules:

- **RESOURCE EVALUATION CRITERIA**: The bot checks if its wood or food supplies are strictly below 10. If this condition is met, it considers it an emergency and adds +5.0 points to the resource score (score_ressource). Otherwise, if resources are at 10 or above, the resource option only receives a baseline score of +1.0 point.

- **BUILDING EVALUATION CRITERIA**: In parallel, the script calculates a building score (score_batiment) based on a fixed cost threshold. It checks if the player's wood is greater than or equal to 15 AND if its gold (or) is greater than or equal to 10. If the bot has enough supplies to afford it, the building option gains +6.0 points. If the player cannot afford it, the building option is penalized by -2.0 points.

- **ARBITRATION AND SELECTION**: Finally, the script compares the two resulting scores. If score_ressource is greater than score_batiment, it returns the resource card. If score_batiment is greater, it returns the building card. In the exact event of a tie where both scores are equal, the bot falls back on a random selection using the random.choice function to choose between the two cards.

3.2 - CODE ORGANIZATION

Placing the AI in its own script allows my teammates (Quentin, Yiru, Rory, and Mathis) to open the file, directly understand how the AI thinks, and easily modify or adjust choices if necessary.

4 - REFLECTIVE ANALYSIS : JOYS, SORROWS, AND TECHNICAL INSIGHTS

These eight months of development on Timeless Empire have been, by far, the most intense, chaotic, but also formative phase of my education. Moving from the status of a high school student to that of an engineering student through a project of this scale brings as many moments of doubt as it does great victories.

4.1 - THE SORROWS

The first major obstacle I had to face was the "blank page syndrome." At the beginning of the year, the sheer scale of the project and the total freedom regarding the topic sometimes left me feeling helpless, not knowing where to begin with the website architecture or the game engine.

Furthermore, the visual and technical creation process was not a straight line. I spent exhausting hours on precise details, often asking myself during moments of fatigue: "Is it really worth giving myself this much trouble for a single button or a specific CSS animation?" Achieving a professional and smooth result requires behind-the-scenes work that is not necessarily obvious at first glance.

The other major difficulty was human and collaborative. Investing so much personal effort into a design or a feature only to receive negative remarks or criticism from other team members was sometimes hard to swallow. It is difficult not to take criticism to heart when you have spent your whole night on a piece of code.

Finally, the transition to Python and the integration of everyone's features (such as the network multiplayer) generated complex synchronization bugs and aberrant behaviors that tested my nerves and patience during console debugging sessions.

4.2 - THE JOYS

Fortunately, these difficulties made our collective and individual successes much more rewarding. My greatest joys occurred when abstract ideas finally took shape on the screen.

I remember several milestone moments:

- **THE CONCRETIZATION OF THE SITE:** Seeing the code behave exactly as intended, with the Wiki visual effects fully functional and the graphic charter respected.

- **GOING INTO PRODUCTION**: The moment the website became accessible to the entire world under our own domain name, timeless-empire.fr. This was concrete proof that our team was moving from a simple student project to a finished, professional product.

- **GROUP COMMUNICATION**: Successfully transforming friction and negative feedback into collective solutions during our Discord calls. I realized that my teammates' criticisms were not meant to discourage me, but served as the fuel needed to push the website and the game toward a result bordering on perfection.

On a technical level, this project allowed me to overcome my blocks. I gained a concrete and robust understanding of advanced concepts in object-oriented programming, code modularity (notably by isolating our scripts to work cleanly as a group), and web administration. I come away from this project with true versatility and the confidence to lead a computer science project from A to Z.

CONCLUSION

In conclusion, my involvement throughout the Timeless Empire project allowed me to develop a versatility that is rare for a first-year engineering student. By taking charge of both the external technological showcase (the website, the Wiki, the domain name) and the most complex internal algorithmic gears (the turn system and the decision-making artificial intelligence isolated in its own module), I was able to understand a software project as a whole.

This human and technical experience within our team concretely validates the requirements of rigor, autonomy, and innovation of our computer science curriculum.

II - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - YIRU XIONG

My main responsibility within the project was the artistic direction and visual design of the game. I worked on the creation of the graphical assets, including buildings, characters, cards, and user interface elements. In addition to the artistic work, I also acted as the team leader, which involved organizing the team, coordinating tasks, and managing the overall progress of the project.

This report presents the work I completed during the project, the artistic choices that were made, and the improvements implemented during the final phase of development.

1 - Visual Identity of the Different Eras

One of the most important aspects of the project was designing the visual identity for each historical era represented in the game. Since the gameplay evolves over time, the visual environment also needs to evolve in order to reinforce the player's immersion and clearly communicate the progression between periods.

My task was to create buildings corresponding to each era while maintaining a coherent global art style. Each structure had to be visually recognizable and historically inspired, while still remaining readable within the game's pixel-art environment.

For example, early-era buildings were designed with simpler shapes and natural materials such as wood and stone, while later eras introduced more advanced architectural elements and more detailed structures. The challenge was to create visual variety without making the transition between eras feel inconsistent.

Particular attention was also given to the readability of the buildings on the map. Since this is a strategy game, players need to quickly identify structures and understand their function during gameplay. To achieve this, I used clear silhouettes, consistent proportions, and distinct visual details for each type of building.

During the final phase of development, I also readjusted the pixel sizes and proportions of the buildings and their bases. Some assets initially appeared too large or too small compared to the map tiles, which affected gameplay readability and unit placement.

I therefore resized and repositioned several elements to improve the overall functionality of the map and create a more balanced visual composition.


Contemporary-church -1.0


Contemporary-church -2.0


Contemporary-house-1-1.0


Contemporary-house-1-2.0


Contemporary-house-2-1.0


Contemporary-house-2-2.0


Contemporary-house-3-1.0


Contemporary-house-3-2.0


Prehistory-house -1-1.0


Prehistory-house -1-2.0

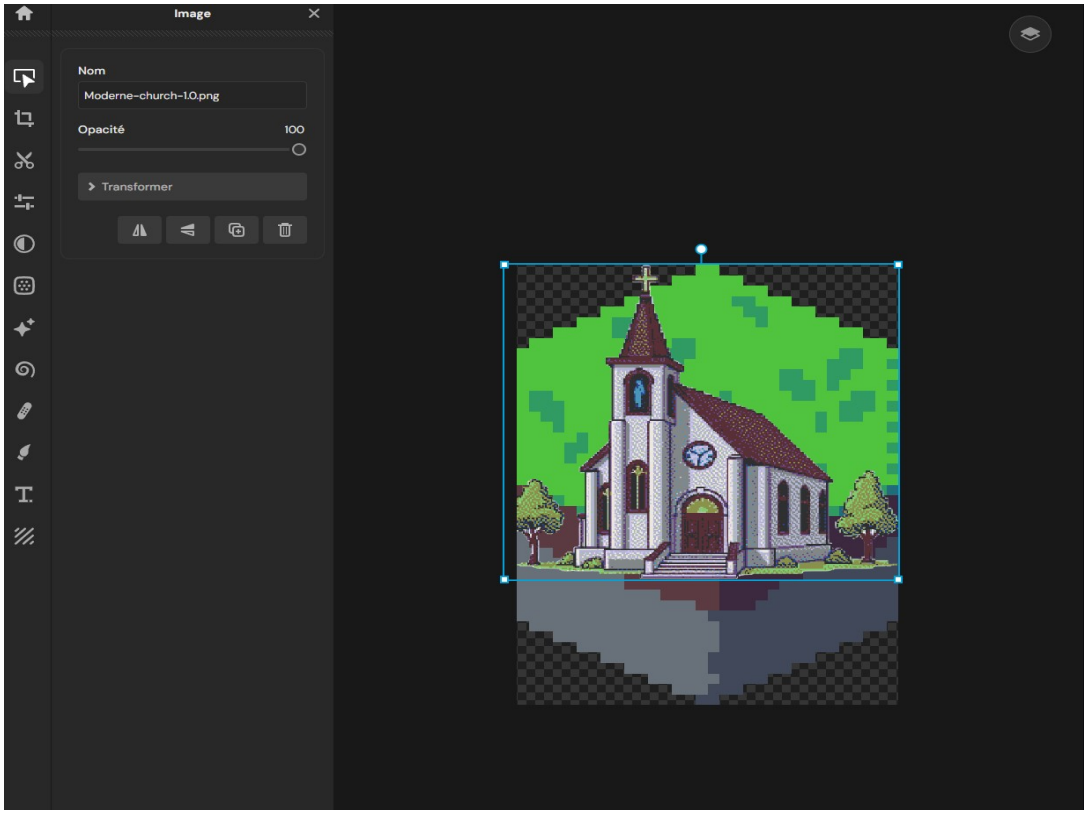

Prehistory-house -2-1.0

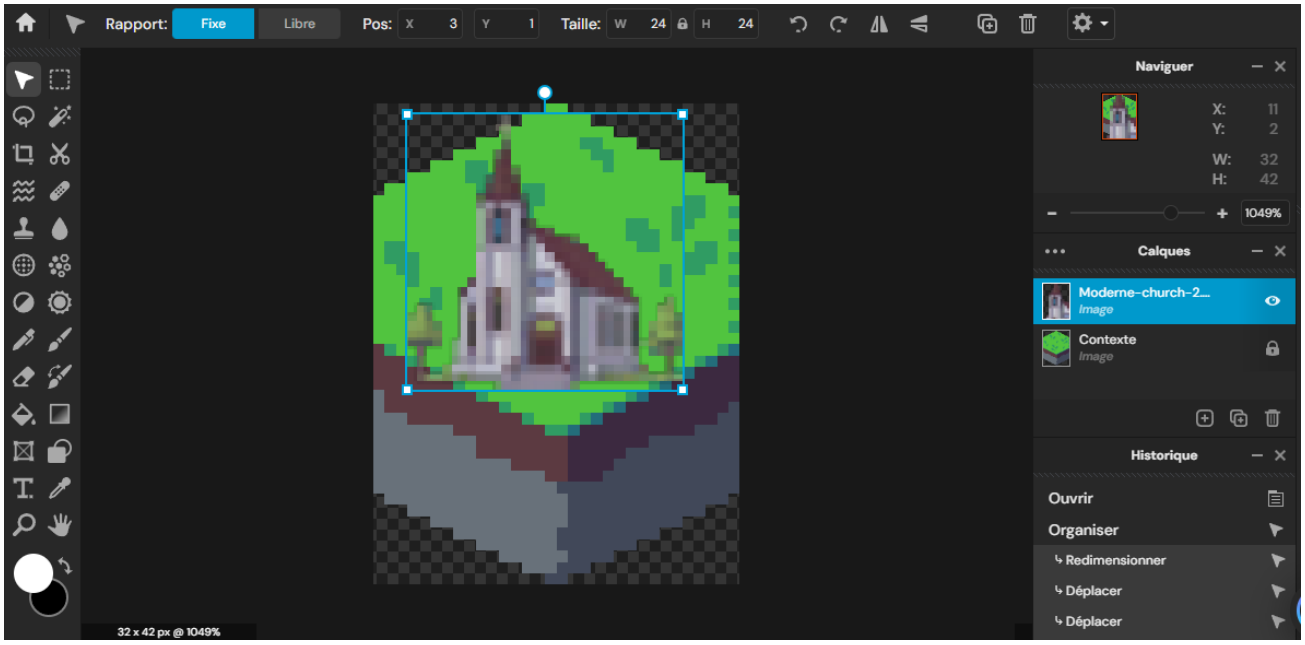

Prehistory-house -2-2.0

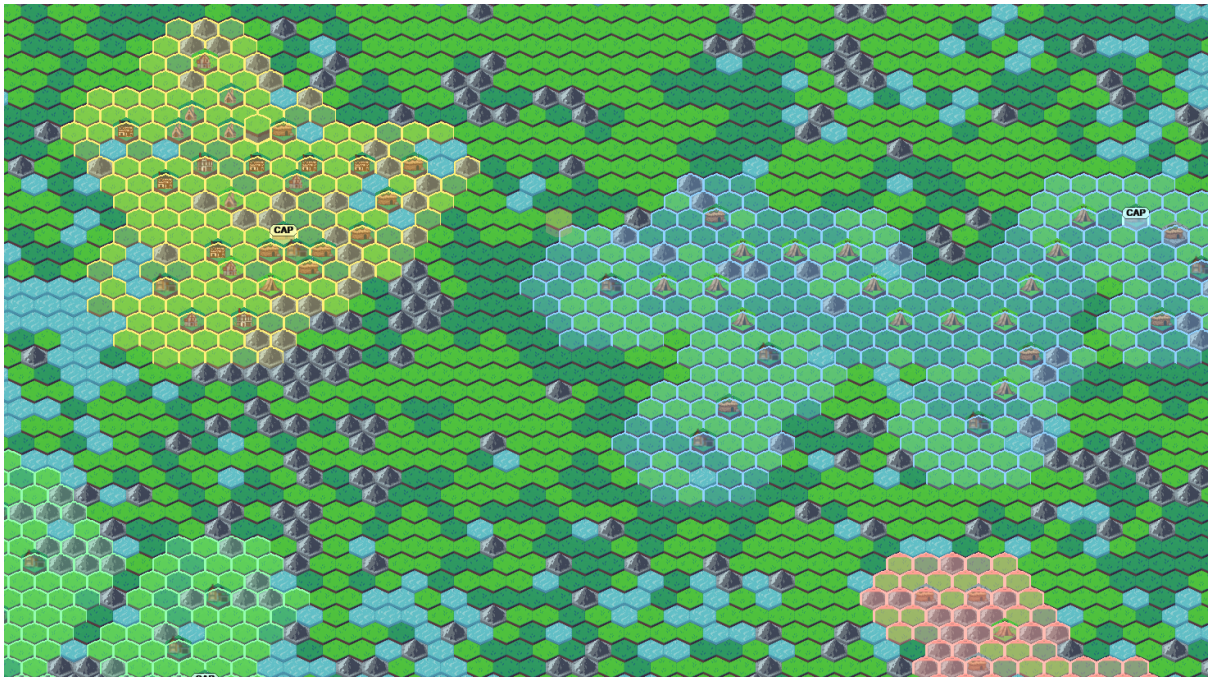

Prehistory-house -3-1.0


Prehistory-house -3-2.0

Example of pixel modification:







2 - Character Design

Another important part of my work was the creation of the game characters. The objective was to design characters that would remain readable at a small scale while still expressing enough personality to be visually distinctive.

Each character was designed according to its gameplay role. I focused on recognizable shapes, color variations, and visual accessories to help players quickly identify units during the game. Since strategy games often require fast decision-making, readability was one of the main priorities throughout the design process.

The characters also needed to fit naturally within the overall artistic direction of the game. To maintain visual consistency, I used the same pixel-art style, color palette, and level of detail as the environment assets. During development, several iterations were necessary to improve the clarity and proportions of the characters. Some early concepts appeared too detailed for the game scale, so I simplified certain elements to improve visibility and animation readability.

In the later stages of the project, I explored the creation of additional character designs in order to expand the game content and introduce more gameplay variety. However, integrating these new characters into the game proved to be more technically and artistically complex than expected within the project timeframe. Although these designs were completed as part of the creative process, they were ultimately not implemented in the final version of the game.

Main characters:

3 - Card Design and Gameplay Assets

Cards are an important gameplay element in our project. They provide information about resources, characters, and various gameplay mechanics. Because players interact with these cards frequently, their visual presentation needed to be both attractive and functional.

I worked on the design of multiple types of cards, including character cards and resource cards such as gold, metal, and other materials used during the game. My role included designing the layouts, creating illustrations, and organizing the visual hierarchy of information.

One of the main challenges was balancing aesthetics and readability. The cards needed to contain enough information for gameplay while remaining visually clear and easy to understand. To achieve this, I created structured layouts with identifiable icons, readable text placement, and consistent color coding depending on the type of resource or character.

The visual style of the cards was also designed to match the rest of the game. I reused similar colors, textures, and pixel-art techniques to maintain artistic coherence between the cards and the game environment.

However, same as the characters, during development, we realized that integrating all of these card designs into the game was more complicated than expected from a technical and

gameplay perspective. Because of this, some of the designs were ultimately not implemented or used directly in the final version of the game. Even so, the work helped define the visual direction and provided valuable experimentation for the project's artistic identity.

Some cards are still being improved and finalized, particularly regarding balancing the amount of visual information displayed. Nevertheless, the current designs already establish the artistic direction intended for the completed version of the game.

Messengers:

Soldiers:

Vehicles:

Car-1

Tank-1

Tank-2

Resource cards:

Gold-1.0

Gold-2.0

Meat-1.0

Meat-2.0

Metal-1.0

Metal-2.0

Money-1.0

Money-2.0

Wood-1.0

Wood-2.0

4 - User Interface and Menu Design

The user interface plays an important role in the player experience. A good interface allows players to navigate the game easily and access information quickly without breaking immersion.

My work on the interface mainly focused on the menu design and visual backgrounds. I created the menu screens with the objective of making them intuitive, clean, and visually consistent with the rest of the project.

The menu backgrounds were designed to reflect the atmosphere of the game while avoiding excessive visual clutter. I used simple compositions and coherent colors so that buttons and navigation elements remained clearly visible.

I also contributed to the placement and visual organization of interface elements to improve usability. Since strategy games often contain many systems and resources, clarity was essential to prevent the interface from becoming confusing for players.

Another important aspect was ensuring visual consistency between gameplay screens and menu pages. The same artistic style and color palette were used throughout the interface to create a unified player experience.

Background-1

Background-2 (used for menu)

5 - Visual Transitions Between Eras

I also created transition videos between different eras of the game. These videos were designed to visually represent the passage from one period to another, helping to clearly communicate the shift in time, atmosphere, and context. They support the narrative structure by making each era change feel more intentional and visually impactful.

By adding motion and visual continuity between distinct phases of the game, these transitions improve immersion and help the player better understand the progression of the story and the evolution of the game world.

contemporary-modern

middle-contemporary

modern-futur

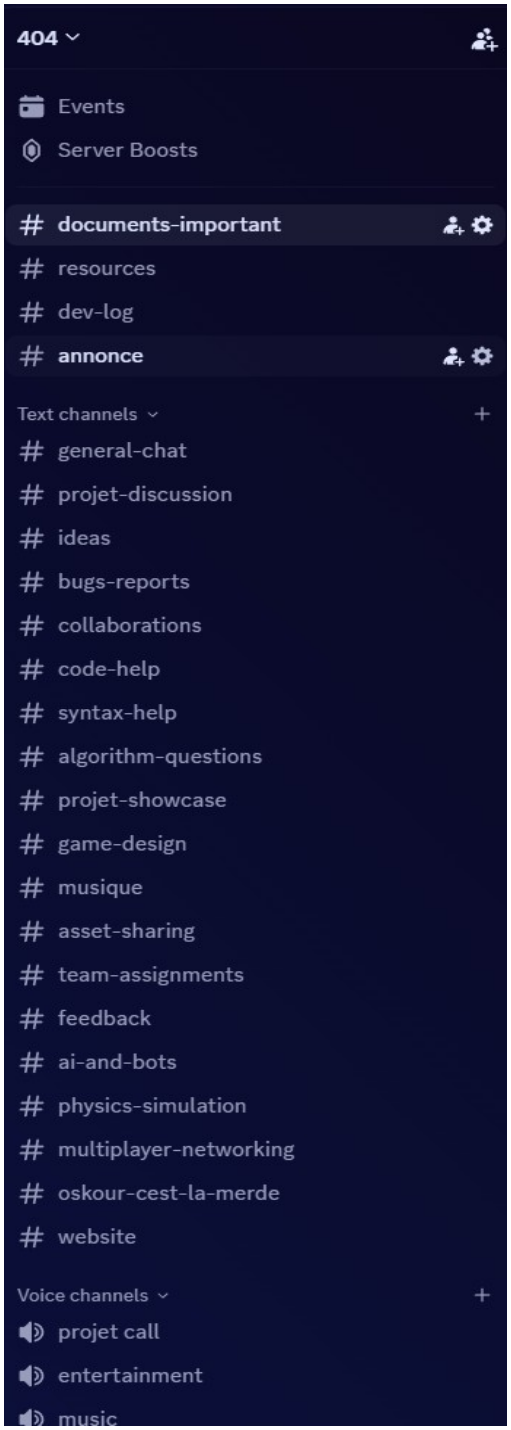
prehistoire - middle

6 - Team Leadership and Project Organization

In addition to my artistic responsibilities, I also worked as the team leader for this project. My role was to organize the team’s workflow, coordinate tasks between members, and ensure that the development progressed efficiently and on schedule. I distributed objectives, monitored the progress of different parts of the project, and maintained clear communication within the team to ensure good collaboration between all members.

I also participated in important artistic and organizational decisions throughout the development process. When technical or production issues appeared, I helped the team find solutions and adjust our planning when necessary. Managing both the creative and

organizational aspects of the project helped me strengthen my communication, teamwork, and project management skills.



CONCLUSION

This project allowed me to develop both my artistic and organizational skills through the creation of a complete strategy board game. My work focused on building a coherent visual identity across multiple historical eras while ensuring that all graphical elements remained functional and readable for gameplay.

Throughout the project, I designed buildings, characters, cards, and user interface elements while continuously improving and adjusting assets according to the needs of the game. The final development phase also included resizing and optimizing several graphical elements to improve map functionality and overall visual balance.

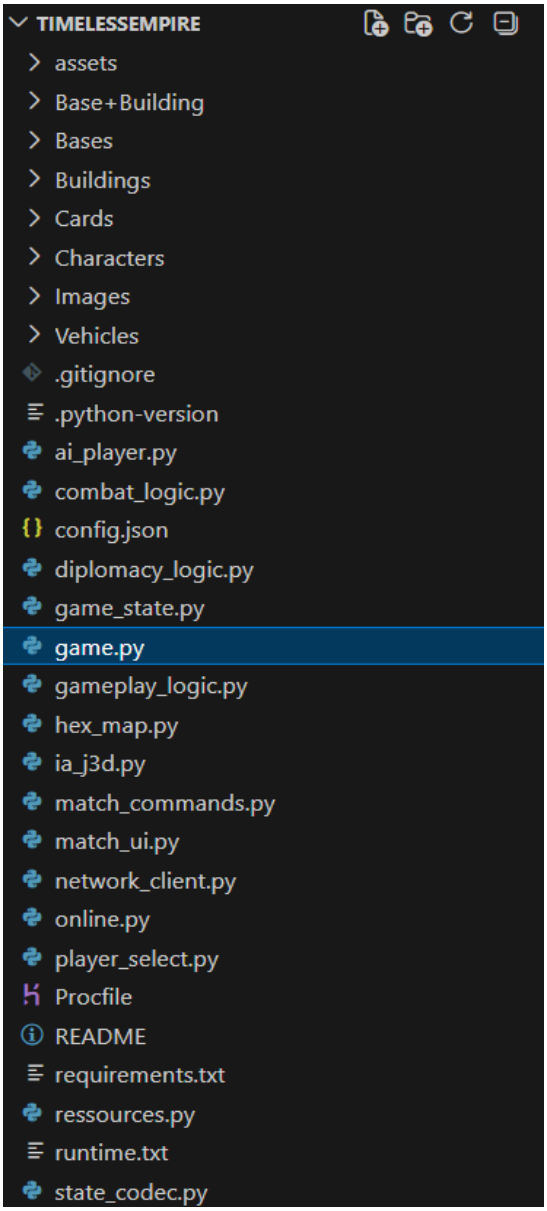
In parallel with the artistic work, my role as team leader helped me gain valuable experience in project coordination, communication, and team management.

Overall, this project was a rewarding experience that allowed me to contribute both creatively and organizationally to the development of the game.

III - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - QUENTIN REDT-ZIMMER

1 - Reorganization of the Project Architecture

I dedicated an important part of my work to reorganizing the project architecture in order to make the code clearer, easier to maintain, and more scalable. As the development progressed, the initial structure of the project became harder to manage because of the increasing number of files and features. I therefore carried out a significant restructuring process to create a more coherent organization for the project.



The different visual and audio elements were reorganized to make them easier to manage and integrate into the game. The resources related to menus, music, buildings, and terrain were arranged in a cleaner and more logical structure. At the same time, I reworked several central components of the program in order to make the overall functioning clearer and easier to modify.

This reorganization was also intended to prepare the project for more complex features, especially the development of the multiplayer mode. By improving the structure of the code, I made the project more stable, more readable, and easier to evolve for the next stages of development.

2 - Development of the Multiplayer Mode

One of the main areas of my work was the development of the multiplayer mode. My goal was to allow several players to take part in the same game while maintaining a coherent and smooth experience.

To achieve this, I developed different systems to handle communication between players and synchronization of game information. I adapted the internal logic of the project so that it could support several participants at the same time and correctly manage the different phases of a multiplayer game.

I also added new game states to handle connections, waiting rooms, and the launch of a game. Specific interfaces were added to allow players to host or join a game session more easily. At the same time, I implemented a system to transmit and synchronize the game state between the participants, in order to ensure a stable and coherent shared experience.

3 - Transition to a Public Online Multiplayer System

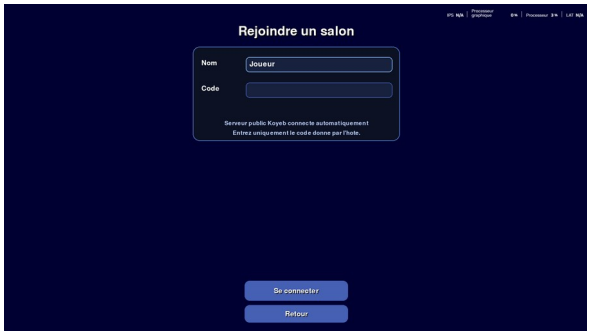
During the development of the multiplayer mode, I first made a mistake in the technical approach I chose. My goal was to create an online multiplayer mode that could be accessed remotely, but I initially developed an architecture designed for a local network, or LAN. In practice, this meant that players had to be connected to the same network in order to join a game, which did not match the real needs of the project.

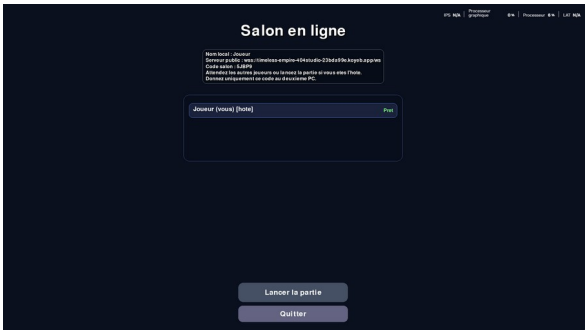
I then realized that, in order to offer a true online multiplayer mode, it was necessary to use a publicly accessible server architecture, capable of centralizing connections and exchanges between players. I therefore had to rework a significant part of the network system in order to move away from the local logic and replace it with a system based on a remote server.

This transition required several important technical adaptations. I implemented a system allowing players to create or join game rooms using connection codes, while also developing the mechanisms needed to handle communication between the client and the server. I also worked on deploying the server in order to make the multiplayer mode accessible outside of a local network and improve the accessibility of the project.

Although this design mistake made me lose time at the beginning, it allowed me to better understand the difference between local multiplayer and a real online multiplayer system, as well as the technical constraints linked to network architectures. This evolution is now an

important step for the project, as it makes it possible to aim for a multiplayer experience that is truly accessible remotely





4 - Visual Rework of the Map and Graphic Elements

I also worked on improving the visual aspect of the game, especially the map and the environments. The goal was to make the game world more visually coherent while improving its overall aesthetic quality.

This evolution was made possible thanks to Yiru's work, as he provided me with new sprites intended to replace the old graphic elements of the project. These new visual resources allowed me to significantly improve the overall rendering of the game and to create a more coherent artistic direction.

I then integrated these new graphic elements into the project in order to bring more variety to the terrain tiles on the map. I also reworked the rendering of the buildings so that they would blend more naturally with the different tiles and their environment. This work resulted in a more homogeneous, immersive, and visually pleasant display for the player.

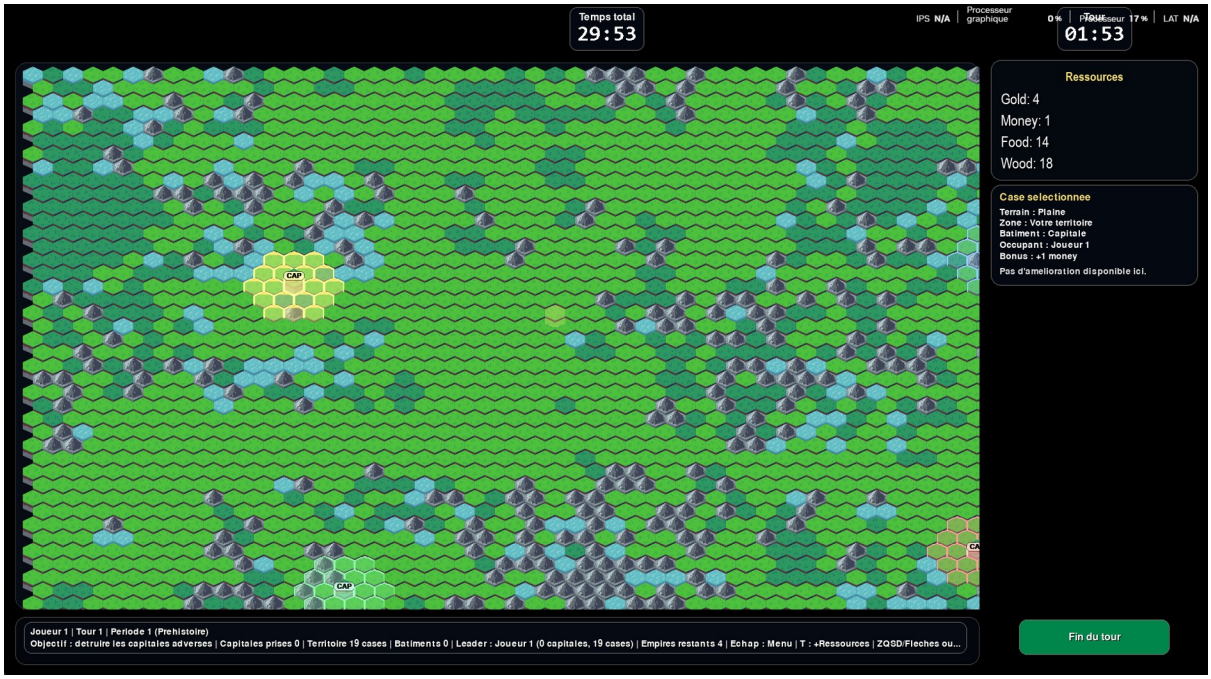
Beyond the aesthetic aspect, these improvements also helped make certain game elements easier to identify, improving the general readability of the map during a game.

5 - Improvement of the User Interface

Finally, I made several improvements to the user interface in order to improve gameplay comfort and the readability of the information displayed on screen. Some interface elements tended to overlap or take up too much space, which could make the information harder for the player to understand.

I therefore reworked different panels so that they would become more adaptive depending on the size of their content. Long texts are now handled more effectively to avoid visual overflow, and some elements were repositioned to create a clearer screen organization.

I also reworked some interfaces related to the multiplayer lobby and the information displayed during a game in order to reduce overlaps and improve the overall ergonomics. These changes made the game more pleasant to use and helped give the project a more polished and complete feel.



Conclusion

After the second presentation, I mainly focused my work on four major areas: reorganizing the code architecture, developing the multiplayer mode, transitioning toward a real online multiplayer infrastructure, and improving the visual aspect of the game and its user interface.

These different improvements allowed me to strengthen the technical structure of the project, expand the features offered to players, and improve the general visual quality of the game. They also provide a stronger foundation for continuing the development of *Timeless Empire* in better conditions and for integrating new features more easily in the future.

IV - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - RORY BUENO

This project represents one of the most important programming experiences I have worked on so far. Before joining this project, I already had experience in video game development thanks to several personal and school projects completed during the NSI specialization in high school. These previous experiences gave me a solid understanding of programming fundamentals, game logic, and software organization. However, Timeless Empire differs greatly in terms of scale and complexity because it is my first long-term collaborative project involving multiple interconnected systems developed by several team members simultaneously.

Timeless Empire is a multiplayer strategy game in which players manage civilizations through different historical eras. The project combines strategic gameplay and resource management. Because the game is designed around interactions between several players, the implementation of an online multiplayer system became one of the central technical challenges of the project.

My primary responsibility within the team was the implementation of the multiplayer architecture and the server systems used to synchronize the game between players. This role required not only programming knowledge but also careful organization and communication with the rest of the team because the networking systems directly interact with almost every gameplay mechanic in the project.

This report presents the work I completed during the development of the online multiplayer system. It explains the architecture choices, synchronization mechanisms, server management, technical challenges, and the improvements implemented throughout the project. It also describes the skills I developed and the experience I gained while working on this large-scale collaborative game development project.

1 - Objectives of the Multiplayer System

The first objective of the multiplayer system was to allow several players to participate in the same game session while ensuring that the game remained synchronized for every participant. Unlike a single-player game, where all systems are executed locally on one machine, multiplayer games require continuous communication between different computers connected through a network.

In Timeless Empire, synchronization is particularly important because the game is based on strategy and decision-making. Every action performed by one player can directly affect the other players and the overall state of the game. If one player sees a different game state than the others, the gameplay immediately becomes inconsistent and unplayable.

The multiplayer system therefore needed to guarantee several important elements:

- All players must observe the same game events.
- Actions performed by one player must be transmitted correctly to all other players.
- The game state must remain synchronized during the entire session.
- The network communication must remain efficient to avoid unnecessary lag or performance problems.
- The system must support multiple players simultaneously while remaining stable.

Another important objective was scalability. The multiplayer architecture needed to be modular enough so that new gameplay systems could later be integrated without completely redesigning the networking structure. Since the project continues to evolve, flexibility was an important design consideration from the beginning.

The multiplayer system also needed to integrate naturally with the rest of the project. Existing gameplay systems initially developed for local gameplay had to be adapted to support online synchronization without breaking their functionality.

Finally, another objective was to maintain good code organization and readability. Because networking systems can quickly become difficult to manage in large projects, I focused on creating a structure that remained understandable and maintainable for the entire team.

2 - Client–Server Architecture

To implement the multiplayer system, I chose to use a client–server architecture. This model is one of the most common architectures used in multiplayer games because it centralizes communication and allows better control of synchronization between players.

In this system, each player runs the game locally on their own computer. These local game instances act as clients. A central server is responsible for managing the communication between all connected clients.

When a player performs an action, the action is first sent to the server. The server then verifies and redistributes the information to all other connected players. Each client receives the action and applies it locally in the game.

This architecture offers several advantages:

- It centralizes communication.
- It simplifies synchronization management.
- It reduces inconsistencies between players.
- It allows better control over player actions.
- It makes debugging easier compared to peer-to-peer systems.

The server therefore acts as the authoritative source of information for the multiplayer session. Instead of allowing clients to communicate directly with each other, all communication passes through the server.

The architecture was designed to support up to five players simultaneously. This limit corresponds to the intended gameplay structure of Timeless Empire, where several civilizations compete and interact during the same match.

Another important aspect of the architecture was separating networking logic from gameplay logic. I organized the code so that communication systems remained relatively independent from gameplay systems. This modular approach improves maintainability and allows future modifications to be implemented more easily.

The client–server model also simplified the management of future features such as matchmaking, lobby systems, or dedicated servers, even if these systems are not yet fully implemented in the current version of the project.

3 - Action Synchronization System

One of the most important technical aspects of the multiplayer implementation was synchronization.

The main challenge was ensuring that every player always observes the same game state despite the fact that each client executes the game locally on a separate machine. To solve this problem, I implemented an action-based synchronization system.

Instead of sending the entire game state continuously through the network, the system only transmits the actions performed by players. For example, if a player selects a card, moves a unit, or triggers an event, only the information describing this action is sent to the server.

The server then redistributes this information to all connected clients. Each client receives the action and reproduces it locally inside the game.

This approach presents several advantages:

- Reduced network traffic.
- Faster communication.
- Easier synchronization management.
- Better scalability.
- Improved overall performance.

Sending the complete game state continuously would require much more bandwidth and would significantly increase synchronization complexity. By only sending player actions, the system remains lightweight and efficient.

To ensure consistency, each action contains several important pieces of information:

- The type of action performed.
- The identifier of the player responsible for the action.
- The gameplay parameters associated with the action.
- Timing or turn information when necessary.

The synchronization system also needed to ensure that actions were processed in the correct order. In a strategy game, execution order is extremely important because gameplay logic often depends on previous actions.

Careful testing was necessary to verify that all clients remained synchronized during long multiplayer sessions. I repeatedly tested scenarios involving simultaneous actions, turn transitions, and complex gameplay interactions to ensure stability.

4 - Server Management and Player Connections

Another important part of my work involved server management and player connection handling.

When a player joins a multiplayer session, the server assigns them a unique identifier. This identifier allows the system to track which player performs each action and ensures that gameplay events remain associated with the correct participant.

The connection management system also handles:

- Player joining.
- Player disconnection.
- Session initialization.
- Communication routing.
- Basic synchronization checks.

Managing player connections may appear simple initially, but it becomes increasingly important in multiplayer environments because network instability can easily create synchronization problems.

I also worked on ensuring that the server remains stable while multiple players interact simultaneously. The server continuously listens for incoming messages from clients and redistributes the necessary information efficiently.

Particular attention was paid to communication reliability. Since multiplayer synchronization depends entirely on network communication, it is essential that messages are processed correctly and consistently.

Another important objective was minimizing unnecessary communication. The server only sends information relevant to gameplay synchronization in order to avoid overloading the network.

The multiplayer system was also designed so that future improvements could later include additional server-side validation systems or more advanced session management features.

5 - Integration with Gameplay Systems

Integrating the multiplayer architecture into the existing game systems represented one of the most complex parts of the project.

Initially, many gameplay mechanics were designed for local execution only. This means that several systems had to be modified so they could function correctly within a multiplayer environment.

Gameplay systems affected by networking integration included:

- Turn management.
- Card interactions.
- Unit actions.
- Resource management.
- Event systems.
- Player interactions.

Each of these systems needed to communicate correctly with the networking layer while remaining functional locally.

One of the main difficulties was ensuring that gameplay logic remained deterministic. Since every client executes actions locally, all clients must always produce the exact same result from the same action. Even small inconsistencies can eventually create desynchronization problems between players.

To solve this issue, I carefully structured the communication flow between gameplay systems and networking systems. Actions are validated and distributed consistently before being applied locally.

Another important aspect was modularity. I organized the code so that multiplayer functionality could remain relatively independent from the gameplay logic itself. This reduces coupling between systems and simplifies future maintenance.

Working on this integration phase helped me better understand the complexity of large-scale software architecture and the importance of clean code organization in collaborative projects.

6 - Debugging, Testing, and Optimization

Testing multiplayer systems is significantly more difficult than testing local gameplay systems because synchronization issues are not always immediately visible.

A major part of my work therefore involved debugging and testing the online system.

I performed many multiplayer test sessions to verify:

- Synchronization stability.
- Correct action execution.
- Turn consistency.
- Network reliability.
- Connection stability.
- Gameplay coherence between clients.

One of the main challenges was identifying desynchronization sources. Sometimes, a small inconsistency in gameplay logic could progressively create larger synchronization problems between players.

To reduce these issues, I added debugging tools and synchronization checks during development. These tools helped identify differences between clients and allowed faster correction of networking problems.

Performance optimization was another important aspect of the project. Multiplayer systems can quickly become inefficient if too much information is exchanged between clients and servers.

By transmitting only gameplay actions instead of the full game state, the system remains relatively lightweight and efficient. This optimization significantly reduces bandwidth usage while maintaining synchronization quality.

I also improved code readability and organization during the debugging phase. Clear code structure greatly simplifies multiplayer debugging because networking systems can otherwise become extremely difficult to maintain.

7 - Skills Developed During the Project

Working on Timeless Empire allowed me to significantly improve several technical and organizational skills.

From a technical perspective, I deepened my knowledge in:

- Network programming.
- Client–server architectures.
- Synchronization systems.
- Multiplayer game development.
- Software modularity.
- Debugging large systems.
- Collaborative programming.

Before this project, I already had experience with programming and game logic, but this was my first time implementing a complete multiplayer architecture in a large collaborative game project.

The project also strengthened my understanding of software engineering principles such as modularity, maintainability, and scalability.

Beyond programming itself, this experience improved my teamwork and communication skills. Multiplayer systems interact with many other gameplay systems developed by different team members, so collaboration and coordination were essential throughout development.

Using version control systems and maintaining clean, readable code also became extremely important in order to facilitate integration between all team members.

Overall, this project gave me valuable experience both technically and professionally.

Conclusion

The implementation of the multiplayer system and server architecture represented one of the most technically advanced aspects of the Timeless Empire project.

My work focused on transforming the game from a local experience into a synchronized online multiplayer strategy game capable of supporting several players simultaneously. To achieve this, I implemented a client–server architecture, synchronization systems, player connection management, and communication structures designed to ensure stable multiplayer gameplay.

One of the most important design decisions was using an action-based synchronization model, where only player actions are transmitted instead of the entire game state. This approach significantly improves efficiency while maintaining synchronization consistency between players.

Integrating multiplayer systems into the existing gameplay architecture also required substantial modifications and careful code organization to ensure compatibility with all gameplay mechanics already developed by the team.

Throughout the project, I gained valuable experience in networking, software architecture, debugging, optimization, and collaborative development. This experience allowed me to deepen both my technical programming skills and my understanding of large-scale project organization.

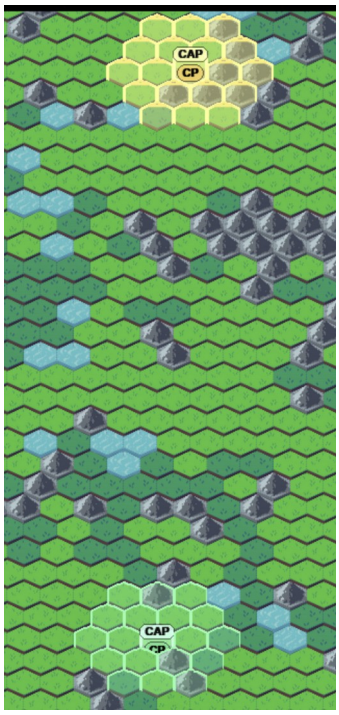
Overall, working on the multiplayer and server systems was a highly rewarding experience that contributed significantly to my growth as a developer and gave me practical experience with real multiplayer game development challenges.

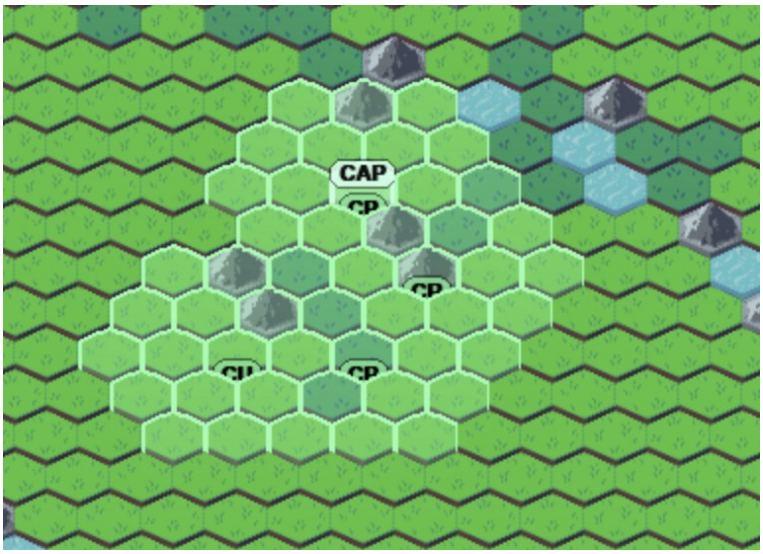
V - FINAL DEFENSE REPORT : INDIVIDUAL CONTRIBUTION - MATHIS CONSTANT

1. Improvement of the Solo Mode

One of my main gameplay contributions was the redesign and improvement of the solo mode system. Originally, the game used to have no AI opponents. This meant that every solo game automatically started with no one else in the game than the player itself and lasted until the timer reached zero, regardless of the player's moves.

Luka had tried to implement one which was making a random choice between the possibilities it had. But our game is not a random game, the player hasti make wise choices to win the game, so should the ia. To make this work and give a sense for the solo mode, I had to implement a new strategic ai which will be able to operate strategically to try winning the game against opponents. It took me a long time and a lot of researches but after many troubles which broke the game more than a few times, i managed to get something working, something which is interacting kind of randomly with its environment but making moves regardless of how stupid they can be.





Once I had found a way to not launch a game alone, I was concerned with the fact that TimelessEmpire is a 5 player game, not a game to play 1vs1 against a bot.

To solve this issue, I implemented a new system that allows players to choose the number of AI opponents before starting a match. Instead of launching the game immediately after clicking the “New Game” button, the game now opens a dedicated selection screen where the player can configure the number of AI participants.

This new interface was designed to remain simple and intuitive. The player can increase or decrease the number of AI opponents using “+” and “–” buttons displayed on the screen. The currently selected value appears clearly in the center of the menu, making the interaction easy to understand. The allowed range goes from zero to three AI opponents, which corresponds to the actual gameplay limitations supported by the project.

This improvement also required updating the internal game setup process inside the main game management system.

From a gameplay perspective, this modification greatly improves replayability and accessibility. Some players may prefer a simpler game with fewer opponents, while others may want a more difficult and crowded match with several AI enemies. The new system allows the same game mode to provide multiple gameplay experiences depending on player preference.

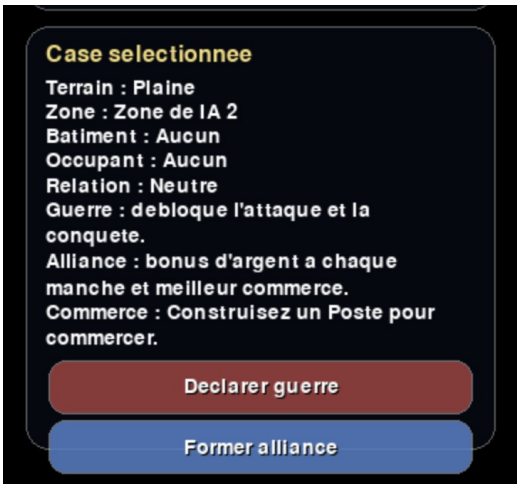
In addition, this contribution improved the maintainability of the project itself. Replacing hard-coded values with configurable systems is an important software development practice because it reduces rigidity and simplifies future modifications. If the team later decides to increase the maximum number of AI players or redesign the balancing system, the current architecture will make these changes much easier to implement.

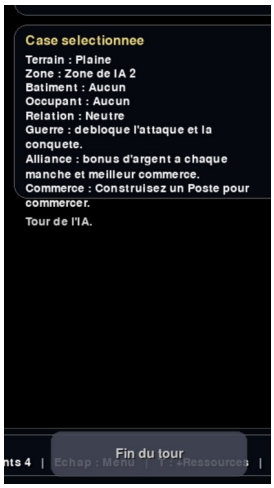
Overall, this modification made the solo mode more modern, flexible, and user-oriented while also improving the technical quality of the codebase.

2. Dynamic Game Initialization and Technical Adaptation

Beyond the visible gameplay improvements, a large part of my work focused on adapting the internal game initialization process to support the new solo mode configuration system. This contribution was highly technical because it involved modifying the way the game creates and manages player entities during the beginning of a match.

One of the most important technical challenges involved ensuring compatibility with the rest of the project. Since many gameplay systems depended on player lists and player indexing, even small structural modifications could potentially create errors in collision systems, score tracking, turn management, or interface updates, for example the fact that you cant see other players ressources, cant skip the turn, cant do anything on the tiles etc.. I therefore had to carefully test different configurations in order to verify that the game remained stable regardless of the selected number of AI players.





Another important improvement involved reducing hard-coded dependencies inside the project. Hard-coded systems often become problematic as projects grow larger because they reduce flexibility and increase maintenance difficulty. By introducing dynamic player management, I helped make the project architecture more modular and scalable.

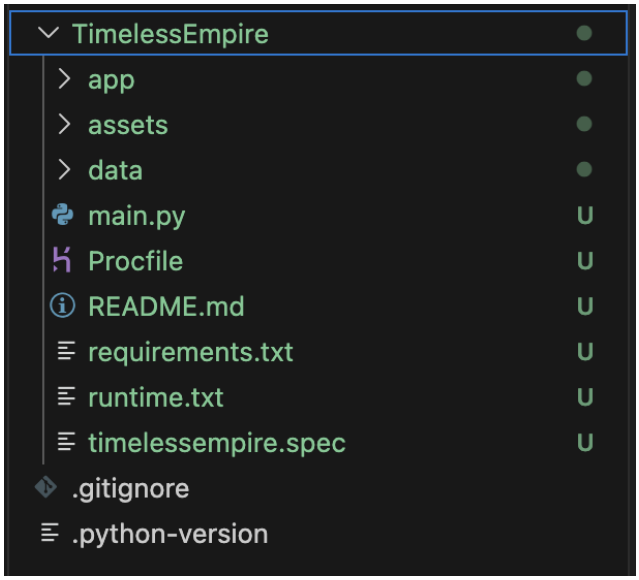
This work also improved the long-term extensibility of the game. Future developers will now be able to modify the number of supported players or introduce additional game configurations

more easily because the initialization logic is no longer restricted to a single predefined structure.

Overall, this technical contribution strengthened the internal flexibility of the project and improved the overall quality of the game architecture while supporting the new gameplay customization features introduced in the solo mode.

3. Reorganization of the Project Architecture

Another major contribution I made to the project was the complete reorganization of the file and folder structure. As development progressed and more systems were added to the game, the original project organization became hard to manage even though Quentin had done a great job to regroup most of the files to make them more structured and have a better and more organized coding. Still, several resources were stored in inconsistent locations, and the overall architecture lacked clear separation between different components of the game. However, the way the project was stored worked very well for us, it was not an issue at all, but for a finished project, i had to redo the structure once again before having to deal with an executable.



The new architecture separates the different aspects of the game into dedicated folders. Gameplay systems, menus, graphical assets, sounds, utilities, and configuration files are now grouped logically according to their purpose. This greatly improves readability because developers can quickly identify where each component belongs.

One important advantage of this restructuring is improved maintainability. A well-organized project is significantly easier to update and debug because developers can navigate the codebase more efficiently. When files are scattered randomly throughout the project, maintenance becomes slower and more error-prone. By introducing a clearer architecture, I helped reduce this complexity.

This reorganization also improved teamwork and collaboration efficiency. In a collaborative project, multiple developers often work on different systems simultaneously. A cleaner structure reduces confusion and lowers the risk of accidental modifications in unrelated systems. It also simplifies version control management because files are categorized more logically.

Another important aspect of this contribution involved standardizing resource management. After moving files into new folders, I updated multiple import statements and asset-loading paths so that all textures, sounds, and menus could still be located correctly by the program. This required carefully verifying that every resource remained functional after the restructuring process.

Although this work was less visible from the player’s perspective, it had a major impact on the professionalism and scalability of the project. Software architecture is a critical aspect of game development because poorly organized projects become increasingly difficult to maintain as they grow larger. By improving the internal structure of the game, I helped create a cleaner foundation for future development.

Overall, this contribution significantly improved the clarity, organization, and maintainability of the project while making collaboration between team members more efficient.

4. Preparation for Executable Packaging and Deployment

One of the most important technical objectives of the project was preparing the game to be converted into a standalone executable application. During development, programs are usually executed directly from source files inside a development environment. However, creating a distributable executable introduces additional technical challenges because all project resources must be packaged correctly and remain accessible after compilation.

I contributed significantly to this preparation process by reorganizing the project structure and ensuring that resource management systems would remain compatible with executable packaging tools. Without proper preparation, executable generation often leads to missing textures, broken sound paths, import errors, or unstable runtime behavior.

A major part of this work involved standardizing file paths and improving dependency organization throughout the project. Since resources had previously been stored in inconsistent locations, some systems relied on fragile or hard-coded paths that could easily break once the project was packaged into an executable file. I therefore updated several resource-loading mechanisms to make them more stable and adaptable.

The reorganization of folders also played an important role in this deployment preparation. Packaging tools generally work more efficiently when the project follows a clean and predictable structure. By separating assets, scripts, and utilities into dedicated directories, the game became much easier to package and distribute.

Another challenge involved ensuring that all imports remained functional after restructuring the project. Python projects can encounter import resolution problems when files are moved between directories, especially when external tools attempt to compile the project into a standalone executable. I therefore carefully updated and verified multiple import systems to ensure compatibility.

This contribution also improved the long-term portability of the project. A properly structured game is easier to share, test, and deploy across different environments. Even though executable generation may appear to be a final step in development, preparing for deployment actually influences many architectural decisions throughout the project lifecycle.

From a professional perspective, learning how to prepare software for deployment was an important experience. It demonstrated that software engineering does not only involve creating gameplay features, but also ensuring that the final product can be distributed reliably to users.

Overall, this contribution strengthened the technical stability of the project and helped prepare the game for future executable distribution and deployment.

5. Conclusion and Skills Developed During the Project

This project allowed me to contribute to both gameplay systems and technical infrastructure while developing important programming and software engineering skills. Through my different contributions, I gained experience not only in implementing new features, but also in improving software organization, scalability, and maintainability.

One of the most valuable aspects of this project was learning how gameplay design and technical architecture are closely connected. For example, the solo mode improvement initially appeared to be a simple gameplay modification, but it actually required significant changes to the game initialization system and player management architecture. This demonstrated how even small gameplay features can have important technical implications.

I also learned the importance of writing flexible and modular code. Hard-coded systems may work during early development stages, but they quickly become problematic as projects grow larger. By replacing fixed configurations with dynamic systems, I helped make the project more adaptable and easier to maintain in the future.

Another major lesson involved project organization and teamwork. Working on a collaborative codebase requires clear structures, understandable architectures, and efficient communication between developers. The complete reorganization of the project files showed how software structure can directly impact development efficiency and collaboration quality.

In addition, this project gave me practical experience with deployment preparation and executable packaging considerations. Understanding how resource paths, imports, and dependencies behave outside the development environment is an important aspect of professional software development that is often underestimated during small projects.

Throughout the project, I also improved my problem-solving abilities. Many modifications required debugging, testing, and adapting existing systems without breaking compatibility with other parts of the game. This process helped me develop a more rigorous and methodical approach to programming.

Overall, this project was a very valuable experience because it combined technical implementation, software organization, teamwork, and practical development challenges. My contributions helped improve both the player experience and the internal quality of the project by building an ai player from scratches, while also allowing me to strengthen my programming and software engineering skills considerably.

VI - INSTALLATION MANUAL

1 - VIA THE WEBSITE

- Go on https://timeless-empire.fr
- Click on “THE PROJECT”
- Scroll all the way down and click on “Download Full Project Archive (ZIP)”
- Go on “Download” and unzip the file
- Launch the executable (TimelessEmpire)
- You are now ready to play

2 - VIA USB KEY

- Plug it to your PC
- Copy the folder
- Launch the executable (TimelessEmpire)

VII - USER MANUAL

1 - INTRODUCTION AND GAME PHILOSOPHY

Timeless Empire is a multiplayer, turn-based strategy game designed for 3 to 5 players. Developed entirely in Python using the Pygame library, the project bypasses external commercial game engines to focus on core computer science and logical architectures.

Unlike traditional real-time strategy games that reward high action-per-minute (APM) and fast clicking reflexes, *Timeless Empire* addresses a specific user need: the need to relax and engage in complex macro-management. The game is built as a satire of human warfare, highlighting the historical absurdity and long-term economic consequences of repetitive conflicts. It is primarily intended for players aged 16 and up who possess the critical thinking required to balance territorial expansion with diplomacy.

2 - USER INTERFACE AND GENERAL NAVIGATION

To ensure accessibility and clean usability, the user interface relies entirely on mouse-based positioning and standard clicks rather than complex keyboard shortcuts.

The Main Menu Screen

Upon launching the application, you are presented with four responsive navigation buttons overlaid on a thematic pixel-art background:

- Play: Launches a local multiplayer testing environment to verify game logic.
- Multiplayer: Initiates client networking connection to link with the online server.
- Options: Allows configuring the visual constraints, specifically display resolution sizes (e.g., *1280x720*, *1600x900*, *1920x1080*, *2526x1440*).
- Quit: Safely terminates the game instance and closes the application window.

3 - THE MULTIPLAYER LOBBY SYSTEM

Selecting the online multiplayer mode connects your game client to the central server hosted on Koyeb.

1. Players are routed to the Lobby Screen (Salon d'attente).
2. The lobby tracks connected participants, ensuring network synchronization before the match starts.
3. Connected users appear sequentially in the lobby frame with a green Connecté indicator status.
4. Once all required players are ready, the host clicks Lancer la partie to generate the synchronized map state across all computers.

4 - CORE GAME LOOP AND SESSION FLOW

Matches advance through a strictly enforced execution loop managed by a central TurnManagerbackend structure:

- Turn-Based Timing Constraints: Each player executes actions individually during their allocated turn. A strict 2-minute timer runs concurrently. If actions are completed early, clicking the Cliquez End Tour banner instantly shifts active priority to the next client.
- Global Session Limits: To preserve processing stability and avoid deadlock states, an overall session timer is capped at 30 minutes per match.
- Interactive Grid Map: The world map is rendered as a cohesive grid of hexagonal territories. Moving your cursor over a territory triggers a highlight border effect. Left-clicking a territory causes the tile to lift vertically, confirming its selection and unlocking contextual building or positioning actions.

5 - DETAILED FUNCTIONAL MECHANICS

5.1 - Resource Dependency System

Players must balance three fundamental economic resources; Gold, Wood, and Food; alongside a derived currency system (Silver/Money).

The game executes a strict logical dependency constraint : base resource income is mapped directly to infrastructure ownership. For instance, a player cannot generate or accumulate wheat or food supplies during turn transitions if they have not explicitly constructed a Farm on their territory. Resources are automatically incremented at the conclusion of each full round according to building production rates.

5.2 - Historical Eras

Empires progress dynamically across five continuous historical periods. Advancing to a higher age occurs every five turns.

- Prehistory & Antiquity: Marked by the discovery of fire. Players are confined to baseline infrastructure: Rock Fields (mines), Gathering Sites (farms), and Lumber Camps (sawmills).
- Middle Ages: Unlocks military combat functions. Unlocks Casernes (barracks), basic Soldiers, Cavalerie, and localized messaging via Carrier Pigeons.
- Modern Times: Expands advanced macro-economic structures. Introduces the Imprimerie (printing press for currency manipulation) and long-range Diligences.
- Contemporary Era: Introduces aggressive Overpopulation constraints. If a city grows disproportionately large, it suffers automated resource attrition. Unlocks mechanization assets: Véhicules and Aviation (missile strike capabilities).
- The Future: Continuity of the contemporary era, enable the players to fight a last time together

5.3 - Automated Combat Resolution

To prioritize macroscopic economic strategy over immediate clicking speed, manual real-time micro-management of individual fights is fully excluded from the software scope. Fights are handled as follows:

- You can choose to start a combat to gather one unit field whenever you are at war with the Empire you want to fight
- Fights are determined by fixed mathematical power level metrics corresponding to each soldier class and technological era. (more war builds = more power)
- In cases of equivalent army power baselines, a built-in random variable function acts as the final logical arbiter to determine the winning troop and apply updated damage parameters.

6 - VICTORY CONDITIONS

Achieving Ultimate Victory

Matches conclude instantly when one of two distinct structural validation criteria is achieved:

1. Total Military Elimination: A player invades and captures all opposing Capital cities, leaving their empire as the sole surviving civilization on the grid.
2. Cumulative Score Evaluation: Upon the exact expiration of the 30-minute global session limit, the engine parses individual data to find the highest accumulated value across two metrics: Battle Points (combat wins), Expansion Points (hex grid territory size).

7 - Building Cost/Reward

BUILDINGS	ERAS	COST	PRODUCTION
Capitale	Préhistoire	0	+1 Argent
Champ de Roche	Préhistoire	7 Bois	+4 Or
Cueillette	Préhistoire	6 Bois	+4 Nourriture
Camp de bûcherons	Préhistoire	7 Bois	+5 Bois
Mine	Moyen Âge	10 Bois, 1 Nourriture, 2 Or	+6 Or
Champs	Moyen Âge	8 Bois, 2 Nourriture, 1 Or	+7 Nourriture
Scierie	Moyen Âge	9 Bois, 1 Nourriture, 1 Or	+9 Bois
Caserne	Moyen Âge	12 Bois, 6 Nourriture, 4 Or	0 (Improve attack)
Élevage	Moyen Âge	9 Bois, 4 Nourriture, 1 Or	+5 Nourriture
Poste	Moyen Âge	8 Bois, 2 Nourriture, 3 Or	+3 Argent

Messagers et diligence	Temps Modernes	10 Bois, 5 Nourriture, 5 Or, 3 Argent	+5 Argent
Bâtiments de siège	Temps Modernes	14 Bois, 7 Nourriture, 7 Or	+2 Or (Improve attack)
Imprimerie	Temps Modernes	12 Bois, 2 Nourriture, 10 Or	+6 Argent
Aviation et véhicules	Époque Contemporaine	18 Bois, 8 Nourriture, 10 Or, 8 Argent	+3 Or, +2 Argent (Improve attack)
Ville fortifiée	Époque Contemporaine	16 Bois, 12 Nourriture, 8 Or, 12 Argent	+1 Nourriture, +3 Argent