Project Functional Specification Document

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: V4 – 16/03/2026

Table of contents

INDIVIDUAL PART - WEB & TOURS.PY

As part of our *Timeless Empire* project, I mainly worked on two important aspects: the creation of the Wiki page of the website, as well as the development of the turn and card system in the file *tours.py.*

I first added a new page to the website called “Wiki”, whose purpose is to clearly present all the cards that a player can obtain during a game. This page helps players better understand how the game works, while also making the project appear more complete and professional. It is not simply a decorative page: it genuinely explains the game mechanics and helps players understand the effects of the different cards.

To build this page, I started from the style already used on the website. The objective was for the Wiki page to integrate naturally with the rest of the project, without looking like a completely separate element. I therefore reused the general structure used on the other pages of the site: a header, a navigation bar, a hero section at the top of the page, and then content organized into several sections. The page is connected to the rest of the website through the `wiki.html` link that was added to the navigation menu.

```
<nav>
        <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="presentation.html">The Project</a></li>
                <li><a href="about.html">About Us</a></li>
                <li><a href="contact.html">Contact</a></li>
<li><a href="wiki.html">Wiki</a></li>
        </ul>
</nav>
```

The content of this page is organized into three main categories of cards: resource cards, building cards, and malus cards. This organization helps make the page easier to read and allows cards to be grouped according to their role in the game.

To display these cards, I used a visual card system based on the existing CSS class `feature-card`. Each card contains several elements: an icon, which allows the player to quickly identify the type of card, a title, a summary of the effect, and a description explaining its concrete impact in the game.

```
<div class="feature-card resource-card">
    <span class="feature-icon">🌲</span>
    <h3>Wood Reserve</h3>
    <p class="card-effect">+10 Wood</p>
    <p>A hidden forest reserve is discovered, giving your empire a large supply of wood.</p>
</div>
```

To make the interface clearer, I then added several CSS styles to visually differentiate the types of cards. For example, resource cards use a green visual identity, building cards use gold tones, while malus cards are highlighted with more red colors to indicate their negative effect.

```css
/* resource cards */

.resource-card{
    border-top: 3px solid #6fd98c;
}

.resource-card:hover{
    box-shadow: 0 10px 20px rgba(111,217,140,0.2);
}

/* building cards */

.building-card{
    border-top: 3px solid #ffd27a;
}

.building-card:hover{
    box-shadow: 0 10px 20px rgba(255,210,122,0.25);
}

/* malus cards */

.malus-card{
    border-top: 3px solid #ff6b6b;
}

.malus-card:hover{
    box-shadow: 0 10px 20px rgba(255,107,107,0.3);
}
```

I also added hover effects to make the page more dynamic. When the mouse passes over a card, the card slightly lifts, its shadow changes, and the icon grows slightly larger. These effects improve the user experience and give the page a more modern look, similar to what could be found in a real video game wiki interface.

This part of the project was particularly enjoyable to work on because it quickly produces visible results. It is satisfying to move from a simple idea presenting the game cards to a fully functional web page that fits the visual identity of the project and is pleasant to explore. I also appreciated the connection between content and design: on one hand I was explaining the gameplay, and on the other I was working on the appearance and usability of the website.

At the same time, I also developed the game's turn system in the file `tours.py`. This part is more technical because it 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 several cards, each with an immediate effect.

The general principle of the system is relatively simple: at each turn, the player receives three card options. There is always one resource card, one building card, and one malus card. The player must then choose one of these cards, and its effect is applied immediately. This mechanic introduces a strategic dimension, because the player constantly needs to decide between improving their economy, developing infrastructure, or managing the risks associated with negative events.

In the code, this logic relies on several important components. First, the `Player` class represents a player and contains all the information related to that player: their name, their resources (wood, food, and gold), and the buildings they own.

```python
class Player:

    def __init__(self, name: str):

        self.name = name

        self.resources = {
            "wood": 0,
            "food": 0,
            "gold": 0
        }

        self.buildings: List[str] = []
        self.trapped_buildings: Set[int] = set()
```

This class also manages several important actions, such as receiving resources, constructing buildings, or destroying a building when a malus occurs. This part of the code illustrates the object-oriented programming logic used in the project: each player has their own state, and different methods are used to modify that state.

The functioning of the game also relies on several global constants that define important rules. For example, I defined the number of turns required to move to a new period, the cost of

different buildings, the buildings available in each period, and the base income received at the end of each round.

```
TURNS_PER_PERIOD = 5

BASE_RESOURCE_INCOME = {
    "wood": 5,
    "food": 5,
    "gold": 2
}

BUILDING_COSTS = {
    "farm": {"wood": 10, "food": 0, "gold": 0},
    "barracks": {"wood": 20, "food": 5, "gold": 0},
    "blacksmith": {"wood": 30, "food": 10, "gold": 5},
}

BUILDINGS_BY_PERIOD = {
    1: ["farm"],
    2: ["barracks"],
    3: ["blacksmith"]
}
```

The card system itself is obviously a central part of the file. Cards are divided into three lists: `RESOURCE_CARDS`, `BUILDING_CARDS`, and `MALUS_CARDS`. Each card contains information such as its name and its effect, which allows different cards to be randomly generated each turn.

```
RESOURCE_CARDS = [
    {"name": "Réserve de bois", "effect": {"wood": 10}},
    {"name": "Chasse abondante", "effect": {"food": 10}},
    {"name": "Mine d'or découverte", "effect": {"gold": 6}},
    {"name": "Caravane marchande", "effect": {"wood": 5, "food": 5}},
]

BUILDING_CARDS = [
    {"name": "Construction rapide", "building": "farm"},
    {"name": "Caserne offerte", "building": "barracks"},
]

MALUS_CARDS = [
    {"name": "Incendie", "type": "fire"},
    {"name": "Tempête", "type": "storm"},
    {"name": "Sabotage", "type": "destroy_building"},
    {"name": "Voleurs", "type": "steal_gold"},
    {"name": "Bâtiment piégé", "type": "trap_building"},
]
```

A function then generates the three cards offered to the player during each turn. It selects one card from each category and then shuffles their order so that the choices remain unpredictable.

```
def generate_cards():

    cards = []

    cards.append(("resource", random.choice(RESOURCE_CARDS)))
    cards.append(("building", random.choice(BUILDING_CARDS)))
    cards.append(("malus", random.choice(MALUS_CARDS)))

    random.shuffle(cards)

    return cards
```

Once the player chooses a card, another function applies its effect. In the case of a resource card, the player simply receives additional resources. In the case of a malus card, negative events may occur.

```
def apply_card(player: Player, card_type, card):

    print("\nEffet de la carte :")

    if card_type == "resource":

        for res, value in card["effect"].items():
            player.resources[res] += value
            print(f"+{value} {res}")

    elif card_type == "building":

        building = card["building"]
        player.buildings.append(building)

        print(f"Bâtiment {building} ajouté gratuitement")
```

Some maluses also introduce more varied effects. For example, a fire can destroy part of the player's food supply, a storm can cause the player to lose wood, thieves can steal gold, and sabotage can destroy a building. I also added a trapped building system, which triggers a delayed destruction in a later turn.

Finally, the overall progression of the game is managed by the `TurnManager` class. This class handles the order of players, the progression of turns, and the distribution of resources.

```
class TurnManager:

    def __init__(self, players: List[Player]):

        self.players = players
        self._played_this_round: Set[Player] = set()

        self.turn_number = 0
        self.period = 1

        self.available_buildings = BUILDINGS_BY_PERIOD[1].copy()
```

It allows the game to determine which player should play next, mark when a player has finished their turn, detect when all players have played, and trigger the end of a round. At the end of each full round, players automatically receive additional resources and the game may progress to a new period.

```
def end_round(self):

    print("\n====== FIN DU TOUR ======")

    self.turn_number += 1

    for p in self.players:

        print(f"\nRessources pour {p.name}")

        bonus = {
            "wood": random.randint(0, 2),
            "food": random.randint(0, 2),
        }

        p.receive_resources(extra=bonus)
        p.receive_resources(extra=p.income_from_buildings())

        print(p.resources)

    if self.turn_number // TURNS_PER_PERIOD + 1 > self.period:

        self.period += 1
        print(f"\nNOUVELLE PÉRIODE : {self.period}")

    self._played_this_round.clear()
```

This part is particularly important in the program's architecture because it connects all the other game mechanics together. Without this system, the game would have cards, resources, and buildings, but no real structure to move the game forward. For this reason, it is one of the most interesting parts of the code to present, as it demonstrates how all the mechanics of the game are coordinated.

This work allowed me to contribute both to the presentation and the core mechanics of the project. The WIKI page makes the game clearer and more accessible, while the turn and card system in *tours.py* brings a real gameplay structure.
It was a valuable experience that combined web design and programming and helped me better understand how to build a project that is both functional and easy to understand.

INDIVIDUAL PART QUENTIN

During the last presentation, I explained that I was mainly focusing on the music and sound design of the game. However, after further reflection and discussion with the team, we concluded that these elements were not a priority at this stage of development. I therefore only implemented the initial beginnings of the game's music, and this work will be continued later when its integration becomes more relevant to the project's progress.

```python
def music_menu(music_file):
    try:
        if not pygame.mixer.get_init():
            return
        if not os.path.exists(music_file):
            return
        pygame.mixer.music.load(music_file)
        pygame.mixer.music.set_volume(0)
        pygame.mixer.music.play(-1)  # -1 pour une boucle infinie
    except pygame.error:
        pass


running = True
clock = pygame.time.Clock()
# Lance la musique du menu
music_menu(menu_music)
```

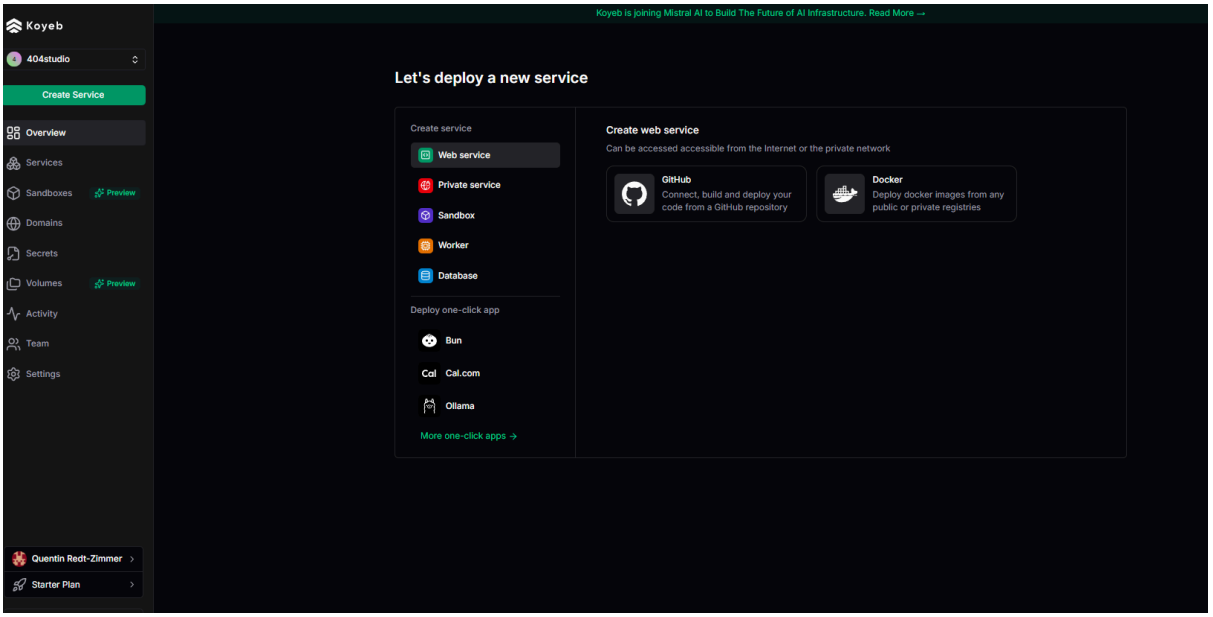
As a result, I focused more on aspects related to the game's programming.

First, I developed the interface for the main menu. This menu now allows the player to start the game, access certain settings, launch a multiplayer session, and exit the game. Implementing this menu was an important step in establishing a basic structure for the application. An initial version of this menu had already been started during the first presentation, but it was not yet functional at that time.

Next, I worked on the resource management system. I began by creating an overlay, which is currently temporary, that displays the resources owned by each player. These resources are updated at each turn thanks to the system previously developed by Luka. The main challenge was ensuring that the resources were properly assigned to each individual player and not shared between them.

Finally, in order to test these different features, I also attempted to implement a local multiplayer system. This system mainly serves as a testing tool and allows me to better understand the overall functioning of the system and the interactions between the different components of the game.

In addition, I set up a server to host the online multiplayer functionality using Koyeb. This server has been made available to Rory so that he can continue working on the implementation and management of the online multiplayer features.



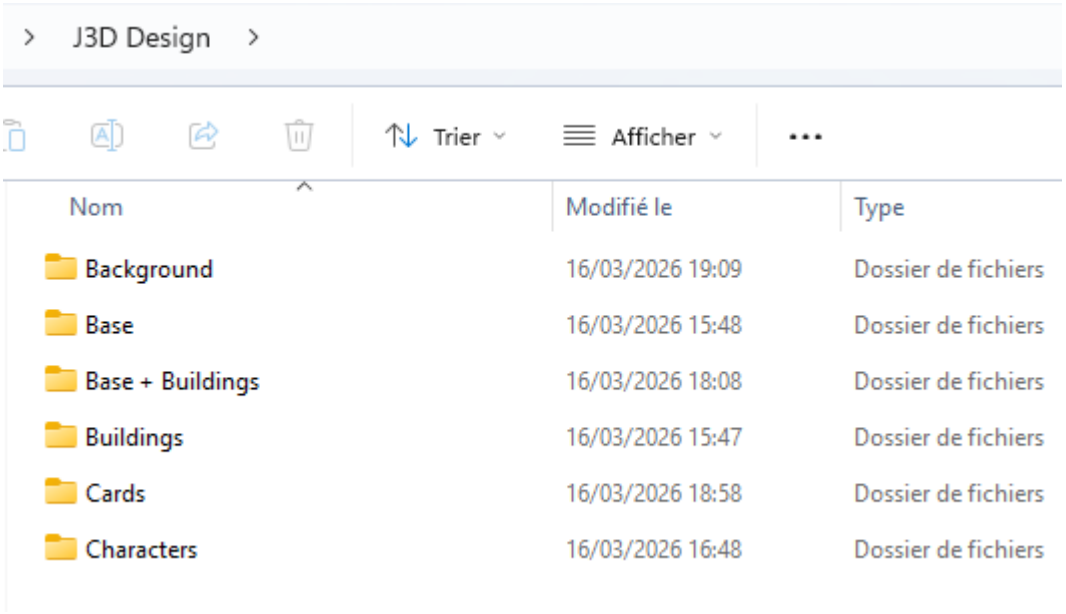
In conclusion, this phase of the project allowed me to establish several essential foundations for the game's development. By prioritizing core programming tasks over non-essential features such as sound design, I was able to contribute to the structuring of the application through the implementation of the main menu and the development of the resource management system.

These elements not only improve the current functionality of the game but also provide a solid basis for future developments. The implementation of a local multiplayer test environment, as well as the setup of an online server, has also contributed to a better understanding of the technical challenges related to multiplayer systems.

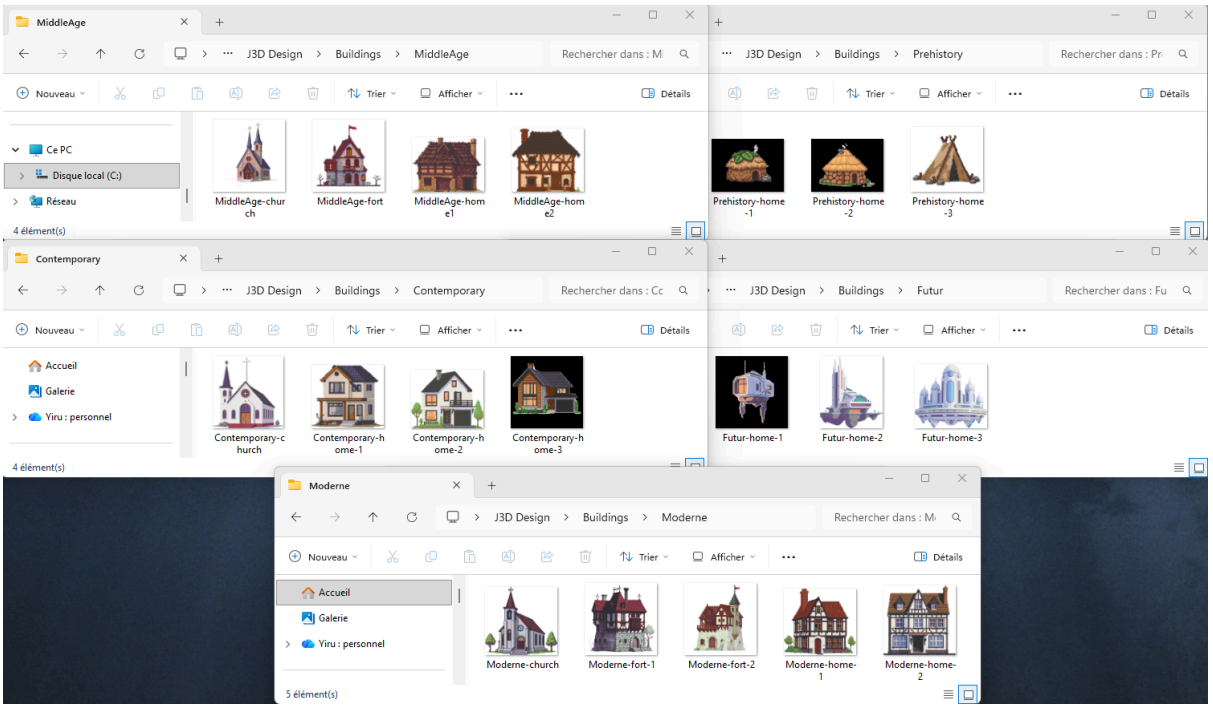
Overall, this work represents a significant step forward in the project, while also preparing the ground for the integration of more advanced features in the later stages of development.

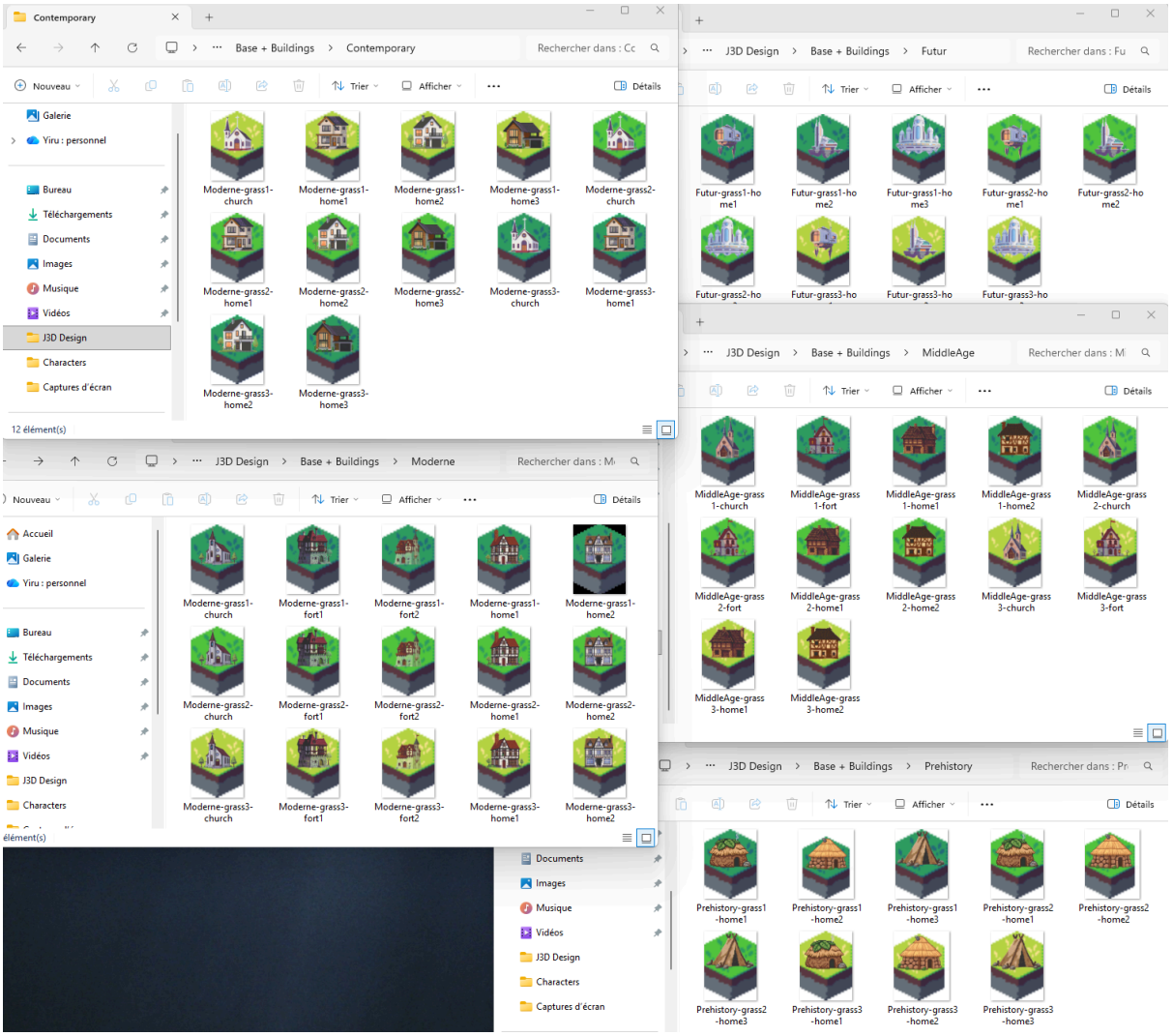
INDIVIDUAL PART YIRU

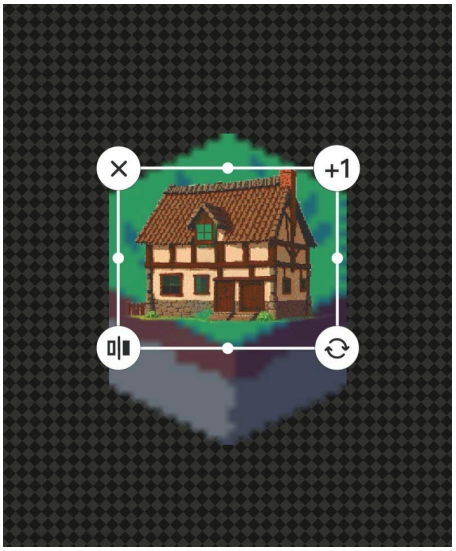
In this project, my main responsibility was the visual and artistic design of the game.



Our game is a strategy board game that evolves through different historical periods, so one of my main tasks was to design the visual identity of each era. I created the designs for all the buildings that appear throughout the different periods of the game. The goal was to make each building clearly recognizable and visually consistent with the time period it belongs to, while still keeping a coherent style across the whole game.




Contemporary
Base + Buildings > Contemporary
Moderne-grass1-church
Moderne-grass1-home1
Moderne-grass1-home2
Moderne-grass1-home3
Moderne-grass2-church
Moderne-grass2-home1
Moderne-grass2-home2
Moderne-grass2-home3
Moderne-grass3-church
Moderne-grass3-home1
Moderne-grass3-home2
Moderne-grass3-home3
J3D Design > Base + Buildings > Futur
Futur-grass1-home1
Futur-grass1-home2
Futur-grass1-home3
Futur-grass2-home1
Futur-grass2-home2
J3D Design > Base + Buildings > MiddleAge
MiddleAge-grass1-church
MiddleAge-grass1-fort
MiddleAge-grass1-home1
MiddleAge-grass1-home2
MiddleAge-grass2-church
MiddleAge-grass2-fort
MiddleAge-grass2-home1
MiddleAge-grass2-home2
MiddleAge-grass3-church
MiddleAge-grass3-fort
MiddleAge-grass3-home1
MiddleAge-grass3-home2
J3D Design > Base + Buildings > Moderne
Moderne-grass1-church
Moderne-grass1-fort1
Moderne-grass1-fort2
Moderne-grass1-home1
Moderne-grass1-home2
Moderne-grass2-church
Moderne-grass2-fort1
Moderne-grass2-fort2
Moderne-grass2-home1
Moderne-grass2-home2
Moderne-grass3-church
Moderne-grass3-fort1
Moderne-grass3-fort2
Moderne-grass3-home1
Moderne-grass3-home2
J3D Design > Base + Buildings > Prehistory
Prehistory-grass1-home1
Prehistory-grass1-home2
Prehistory-grass1-home3
Prehistory-grass2-home1
Prehistory-grass2-home2
Prehistory-grass2-home3
Prehistory-grass3-home1
Prehistory-grass3-home2
Prehistory-grass3-home3


+1

In addition to the buildings, I also designed the characters and the cards used in the gameplay. For the characters, I focused on creating simple but distinctive designs so that players can quickly understand their role and identity during the game.


Caesar

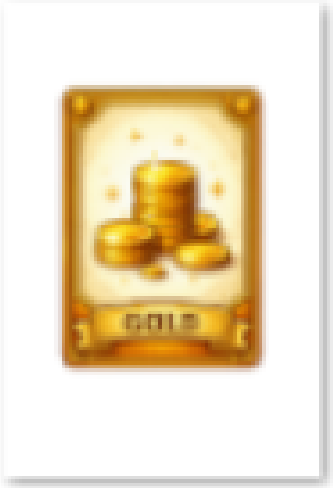

Cleopatra

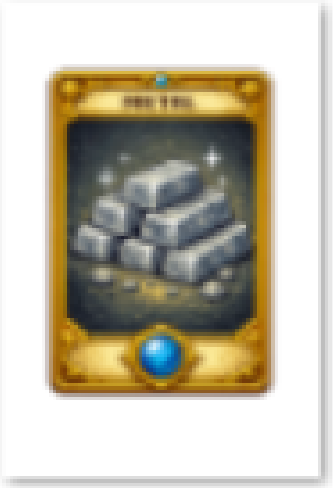

Elizabeth


Frederick the great

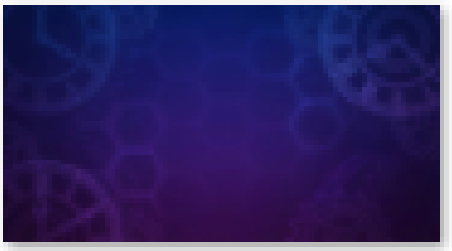

Napoleon

For the cards, I started working on both the visual layout and the illustrations so that they are easy to read while still being visually engaging, they are still not yet finished for now.


Card-gold


Card-metal

Another important part of my work was designing the user interface elements, especially the menu. I created the backgrounds for the menu pages, making sure the interface is clear, intuitive, and consistent with the overall visual style of the game.



Background-1

Background-2
(used for menu)

Overall, my role was to make sure the game has a cohesive visual style and that every graphical element, from buildings to cards to menus,contributes to the player’s immersion and makes the game more enjoyable and easier to navigate.

INDIVIDUAL PART RORY

My main contribution to the project focused on the implementation of the multiplayer system. The objective of this part of the project was to transform the game from a local experience into a shared game where several players can interact within the same match.

The first challenge was to design a system that would allow multiple players to play simultaneously while ensuring that the state of the game remains identical for everyone. In a strategy game like Timeless Empire, where players make decisions that directly affect the world and other players, maintaining synchronization between all participants is essential.

To solve this problem, I implemented a multiplayer architecture based on a client–server model. In this system, each player runs the game on their own computer, which acts as a client, while a central server is responsible for managing the communication between all connected players. When a player performs an action in the game, such as selecting a card or performing a specific gameplay action, this information is sent to the server.

The server then redistributes this information to all other connected players. Each client receives the action and applies it locally in the game. This mechanism ensures that all players observe the same events and that the game remains synchronized between every participant.

An important design decision was to transmit only the actions performed by players instead of sending the entire game state over the network. By sending only the necessary information, such as the type of action performed and the player responsible for it, the system reduces the amount of data exchanged and simplifies synchronization between clients.

Another aspect of this implementation was managing player connections. When a player connects to the server, they are assigned a unique identifier. This identifier allows the system to track which player performs each action and ensures that the game logic remains consistent.

The multiplayer system was designed to support up to four players in the same game session. This limit corresponds to the intended design of Timeless Empire, where each player manages their own civilization while interacting with the others through the different game mechanics.

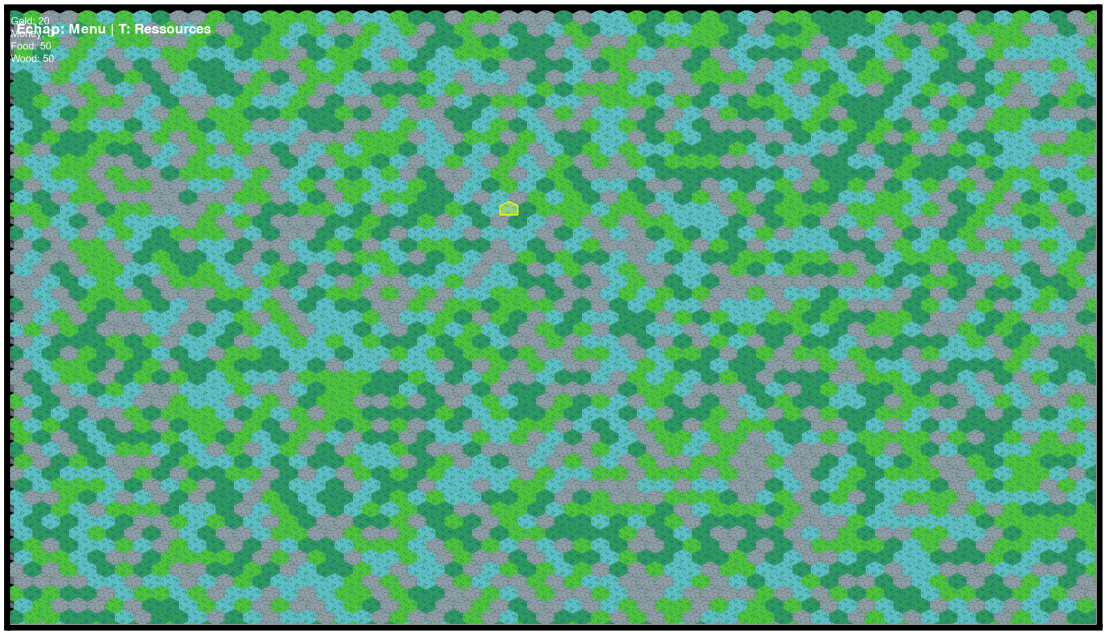
Integrating the multiplayer system into the existing project also required adapting some parts of the game structure. Certain gameplay elements had to be modified so that player actions could be transmitted through the network and correctly applied by all clients. This integration step was essential to ensure that the multiplayer functionality works smoothly with the other systems already developed by the team.

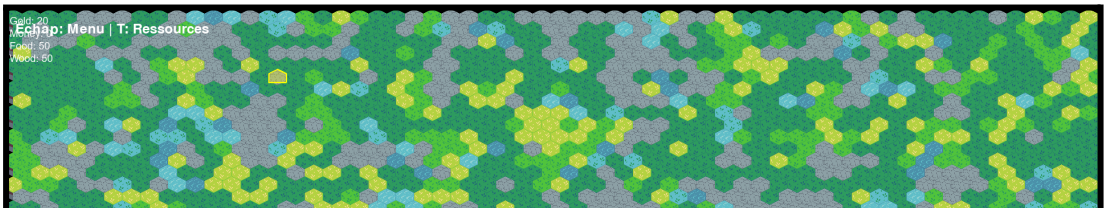
Working on this part of the project allowed me to explore new aspects of programming, particularly network communication and the challenges associated with synchronizing several running programs. It also required careful testing to verify that the different players always remain synchronized during a multiplayer session.

INDIVIDUAL PART MATHIS

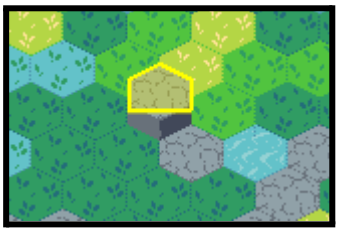
My part in this project was to supervise the advancement of the game itself, meaning looking toward the functionalities of the game. Therefore, I started by designing new buttons to navigate in the game and a new banner which is used as our title for the moment. Even though the designs doesn't match perfectly our game vibe, it's still a try to understand how to deal with them and where to place them.

Since last time I also had to deal with one of the main issues, if not the biggest, the map. This latter was very hard to achieve because it's not simple hexagons but rectangles containing hexagons and having them fit perfectly and in the good way took me a lot of time. Once I had managed to get a working map, I realized it kind of looked messy with random hexagons everywhere. After giving each hexagon probabilities to "expand", meaning that the tile which is at position l + 1 x c had more chance to be the same as the one before and same onto the column. I then managed to get a more "realistic" and smoother looking map which is a great base.





I also added a highlight effect onto the tiles to show the player which tile his cursor is on. This latter doesn't work properly, because as I've already said, the shape of the real image is not what we can see, it's in fact a hexagon in the middle of a rectangle. Moreover, I coded a way to lift an hexagon once the player has selected it. At this time selecting a tile doesn't do much but in short time selecting a tile will enable the player to set a structure or select it as a path to travel to.



Nevertheless, I've also included a timer of 40 minutes for each game so the players can keep track of the time left at the game.

Finally, I’ve also created a lobby where once the online functionality is set, players will be able to wait for each other before starting a new game. It will also allow the server to make sure everyone is connected properly before doing anything but I’m not part of this aspect.

What’s left to do is design a working turn by turn mode where each player will be able to interact with the environment now that the map is working. I will also need to work on the inclusion of the new constructions made by yiru and work on all the interactions that will be able to do such as collecting resources though through the map, fight with other players and include an ai which will be given choice each turn and will decide if she agrees an action or not. Even though it’s not fully operational and the overall needs to be smoothen, it’s still a massive advancement that I believe we are proud of.
