Project Report - Year Zero

Bearth Studio

May 17, 2019

timothee.ribes nicola.brankovic

enguerrand.vie axel.ribon

Contents

1 Introduction

1.1 Presentation of the Group

Bearth Studio is a video game development studio born on December 16, 2018. Composed of four members, its only claim is to carry out its first project, consisting of its first game: ***Year Zero.***
Coming from diverse and varied backgrounds, its members have di ff erent cultures and educations. Within ***Bearth Studio,*** we believe that our di ff erences contribute to a greater creative ambition and to a broader vision of our project as a whole, so that it is enriched by it. We will then be able to resolve the technical and artistic challenges that will arise during the development of ***Year Zero.*** Currently having only one project in progress, we remain focused on this one but it is always possible that in the future we will work together again on other projects so that ***Bearth Studio*** grows and continues to present quality games, witnesses of the common ambition that its members share.

1.2 Group name

Bearth Studio comes from the contraction of ***beyond Earth, "*** beyond the Earth "and ***birth,*** the birth". We then understand the meaning "birth, beyond the Earth". Closely linked to the name of his first game ***Year Zero, Bearth Studio***
then represents the birth or rebirth of humanity, beyond the known limits of the Earth. Wanting to name our group in relation to the name and theme of our first game, finding the name of the latter guided us to determine that of our group. This is why one only finds its full meaning with the other. The frame was then set for ***Year Zero,*** presenting a humanity that left Earth, seeking to live rather than survive. This departure is therefore year zero.

We made sure that the name of our group was taken from our first game so that it was he who defined the first codes of our studio as well as its peculiarities. We did not want, on the contrary, to find a name that only we could understand and that would not touch or mark the mind of anyone.

1.3 Bearth Studio and Year Zero logos

The logo of ***Bearth Studio*** is made up of three distinct parts. The first is the one we see at first glance: a stylized planet, surrounded by rings like the planet Saturn, for example. We then see that a rocket escapes from the orbit of this planet leaving behind

she a streak of color. The studio's name appears below in a digital, spatial and futuristic-looking font. The image we have of the logo, presenting our band, is simple and eye-catching so that the player remembers the studio logo. All of this having the characteristic of being refined and modern in order to show the main theme which is that of space, demonstrating the studio's continual propensity to be turned towards the future.

Regarding the logo of ***Year Zero,*** it is about a circle containing in its center the letter "Y" stylized in a slightly futuristic font which will remain that of our game. The circle then represents the "0" of ***Year Zero*** with the "Y". The colors used are in the same range as those of the logo of ***Bearth Studio.*** A rusty, metallic appearance has been given to the letter "Y" to match the idea of a future as a Metal Age. The slightly dirty appearance thus given can make one think of the old Earth left by its inhabitants in the history of the game.

Figure 1 - Bearth Studio logo

Figure 2 - Year Zero logo

1.4 Members of Bearth Studio

Bearth Studio is composed of four members from class C1, from the first year of the integrated preparatory cycle of the EPITA computer engineering school.

Group project manager, **Timothy Ribes** uses all his skills to create real cohesion within the group, to inspire and support his colleagues throughout the project. He is in charge of the various sound aspects of the game.

Brimming with inspiration, but also not neglecting the work, **Nicola Bran- kovic** takes care of the writing aspects underlying the game and brings it to life for good in that it will be placed in its context, thus laying down a framework, a story and a universe. It is in relation to the work of all the other members of the group that Nicola works.

Mastering some software necessary for the proper development of the graphic and visual aspect, **Enguerrand Vié** is responsible for the direction of the artistic aspect of the game. Having to ensure that each aspect of the game respects a sort of graphic charter, color codes, shapes, etc.

Having previously acquired some experience with Unity, **Axel Ribon** helps other members from a programming point of view or from a game engine management point of view. Also proficient in object oriented programming. It is, for these reasons mainly responsible for the main code of the game, which constitutes the body of the game.

2 General presentation

2.1 Presentation

2.1.1 Origin

Before choosing the content, or even the universe of our game, we had to determine the genre. Nowadays, many of the world famous games are ***FPS,*** of ***First Personal Shooter,*** or shooting games with a first person view, seeing on the screen what the character being played sees. After a quick review of our knowledge and capabilities in terms of development, we concluded that we would not do a ***FPS*** because requiring much more resources and time for the same rendering, compared to a game of a strategy genre. Each member of our group is passionate about strategy games

in real time : ***RTS, Real Time Strategy,*** the idea of making one ourselves quickly came to us and we all agreed.
As for the world of the game, noting the absence of a real strategy game taking place in space, we decided to try to fill this void and create our own. It would then have been easy to reproduce an already existing game in the same universe but we preferred to recreate one from scratch, taking various inspirations but especially with a view to making a game in our image, which contains the mechanics that we wanted, in other words, that we make our own game.
We wanted to integrate an alien race into our game, drawing inspiration from insects such as bees with their way of making their hives and of organizing themselves, or even ants. The whole being seen from a large scale. We also thought about making our base terrain a sphere so that by moving the camera in one direction, we would come back to the starting point after a while, but we abandoned this idea because it could lead to in players motion sickness, similar to motion sickness.

2.1.2 Nature

Year Zero is a space strategy game in which several teams compete for supremacy in the universe. It controls buildings that produce units on a map with fixed boundaries.
The chosen game engine is Unity and the language used is CSharp (C). The art direction is fanciful and unrealistic in order to give a dark but not shocking tone. The objective is to make the game all public. The cardboard aspect was preferred to realism in order to make the atmosphere of the game quite light but without losing the strategy aspect. However, we wanted to keep a certain degree of realism in order to keep a dramatic side in view of the history of the game. From a sound point of view the game is quite epic with great catchy themes. But he juggles with calmer and more melancholy themes to recall the desperate condition of humanity. The epic aspect being purely fun and the dramatic aspect serving to reinforce the intentions of the scenario.

2.1.3 Purpose and interests

A Year Zero session lasts between 15 and 30 minutes on average. The game combines pure strategy as well as dynamism in order to create a real notion of progression during the game. The time required for a game may seem long, but it is necessary to give players time to strategize

Figure 3 - Cartoon but serious aspect
Endless Space 2

and install its base.
We also believe that speeding up the games would certainly make the game more accessible but also less demanding. We wanted to create a game that was easy to understand but difficult to master so that the player would derive real satisfaction from defeating his enemies.

2.1.4 Year Zero in the history of video games

Year Zero is a real time strategy game. The genre has already been around for almost 40 years with the game ***War of Nerves !.*** But this one has democratized and renewed to find its current form with ***Warcraft*** released in 1994. Other saga of ***RTS*** famous were born as ***Age of Empires*** whose particularity is to be able to advance in the periods of history as the game progresses. We can still cite ***Age of Mythology*** taking place as its name suggests in a mythological context. More recently ***Battle for Middle Earth*** taking place in the universe of ***Lord of the Rings.*** But in the history of ***RTS*** we especially remember the saga ***StartCraft*** for its competitive aspect and ***Warcraft iii*** for having added the notion of "Hero" leading a few years later to the creation of a new very popular genre that is the ***MOBA, Multiplayer Online Battle Arena.***

2.2 Functional

2.2.1 Contextual reminder

Humanity has exhausted all the resources of planet Earth, abandoning it in its pollution. It was the scene of many wars leading to the virtual extinction of humanity. In a last hope, the remaining civilizations set out to conquer the universe to develop again. The starting point for the conquest of the universe and the rebirth of mankind is called
Year Zero.
During the main campaign the player will have to fight various battles with alien civilizations to become more and more powerful and to assert his supremacy. Its battles may resemble the classic game mode with the management of its base. The campaign can also be an adventure where the player controls a regiment of troops but without buildings the goal being to survive until the end of the mission. Still other missions will require the player to resist for a while in the face of waves of attacking enemies.

2.2.2 Game features

The goal of the player is to develop his civilization in order to destroy the enemy team (s). For this he will be able to collect resources in order to construct and improve his buildings, and thus form an army. The game will be broken down into two game modes. Single player mode and multiplayer mode.

Single player mode:

Countryside - The campaign is divided into x missions which follow one another. Once the player has completed a mission, he can move on to the next and so on until he reaches the end of the game.
The di di erent missions of the game offer various objectives in order to break up any possible monotony:

- Classic missions require the player to develop his base as well as his army to destroy the opponent.

- The survival missions are identical to the classic missions except that it is not necessary to destroy the enemy but only waves of troops in a given time. The player is cloistered around his base and cannot attack enemy bases.

- Adventure missions provide a regiment of troops at the start of the game to the player who will have to use his strategy to cover the entire level while keeping at least one unit alive.

Quick Game - This mode corresponds to the multiplayer mode but offline. The only di ff erence being that only artificial intelligences can be faced here (See multiplayer).

Multiplayer mode:
The heart of the game, this mode allows players to try out their strategies against each other. The only game mode present is the classic mode of the campaign mode with the difference that the artificial intelligences can be replaced by real players or not.

2.2.3 Progress of a game

We will describe a multiplayer game. A player must create the game from the game menu. He must decide on the game card, the name of the game as well as the maximum number of players. Players wishing to join the game must enter its name in the appropriate menu to enter the "waiting room".

In this waiting room each player decides on the color associated with him, the race of his civilization as well as his team. Then they must check the box " ***Ready*** **"**to allow the player who created the session to launch the game. The players are then distributed randomly on the map.
Each player then has a space station as their main building. This allows you to produce builders who build and repair buildings, but also undermine the player's resources. The station can be improved to become more resistant but also and above all to be able to unlock new buildings to construct.

The player must also develop his improvement tree which will allow him, via a resource cost to unlock new units, to improve them as well as to unlock various production improvements on his buildings.
Manufacturers can therefore mine various resources, particularly on asteroids present on the map. These resources are necessary for the construction of buildings and units, but they are not inexhaustible. By dint of being mined, the asteroids will destroy themselves forcing the player to find others. The player just has to select a constructor and assign it an asteroid so that it makes round trips to the nearest space station so that the resource of this asteroid is stored.

There is another resource that is food, not found in space, the player must build space farms that will raise cattle to produce meat at regular time intervals.

Finally, the player must pay attention to the population. Indeed, each unit has a population value, and the sum of the population values of all the units cannot exceed the maximum population of the base.
The maximum population of the base will depend on the number of accommodation buildings present. With a ceiling of 100 which cannot be exceeded. Other buildings are constructible and in particular to produce di ff erent types of troops.
Each of its buildings is given a rally banner, so each unit produced will travel to that rally point. As for defense, the player has at his disposal the possibility of building defense turrets which will automatically attack enemy troops nearby. The player can also place sentries that will alert him to the presence of enemy troops. Because in fact, being in space, the player does not have the possibility of building ramparts, so he will not be able to take refuge behind but must prepare his defense in order to survive. Hence the relevance of properly placing your sentries to give it maximum time before the imminent attack. Turrets can slow the enemy down but rarely defeat their entire army.

Buildings are not equipped with thrusters and therefore cannot move around on their own. They are therefore in orbit around the space station.

2.3 Technical and Methodological Aspects

2.3.1 Material resources

We used a variety of materials to develop our game. So Visual Studio was used for the code part, Blender for the 3D modeling, Photoshop for the design and Git for the " ***versioning "*** of the project. Everything was put together to build the game with Unity. We have 4 laptops at our disposal as well as the EPITA premises. For our research we used the Google search engine. EPITA provides us with editions of Windows 10 as well as an O ffi ce 365 license which we use to organize ourselves schematically on the construction of the game.

2.4 Intellectual Means

To help us during development we have access to ***MSDN*** for everything concerning the C language. Unity provides documentation for its use and we also used a lot of forum topics already created and solved. These forums are mainly from Unity sites and

StackOver fl ow. These resources allow us to benefit from the experience of others to learn faster. We were also able to use explanatory videos on certain aspects of Unity that had not been mastered or on other software that we had to use.

2.5 Economic Means

The game is not monetized in any way. Everything is free with no additional paid downloadable content or subscription or " ***lootbox ".***
We believe that the success of the game alone would be a reward and if so, an advertising showcase highlighting the studio.
The base game is not paid for the simple reason that it is the result of a school project and we do not believe that this type of project should be paid.

3 Project breakdown

3.1 Task progress schedule

Support task	**Defense 1**	**Defense 2**	**Defense 3**
Graphics	30	60	100
His	20	70	100
Menu	60	80	100
Network	50	70	100
AI	10	60	100
Game mechanics	60	70	100

3.2 Table of distribution of tasks

	Timothy	Nicola	Enguerrand	Axel
Graphics				
Troops			X	
Buildings			X	
Menus			X	
Interface			X	
Trailer			X	
His				
Sound effects	X			
Music	X			
Interfaces				
Main Menu				X
In-game menu				X
User interface				X
Controls				X
Network				
Lobby				X
Online				X
AI				
Bots				X
Automatisms				X
Game mechanics				
Selection				X
Resources				X
Combat				X
Construction / Production				X
Content				
Map			X	
Countryside	X	X		
History		X		
Other				
Discord			X	
Site		X	X	

3.3 Development on distributed tasks

3.3.1 Timothy

Not presenting any particular quality, I devote myself to the role of project manager aiming to help my colleagues as best as possible thus facilitating mutual aid, communication and cohesion which are fundamental principles for a group project. However, I play the guitar and will therefore use my musical abilities to take care of everything that has to do with the sound in the game, that is to say to create an atmosphere specific to the game. ***Year Zero*** and touching for the player with adapted music, with a futuristic tone but also with the tension that this kind of game creates throughout a game. As part of learning C # at EPITA, being able to apply it in a concrete context of this game in subjects allows me to deepen it and discover more complex subjects. In fact, with Nicola and Axel, we are going to have to code sufficiently powerful artificial intelligences so that they can give difficulty to the player alone without exceeding him too much. Finally, once the game is functional in its multiplayer part, with Nicola we will have to script parts to make it a solo adventure that would seem minimalist compared to professional editions of this genre, but will still allow us to express the own universe. ***Year Zero***
developed by the e ff orts of the whole group.

3.3.2 Nicola

Being a big consumer of real-time strategy games (***RTS)*** I dedicated myself to game mechanics and AI to make our game fun to play and balanced. A ***RTS*** Often requires performing multiple actions at once, allowing players to order buildings or units to perform multiple tasks one after the other, quickly, is a key part of the gameplay of a strategy game. Thinking about how the game's mechanics work and transcribing them into an algorithm and then into C # is enriching and pushes our team to constant mutual aid and to debates on the limits of this automation between Axel, Timothée and me. Also taking care of the campaign and the history of our game, I will have to work hand in hand with Enguerrand and Timothée in order to provide a certain coherence with the musics, the themes and the designs. Our goal is to give a soul and a unique atmosphere to our game in order to provide a unique experience for players,

3.3.3 Enguerrand

Possessing above all qualities of graphic work, my work in the project ***Year Zero*** will be particularly focused on the graphic aspect of the game. I am in charge of all the 3D modeling with regard to the 3D models of the units and those of the buildings. Helped by Axel, I will also have to take care of dressing the game interfaces including menus, buttons, interface in a part etc. The whole in order to create a futuristic, spatial atmosphere, with obviously the colors which relate to it. The same work will be done in parallel on the menus of " ***lobby"*** multiplayer. I will also be in charge of a programming point of view, of the whole combat system of our game, helped this time by Nicola. It will then be a question of making the units interact with each other, so that the combat is alive and dynamic. One aspect that combines programming and graphics is the dynamic generation of environments, of the game map. Wanting to integrate a system of random generation of maps so that the player never tires of replaying a game of ***Year Zero,*** I will also be responsible for developing such a system.

3.3.4 Axel

My work in the team has been above all algorithmic. I took care of certain mechanics that my colleagues relied on to do their work. So I was responsible for providing them with their own code and being able to explain it to them. I think that communication is essential to be e ffi cient and productive and my work has illustrated this need in the sense that it guarantees that we exchange on our work. In addition, my work also included the interface with the user and therefore invited me to put myself in his shoes and ask their opinion of those around me. Although very abstract at first glance, my tasks demanded both technical and social qualities.

So I was in charge of the unit selection algorithm which is the pillar of the game in the sense that any added content requires unit selection to be able to be tested.
I was also responsible for the development of the menu code. This task may seem simple at first glance but the goal of a developer is not only to make the game playable but also to facilitate the work of others by coding a very adaptive and easy to use system. Everything had to be developed so that once the algorithm is finished there is no longer any need to touch the code but only the interface and in the simplest and fastest possible way. I was also in charge of what we called the automatisms. These are light

notions of artificial intelligence in units. We can for example cite the patrol system, the reactions of units when enemies approach or their ability to mine and bring ores back to the space station. The notion of intelligence being too weak to be called artificial intelligence and the fact that these reactions are totally dissociated from the actions of the player, we preferred to dissociate these two parts.
However, I was also in charge of the artificial intelligence of the players controlled by the computer. This was my most difficult task because I had no experience in this area and the AI in a strategy game is much more complex than in a majority of other types of games.
I also developed everything related to the production of units and buildings, ranging from the display of the cost of these to the production through a mode of placement of building in the base, as well as the operation of the associated interface. This is a complex task because all these systems fit together and therefore modi fi cation of one of these systems affects the others.

Another part of my job has been to implement multiplayer. And all that it implies to know the menus to create or join a server as well as the waiting room where the players can choose their race and their team. But multiplayer has mainly been implemented throughout the game and in the management of the game. It is also a massive work in terms of optimization to avoid possible latencies during the game.
To sum up, I took care of the multiplayer, the unit selection, the interface from a code point of view, the construction of its base, the management of the units as well as their production and their capacities.
Finally I took care of the combat system.
I am above all a coder and not an artist, which is why this aspect of the creation of the game is not attributed to me.

4 Development on distributed tasks

4.1 Timothy

My work as a project manager is between the di ff erent areas that we use, while being focused on the sound aspects but also the part with one player.

4.1.1 Tutorial

The tutorial consists of a scripted part in which the player must perform the actions requested to move forward while understanding and learning the mechanics of the game. The first difficulty that arises in this kind of exercise is to put oneself in the player's shoes and for that to forget as much as possible his knowledge and habits in relation to the game on which we play, or even in relation to the whole genre. Likewise, we have to get him to understand the actions he has to perform and for that we have decided to display messages and pause the game while the player reads and can move forward at his own pace with his mouse. We have chosen to make a simple tutorial which explains only the concise basics of the game in order to leave some experience of discovery and to do something very light that can be replayed very quickly if necessary to remember the main workings of units and buildings. . On the technical side this consisted mainly of using the tools set up for the operation of the game and how to link them together by a script that executes each step to be performed. This requires communication with Axel for the understanding and sometimes the adaptation of the code, but also the graphic adjustments of Enguerrand which allow the game to always remain aesthetically coherent. It is therefore our ability to work together on the same elements that is brought into play at this time.

4.1.2 Website

For my part, I have never taken part in the design of a website, so I particularly worked on the aspect of rendering and functionalities with Nicola who already had skills in this area.

4.1.3 Dubbing

The set of music and sounds that will be an integral part of the game, especially during the games and the various interactions of the player and the actions of the units themselves, must be developed for the next support at the same time as the internal graphics of the game. In addition, units will have voices that respond to orders. For example, during an attack order given to a vessel, we will hear "attack". In addition, the texts and dialogues of the campaign will be dubbed to really create a story with a narration corresponding to the scenario of Year Zero.

4.1.4 Mission

The mission is part of the single player experience offered by Year Zero and follows on from the tutorial that must be completed to access it, and will present the most attractive mechanics in a real-time strategy game, combat whose operation and implementation are detailed in the work of Axel. The enemy therefore does not yet have any economic or production buildings, and the player can concentrate on the simple fact of getting rid of the units which are going to attack his base by following the instructions introduced by the narrator. On the other hand, an adjustment work will have to be done with the rebalances of the game so that this part remains fairly simple to perform for the novice player. Finally, this mission also aims to introduce the scenario by making the first contact between him and the antagonist who sends troops to destroy him. In addition, another endless mode has been added, it is a mode that we had planned to add and which is similar to this mission in which the player will therefore once again face waves that do not will not rest and will repeat themselves endlessly, always being stronger to constitute a challenge to reach the most advanced waves.

4.1.5 Sounds

At the same time, we had to continue to develop the sound atmosphere of the game, already for the voices of the characters who arrive as the story progresses, which forced us to do audio editing using the Audacity software. and for which we have chosen to use our voices despite our lack of experience in the field of dubbing. We have realized the diversity of professions involved in a game. We therefore hope to be able to be serious in this area but still leaving our trace through our voices as modified as they are. We therefore take care to prepare the replicas with a certain idea of the tone they should have, then we carry out tests until we have a relevant version from which we then remove the noise and to which we add height e ects to stick to the character who must be embodied. Too, I made sure that the musical aspect is functional in the game, that is to say that we can launch the musics according to the moment, that they do not overlap, nor restart or cut themselves according to the environment changes like joining a game or changing menu. In addition, so that the music in this game is an integral part of the game experience I thought of, better than making a list of random sounds that play as you go through a game, combine their atmosphere with state of the game. In fact, rather quiet music will be played at the start of the game, then as the player unlocks nor do they restart or shut down depending on changes in the environment such as joining a game or changing menu. In addition, so that the music in this game is an integral part of the game experience I thought of, better than making a list of random sounds that play as you go through a game, combine their atmosphere with state of the game. In fact, rather quiet music will be played at the start of the game, then as the player unlocks nor do they restart or shut down depending on changes in the environment such as joining a game or changing menu. In addition, so that the music in this game is an integral part of the game experience I thought of, better than making a list of random sounds that play as you go through a game, combine their atmosphere with state of the game. In fact, rather quiet music will be played at the start of the game, then as the player unlocks

units or technologies, the sounds will be more and more energetic with the aim of arriving at an epic framework for the great endgame battles which will involve many units.

4.1.6 Balancing

The balance of a strategy game, even in its less advanced version, is a main concept, so that the various units form a whole. It allows each ship in our game to have its place and its usefulness either by having advantageous statistics for a higher cost, or lower but with more powerful spells. Achieving a perfect balance would require data analysis on a huge number of games and players, which is why we have relied on personal group practice sessions which will have to be pushed further for the next defense in order to continue to improve balance and a good, interesting experience. For this we have created a table gathering all the units to which we have assigned the various statistics: attack speed, damage, range, health points, speed and their cost in the three resources of the game. To this we have added coe ffi cients to each of its values and grouped them into a value specific to the unit. Our goal then was for all units to have the same value while modulating their di ff erent characteristics, but we ended up with something where the values tended to increase with the end of the game because we had neglected the research to be done through the game. 'skill tree necessary to obtain these units which will be balanced for the next defense with the project of having a complete playable and pleasant course of the game.

4.2 Nicola

4.2.1 Website

Not being specialized in the creation of Internet sites, I decided to use the Wix tool in order to provide a clean result. I wanted to keep the theme of the game in the website, so I reused one of the dominant fonts in our game and the menu background image. I then followed the plan proposed in the Project File, and therefore incorporated the download links of the Specifications, the defense report and the "lite" version of the game, and a complete presentation of members of Bearth Studio. However, my capacities as a graphic designer and my good taste being only subjective and limited, I then gave way to Enguerrand who took charge of finalizing the website.

4.2.2 Tutorial

The tutorial aims to present the basic mechanics of the game, it sets the atmosphere of the game, and gives the main keys to victory for an RTS. During this, the player will be presented how to select the units / buildings but also the base buildings and their utilities, the use of the Builder which is the construction / harvest unit, the combat units and how to create them, how to collect resources, how to use them and finally the functioning of the SkillTree, original element at the center of the gameplay of Year Zero. In order to force the player to listen to the instructions of the tutorial, we had to limit the actions that this one can carry out, and to verify that the requests of the tutorial are satisfied before continuing. The tutorial, being one of the first experiences of the game that we have been able to have,

4.2.3 Single player mode

The bases of the single player mode having been implemented for the defense two, we decided, with Timothé to provide an outcome to the Scenario of Year Zero. This time, the mission is intended to be longer and more complete and will once again be doubled. Allowing Year Zero to have a complete scenario is an essential brick, among all those which allow it to be at the level of current video game standards. In the new missions implanted, this time, the player will have to fight against the artificial intelligence of Year Zero, and will aim to destroy all the buildings of the enemy. Everything is scripted and o ff ers aids and indications to the player in order to make him beat our very competent AI.

The single player mode in general has also been revised, especially in terms of balancing, so that the difficulty follows the player's progression curve.

4.2.4 Scenario

Following consultations with the group, the Year Zero Scenario, presented at the first defense, has been modified. This one can be found in the promotional video for the game, but here's a synopsis:

After the slow and inevitable destruction of the earth by man, he was forced to orient himself towards the heavens for his survival. Unfortunately space is far from uninhabited, and humanity is forced to meet many challenges. Among them, an alien race having conquered many systems, extremely aggressive, it can adapt to all environments, but the most irritating? She can take possession of the bodies of your friends, your brothers in arms,

your leaders, your family. . . The fights are almost permanent and the man has nowhere to settle. Humanity is near extinction, and desperately awaits its messiah.

4.3 Enguerrand

4.3.1 Creation of logos

Responsible for the visual part of the game, I looked first, when writing the specifications, on the design of a logo for the game, as well as for the group. A sort of video game development and publishing studio. Already presented when rendering the specifications, the logos of Year Zero and ***Bearth Studio*** predominantly contain the colors red, blue and purple in various shades. Wanting to give a spatial aspect to relate to the main theme of the game, the logos became a benchmark throughout future design as far as game design is concerned.

4.3.2 Graphic identity design

Apart from the logos, which are only a less visible part than the interface of the game for example, it was necessary to determine the guidelines of the artistic direction of Year Zero in order to keep the same colors, shapes and so on. All in the pretension of wanting to create a catchy and immersive atmosphere. It is therefore in warm color tones, approaching red, purple and blue that Year Zero will take shape. The font that we will mostly use is ***Orbitron Black*** for most of the texts and for some titles we will use Space Age.

Figure 4 - Police Orbitron Black

Figure 5 - Police Space Age

4.3.3 Main menu

It was towards the main menu that I leaned first. To the already existing menu, made up of buttons without texture, and the default Unity background, I added a background which is an image in the space where we see nebulae. I added the logos of Year Zero and Bearth Studio at the bottom left and right respectively. Finally I created in Photoshop the title logo Year Zero that I imported into Unity in order to put it above the buttons.

Not having a precise idea of how I was going to design the buttons, I tried a few versions with various colors, still creating a button image in Photoshop that would overlap the Unity button.

Finally, I created a new texture for the buttons that I imported. It turned out that it stuck well to the bottom and to the rest, so we kept it.

Figure 6 - Button texture

I declined this same texture in several versions in order to integrate as well as possible in all the scenes of our Unity project and in all the situations where we were going to need it.

Figure 7 - Button for a fill bar

Figure 8 - Square button with rounded edges

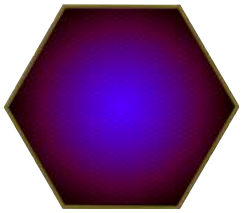


Figure 9 - Hexagon shaped button

Figure 10 - Rectangle button with rounded edges

Finally, it's towards hexagon-shaped buttons that I'm leaning, wanting to innovate a bit so that we don't have buttons that seem to be Unity's only by default. Instead of marking where these buttons redirect, I thought to myself that it was probably more self-explanatory and also more visually pleasing to put icons instead of text. We then had a menu that was starting to take shape.

I also added for an e ff ect of immersion within the menu, two layers of stars materialized by two images containing transparency and stars. I added an animation to them so that they turn on themselves without stopping. I also created via a small script, a parallax effect (which we will find later). Parallax is a way of rendering depth and distance. Here, it is a question of making follow the movements of the mouse in the main menu to the background image but with a slight coe ffi cient. The effect is very light

Figure 11 - Main menu

here it is only visual.

4.3.4 Adding graphic resources

At the end of the first defense, I said to myself that it was necessary to delete everything that, until then, was ***by default.*** It was in this process that I had the idea of designing a cursor that would replace the one that ***Windows*** possesses.
It was then necessary that this cursor can be integrated into the graphic charter of the game without losing visibility. After several inconclusive tests, often because the colors chosen made it less visible, I finally arrived at the current version which is integrated into the game.

4.3.5 Loading screen

Besides the main menu scene, there is a scene that is rarely visible since it happens very quickly. This is the loading screen scene. It is simple in that it only consists of the same background as the main menu, as well as a progress bar and a message that displays.
LOADING ...

4.3.6 Button animations

Adding animations to the buttons was necessary to create dynamism in the navigation within the menus. The animation is the same for all the buttons of the main menu and its submenus. It consists of the enlargement of the button. So when the mouse passes over the button, the latter sees its size increase slightly.

4.3.7 Game screen

Global environment

When you arrive in a game, you are in space. It is necessary to have a suitable environment. Thus the ground from which the units move is encompassed in a ***Skybox*** which corresponds to a sphere whose texture is an image, here of space. The rendering makes sure to respect a kind of immersion in this sphere and gives the impression that the depth is infinite.

E ff and parallax

When you move the camera around the game, you only get the impression of movement if you see units rolling by. This is why I added more than four layers of stars below the ground, all at di de erent heights so that when moving, the stars more or less follow the movement depending on how far they are from the ground. ground or not.

Inteface

Most of my work until this defense has been the design of the interface in a part. Everything was done in Photoshop and then imported into Unity. I realized in an image which will constitute the overall interface in the game.

Figure 12 - Interface image

In the same way as for the main menu, the action buttons of the units have, instead of a text, an icon, as well as the texture of a square button (***cf: Figure 7).***

Figure 13 - In-game interface

There are also navigation buttons above the screen. The texture associated with them is the rectangular button with rounded edges. The text has been left for a better understanding of their action and the color has been put on a golden, yellow which is doing well to the rest of the interface.
To their right, there is a button for the skill tree, with its associated icon. Also to its right, we find the resource indicators. Originally, they are buttons but we cannot interact with them, hence their darker color than the others. The color of the text and numbers has also been set to gold, yellow.

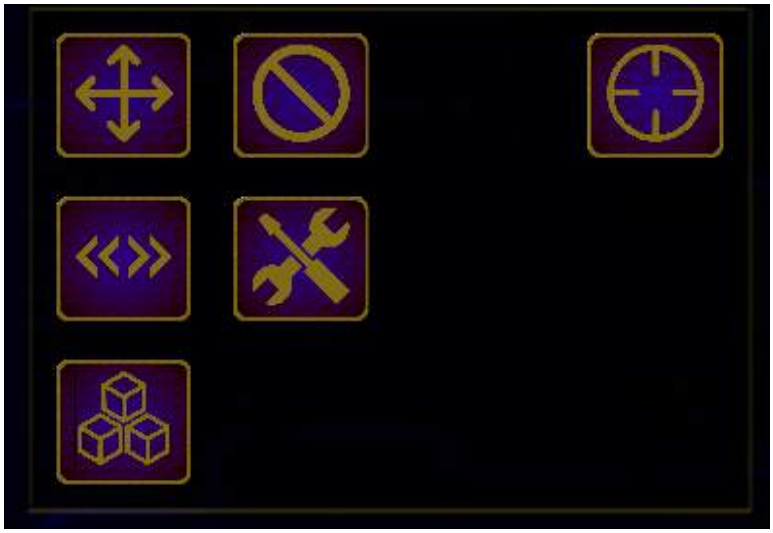


Figure 14 - Buttons with icons

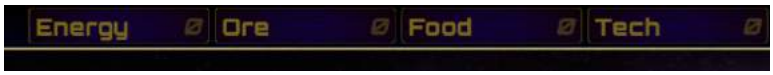


Figure 15 - Resource buttons

There are also the minimap buttons, also formed by their texture and an icon, like all the others made by me in Photoshop. We also find the inactive constructor button made in the same way. All of these buttons have the same animation, which changes their color depending on their state. The four distinct states being: ***Highlighted*** if the mouse is on the button, ***Pressed*** if the mouse clicked on the button, ***Disabled***
if we can not interact with the button as well as the default state.

Figure 16 - Button hovered over by the mouse

Figure 17 - Button clicked

4.3.8 Improving visibility

A game in which the dominant environment is space, like Year Zero, often has dark colors. Texts should be presented in bright colors so that they are clearly visible to the person in front of the screen. So most of the messages will have colors like red or yellow-gold. For some they could be even more visible and it is with this in mind that I applied a black gradient background to them so that they are even more striking among everything that is displayed on the screen.
Thus the tutorial messages explaining how to play will be a little more visible as well as the help messages on the action buttons of the units, which did not include any help message before.

In addition, some menus have been improved with some graphic details to make them more dynamic. This is the login screen, and the ***lobby,***
the waiting room before the game starts in multiplayer.
I also applied Year Zero's graphic designs to the menu that can be found in-game, during a game. You can access it by pressing the shortcut written

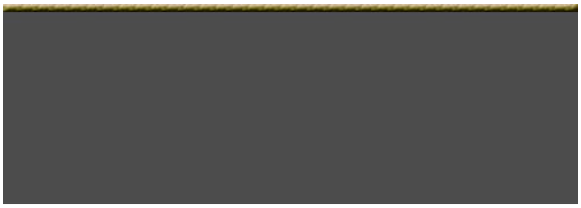


Figure 18 - Background banner for texts

Figure 19 - Help for unit actions

on the navigation buttons at the top of the screen or by pressing the key ***escape.***

Figure 20 - Main menu in a game

4.3.9 3D Models

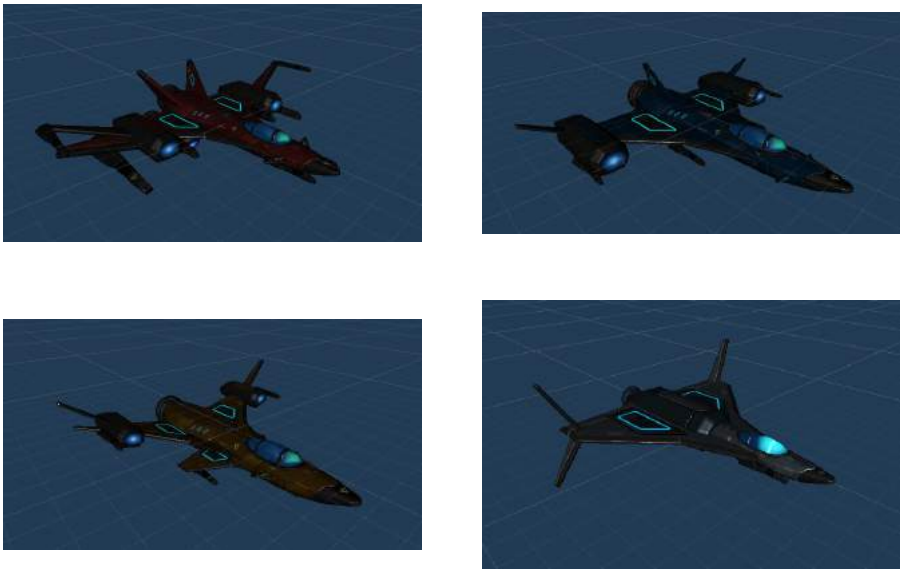
4.3.10 The units

Year Zero, like any good strategy game, o ers many units. They must then be dressed so that they can be differentiated. After a lot of research for 3D models that could match our needs, I found an assortment of resources that will be able to constitute a major part of the models associated with the units, for those who will play humans at least. Thus we find no less than fifteen di ff erent vessels, each of which can be declined in several colors, which will eventually mark the di ff erent teams with the color chosen in the game waiting room.

Some of the ships present will not be part of the game, we have not fi nalized them all yet, however the variety of different colors and texture types available to us give us many possibilities for the game in fi nal version to provide models in line with the Year Zero graphic charter.

Here you can see up-close images of some of Year Zero's ships.

Figure 21 - Year Zero ships in multiple colors



An important aspect of ship models is that they are modular. That is to say that they can be broken down into several constituent parts. Thus it will be easier for us to materialize a destruction or even an animation of construction of these vessels.

Besides these models corresponding to the units themselves, there is a whole aspect of the units in 3D, which is often forgotten, but we cannot miss it. This is the set of projectiles that will be fired by the units. It goes from

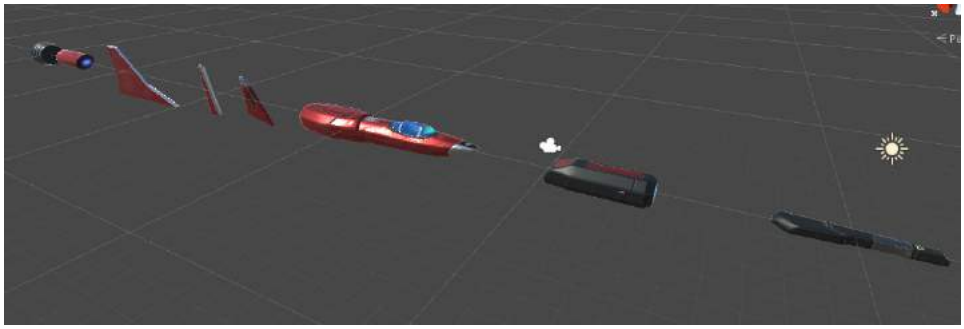
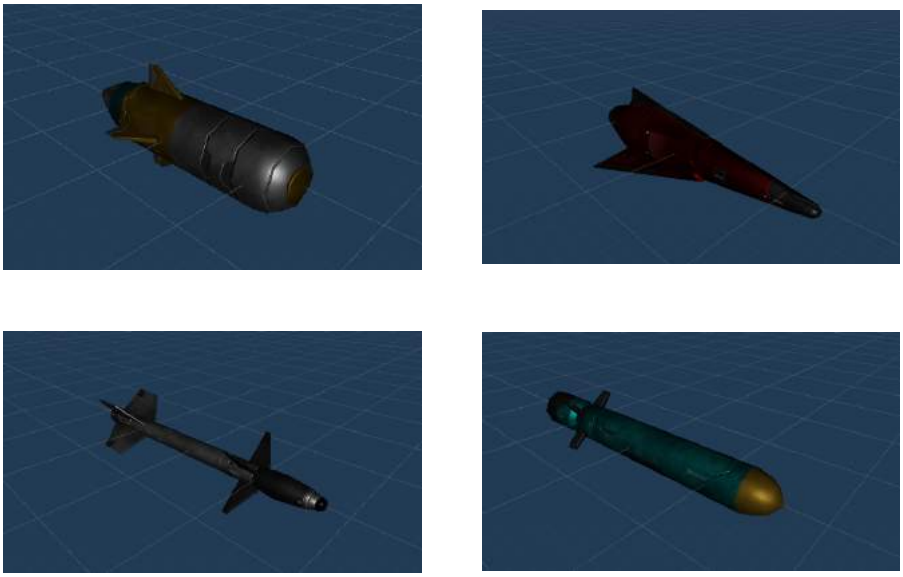


Figure 22 - Modularity of a vessel

simple laser to missiles, to bombs and the like. Besides the laser which is not yet in the game, since it consists of a particle animation and the latter has not yet been done, it is not found in this version of the game. Likewise for the other projectiles since they will be associated with particle e ects, missile reactors, explosion e ects etc. All this will have to be present in the version of the game that we will return for the third and final defense which will mark the end of the project. We are obligated to provide a complete set for this event.

Below are some examples of the models of projectiles of all types that will join the 3D models of Year Zero.



4.3.11 Buildings

In Year Zero there are no less than ten different buildings. Most of them are of the same type, fixed and relatively similar, and a few other unique ones such as the radar or the defense turret. Giving models to these buildings is not a simple task since they must re fl ect the atmosphere of the game, not shatter the unity of a spatial environment that we find in all aspects of what the game tells us. shows on the screen. So, I added 3D resources corresponding to modules of a futuristic if not spatial type which, put together should be able to give us enough possibilities for our many buildings, so that each one has a unique model, standing out from others. Most are not yet assigned as I am still looking for models that might be better suited.

4.4 Additional models

A spatial environment is intended to be empty, but it appeared that filling this void at least with some additional graphic elements would be a good idea and would make the environment even more engaging. This is why I added some asteroids which will take place in the map of the

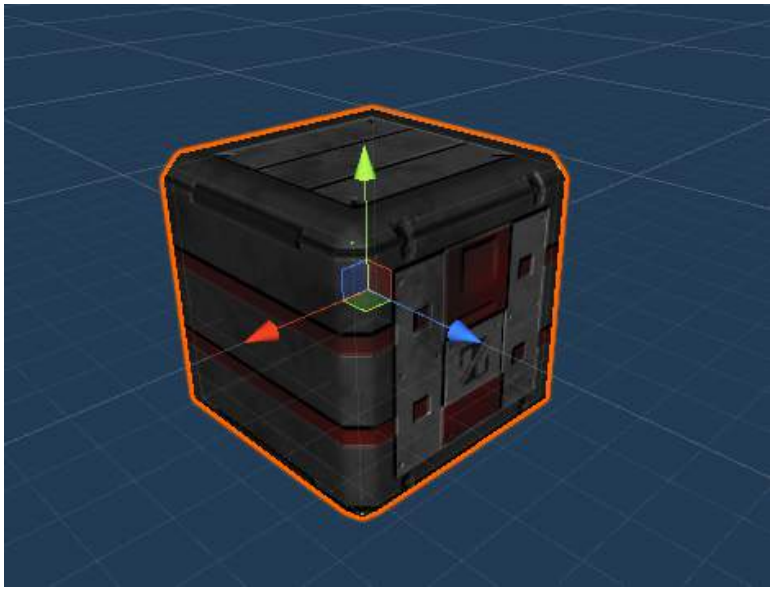


Figure 23 - Example of a modular model

game, in a game. They are only purely decorative, one cannot interact with them, in particular to collect resources. They are placed higher or lower depending on the reference level, which is that of the units and buildings, so that the spatial environment e ect is kept to a minimum and improves the player's immersion. Asteroids come in several sizes and have different textures that we will apply to them in order to choose the appearance they have according to the type of space environment of the game. For example, we can apply the ice texture to asteroids during a game with an ice space as the chosen card. This is one of the features that I plan to add to the fi nal set, presented during the third defense.

Figure 24 - Small size

Figure 25 - Medium size

Figure 26 - Several asteroids

Always in the concern of improving the immersion and the 3D rendering of the game, I also added 3D resources of space debris. There are many various debris, buildings or even vessels. Depending on the map, we can find many of this debris as an element of scenery in the same way as asteroids.

It is important to also note that some of this debris will be used when a unit or building is destroyed, then we will display the corresponding 3D debris model.

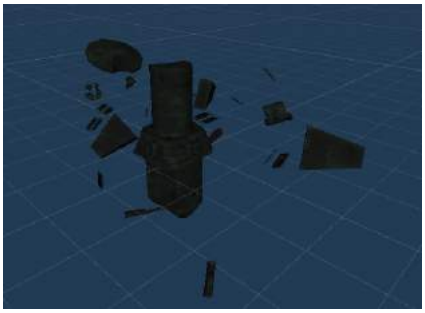


Figure 27 - Debris model

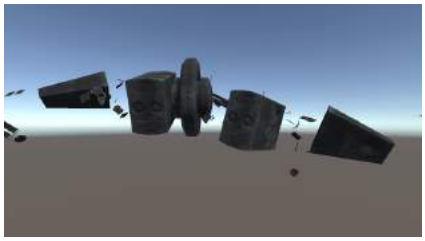


Figure 28 - Other debris

4.4.1 Website

I also contributed to the Year Zero website, particularly on the addition of content within the various sections present.

4.5 Discord Download

Regularly using Discord software for the general organization of the group as well as for vocal meetings with the members of the group, I chose to add the integration of year Zero within Discord. That is, when you launch the game, Discord displays the profile of the person playing the game, that it is playing it. It is a purely cosmetic and non-essential addition, but it contributes to the image we want to give to the game, testifying to a certain professionalism that we are trying to achieve.

4.6 Axel

Here I will present each feature I added to the game, how I felt, the issues that occurred.
First of all in order to structure my part correctly, I will not present my work in chronological order but rather by section. The di ff erent systems being intertwined with each other, I was not able to develop them all one after the other but by building everything little by little. However, I will keep the order as chronological as possible.

4.6.1 Controls

Selection

Beyond creating the project on Unity, my first task was to develop a unit selection system. The principle is a simple one, a left click on a unit selects it, but to this is added the selection of several units which occurs when we perform a left click on the mouse and then move it to form a rectangle used to select all the units in this rectangle. I then gave some speci fi cities to this algorithm, for example, if we select a building we can only select one, if we double-left click we select all the units on the screen which are of the same type as the one under the mouse pointer, or again, if we draw a rectangle around units that do not belong to us, we will fi nally select only one. Too, right clicking on a unit will perform the appropriate interaction. This algorithm is one of the most complex that I have been given to develop within the framework of this project as it takes into account parameters, I had to think it over and rethink it continuously as I added to it. features.

My first di ffi culty was how I could select the unit under the mouse pointer. After some research, I discovered the principle of Raycast, this Unity functionality allows to send a ray from a point and following a defined direction vector and returns the object encountered by the ray. So I send a ray from the position of the mouse and directed to the point just below the mouse to access the unit and thus select it. The multi-selection system is simpler but also much more demanding, it will go through the list of units and transpose their 3D coordinate into 2D then select them if they are in the drawn rectangle.

Player control system

As I added functionalities to the game, it appeared to me that a system had to be created to manage di ff erent control pro fi les, for example when we want to place a marker on the map we press the button associated and we then go to a new very simpli fi ed key pro fi le in which a left click on the minimap will place the marker and a click outside the minimap or else pressing the ESC key will cancel the marker mode to return to the controls classic. Without this system it would be complicated to isolate certain functionalities and for example opening the pause menu would not prevent the selection of units or the movement of the camera, which poses a problem.

System training

This algorithm was not one of our priorities, but its usefulness is indisputable both from a technical point of view and for the player. In fact, this is a system allowing you to move your units in a rectangular formation. This system prevents all the units from trying to reach the same point when they are made to move because technically only one will reach it before the others and will thus prevent by its presence the others from reaching this point leading these units to all of them. endlessly scramble to reach their destination. The algorithm is quite simple and de fi nes a different destination for each unit. The only problem I had to face was the case where we did not select a multiple of 5 units, in fact the rows being 5, the formation was relatively ugly and especially if one selected a single unit it did not go to the desired point but a few centimeters above because it started the formation. So I modified the algorithm so that instead of forming a line in the order 1 2 3 4 5, it forms it in the order 5 3 1 2 4.

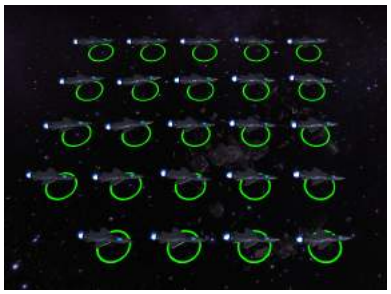


Figure 29 - Training

Camera controller

It is one of the first algorithms that I implemented because it is essential and simple. By default, the camera is oriented 60 downwards to have a fairly general view of the game, but still providing a certain depth. The rest is simple, if we press the classic ZQSD keys or place the mouse on the edges of the screen, we will be able to move the camera. Another feature implemented later allows, thanks to the mouse wheel, to zoom the camera in a way, in fact the camera will approach the ground and reduce its angle of rotation so as to be almost horizontal. This view is more aesthetic than anything else because it is not very practical but o ers a more cinematic and epic view of the game.

4.6.2 Managers

I named all classes with a singleton manager. This single tone makes it possible to limit the number of instances of this class to one and to make it accessible by absolutely all the other scripts. These algorithms are generally essential and require recurrent access from other scripts.

Chat Manager

This program, implemented quite late because it is not essential, allows general management of all messages sent by players. The visual part of this algorithm will be detailed later in the report. Concretely, each time a player sends a message it will be stored here thanks to 2 pieces of information, the player who sends it and its content. Then it will then be sent to all the other players in the game thanks to an RPC (see multiplayer game).

Instance Manager

This makes it possible to make the link between various local actions and multiplayer information. At the start of the game, he will retrieve the current player information thanks to his custom properties (see multiplayer) defined in the waiting room (team, race, color, starting coordinates). Then he will instantiate the starting troops with the right coordinates. This manager will then manage the notion of failure that will occur when a player no longer has a unit or the case where a player disconnects leading to the disconnection of all the others.

Player Manager

This manager will manage everything that appeals to resources, it is in fact here that we add or remove the resource and that we find the functions allowing to pay for constructions or the production of units. It will also manage the di ff erent space stations so that units such as Constructors can bring their resources to the nearest space station.

4.6.3 Game

Gameresource

A Gameresource is simply a resource, minerals, energy, and food. It simply contains its name, quantity, and various methods of modifying its values.

Population

This particular GameResource manages the population. It is made up of a current maximum population and a current population. If the current population is equal to the current maximum population then no more mobile unit can be created. To increase this maximum, houses must be built.

Placement

This is the building placement system. It is broken down into 3 subsystems:

Raycaster:

A Raycaster will send a Raycast down and say whether or not it meets the ground, if not or if the ground is either too close or too far away then a boolean variable will signal that the cell on which the Raycaster is located is no. is not available for construction.

DetectionCell:

This is composed of 5 Raycasters arranged like the points of the five on a die so as to be able to test the square fairly precisely. A DetectionCell represents a space on the Map and will observe each of its Raycasters, if at least one of them appears to be unavailable for construction, the initial green colored space will turn red.

PlacementGrid:

A PlacementGrid will be generated when we want to place a building, it will then obtain the size of the building in squares and will generate as many DetectionCell as the building takes up squares. If at least one of the DetectionCells is red then the PlacementGrid will prevent the player from placing their building. The center of the PlacementGrid corresponds to the mouse pointer but with the box system.

This system required some thought before being developed, I could have contented myself with an algorithm simply preventing two buildings from being placed one inside the other, but I had more ambition and wanted a system. which shows the player which square (s) are problematic. Being in space the distance between the ground and the Raycaster is useless but the system is there and very functional. On the other hand, the box system is an artifice. There is no box strictly speaking but simply a calculation which will create a modulo of coordinates on which a building can be placed. For example if the modulo is 5, and the player places his pointer at x = 12 then instead of the center of the PlacementGrid being at x = 12 it will be at x = 10, if the pointer is at x = 13 that of the PlacementGrid will be at x = 15 and so on.

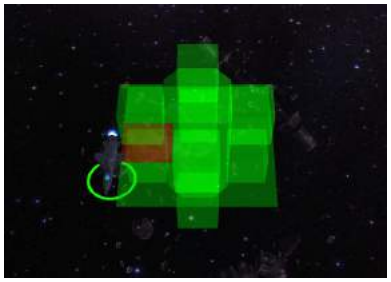


Figure 30 - Construction system

Tasks

The Tasks system has been developed especially for buildings to give them the ability to perform tasks as the name suggests. At the beginning I wanted to create several types of tasks, those of production and those of improvement. In the end, only the first was implemented, the other having become the skill tree. A production job simply produces one unit. Each task is paid for in resources and takes a certain time to be carried out.

Although simple to explain, this algorithm took me a lot of time because it mixes up a lot of previously developed systems, it was necessary to correlate everything and avoid creating bugs. All while updating the interface to give it the right information.

4.6.4 menus

Main Menu and In-Game Menu

Once the heart of the game was well underway I wanted to incorporate multiplayer quickly, but first I had to create menus for it. Given their large number, I had to create a very general algorithm allowing me to simplify as much as possible the implementation of a new menu. Indeed, beyond doing things right for the player, you also need to create tools that are powerful and intelligent enough to make the developer's job easier.

Main Menu :

Allows you to launch the single player, multiplayer mode, go to the options, see the credits, or even quit the game. Single player menu:

Allows you to choose between campaign mode, quick game mode and endless mode.
Campaign menu:
Allows you to review the introductory cinematic, do the tutorial or launch one of the game's campaign missions. Multiplayer menu:
Allows you to choose between joining a game or creating a new one. To join a game, just type in its name and click on Join.

Game creation menu:

Allows to create a game by choosing the map, and if the player creates a multiplayer game to give a name to the game as well as to define the maximum number of players (including AI).

Options menu:

Allows you to adjust certain parameters. It is sorted into several categories:

- Gameplay: Allows you to define the speed of the camera scrolling with the keyboard and the mouse, to deactivate the scrolling with the mouse and to activate or not the help bubbles. - Nickname: Allows you to choose your nickname

- Video: Allows you to choose the resolution, activate or not the full screen, and choose the level of graphic quality of the game.

- Sound: Allows you to adjust the various sound volumes of the game

I reproduced the same menu during the game, they only miss the option to change the nickname because it makes sense that it is impossible to change it during a game.

AlliesMenu

The allies menu allows you to send resources to your allies. It uses the RPC system (see multiplayer).

loading times

Load times have been added and remain relatively unhelpful. Indeed, the game is not very resource intensive, the textures are not in very high resolutions and the models have relatively few polygons, the loading times are barely noticeable, but the system is there and operational . It uses Unity's ability to be able to load a scene while keeping the current one usable, and for the loading progress bar, Unity provides us with a variable between 0 and 1 giving this progress.

4.6.5 Interface

In this subsection I will explain everything that appeals to the interface during the game.

Cards

I named the little cards for each character selected as Card. The panel which contains them then allows each unit to display an image of its model as well as its life gauge to have an overview of the selected troops. Its usefulness is not only visual since clicking on one of these cards allows you to sub-select a unit. This means that the actions available will be carried out on the underselected unit and not all the troops.

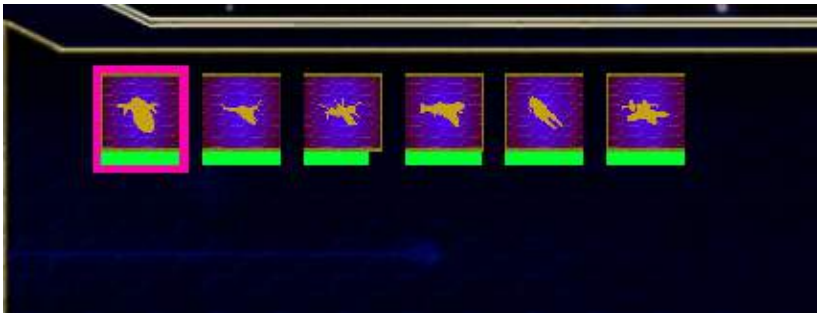


Figure 31 - Cards

Cat

The visual part of the cat is divided into 2 panels: The first does not offer any interaction, it is located on the left of the game screen and displays the messages sent but only keeps them for a while, after a few seconds the messages disappear. The second is accessible from a menu and displays all the messages sent without ever deleting them. It o ers a text box to be able to send messages.
The other method of sending messages is to press enter in the game which brings up a text box to write your message.
These 2 panels use the Chat Manager which serves as a kind of message bank.

advancement bar

It is simply a progress bar that appears when you select

a building under construction and which gives the progress of this one.

panel description

Appears when you hover the mouse over a button to construct a building or instantiate a unit. It contains the name of the object to be created, its resource cost as well as a small description of the object.

Figure 32 - Panel Description

Help

This is a small panel that will display the usefulness of certain buttons when hovering the mouse pointer over them because some buttons use icons which can be confusing. This panel can be deactivated in the options. It occurs in particular when the mouse is hovered over the icons on the minimap.

Figure 33 - Help bubble

Help Menu

This menu contains many tips to allow the player to advance properly in the game. It can help in case the player forgets to use certain game mechanics for example.

jobless button

This button is used to indicate whether or not certain manufacturers are unoccupied, and if so then clicking on this button selects that unit and moves the camera above it.

Monodescr

This panel appears when selecting only a unit and only displays its name and the amount of resource it carries if it is a constructor.

Portrait

The portrait shows the life of the underselected unit as well as its model. This is not the display of a model from the game's Assets but of a camera which is placed in front of the under-selected unit, the portrait simply shows what this camera sees.

Figure 34 - Portrait

Resources

When launching the game, the resource panel will generate as many panels displaying the name and quantity of resources as there are resources added to the game.

Tools

The tools panel is one of the most important features of the game. A tool here designates a button associated with a unit and which will perform an action with this unit. For example: The one to move the unit, to stop it, to display the construction menu, to instantiate a unit, to go and repair a building and so on.

This panel going to be used very often it had to be well coded, to be powerful enough but also easy to use for the developer. In fact, for some particular tools it is necessary to create particular scripts but for the units it is enough to drag and drop the prefabricated of this unit in the list of tools of another unit so that it can generate a tools which create a Task instantiation for example.

Figure 35 - Tools

fl oating life bar

For each instantiated unit, a fl oating life bar will be generated and will show the life of a unit when the mouse is passed over it.

minimap The minimap (or mini-map) is actually a simple camera placed above the game and which gives an overview. Each unit is assigned an icon which will only be displayed on the camera. It also contains other features:

- If a unit is in the fog of war it will not appear on the minimap

- A left click on the minimap moves the camera so that it looks at the place where the pointer is

- A right click moves the selected units to the point on the minimap
- A rectangle indicates the area observed by the camera
- A button allows you to place a marker on the minimap, all players from

Figure 36 - Floating Life Bar

the team will see it too

- A button allows to display or not the bottom of the minimap
- A button allows to display or not the units
- A button allows you to activate or not the system for training troops
- A button allows you to change the management of the display of colors according to the team.

Figure 37 - Mini-Card

Selectionbox

This is the rectangle that will show the player which zone he is selecting.

TemporaryMessage

Placed at the top right of the screen, this zone will display temporary messages to justify the player why he cannot yet perform a

action. For example if he tries to build a building when he does not have enough resources.

4.6.6 Units

In this sub-part I will explain everything that appeals to units in general, more specific units such as buildings or mobile units will be detailed later in the report.

Interactable

This is the basic unit, it has no method or attributes, it only de fi nes that we can interact with this object by right clicking on it.

Selectable

It inherits from Interactable, and makes it possible to make a unit selectable. In fact, this class defines the notion of Highlight (when you pass the mouse over a unit without selecting it, a colored circle will activate to show that the selection is possible) and of selection (the circle is opaque). This circle will have a di selon erent color depending on whether or not the unit belongs to the player, his team or the enemy teams. The class also contains the name of the object as well as its cost, description and the time required to produce it.
This class also makes it possible to generate the field of vision, that is to say how far the fog of war (see fog of war) is dissipated around this unit.

Figure 38 - Normal ted

Figure 39 - Highligh-

Figure 40 - Selected

DestructibleUnit

It inherits from Selectable, and makes it possible to make a unit destructible. In fact, this class defines the notion of the life and death gauge. This is where we de fi ne

the life of a unit, that it can be healed and destroyed.

Resource Unit

She inherits from Interactable. This unit contains a certain amount of a certain type of resource and is destroyed when empty.

4.6.7 Buildings

Constructedunit

It inherits from DestructibleUnit, it is the base building. It has a Task System (see Tasks), and it has an InConstructionUnit attached (see InConstructionUnit) as well as a model of the building but all green and transparent (called Ghost, it is used to display the building when it is desired. to place). Another system is attached to it, that of repair, in fact if the building does not have its maximum life, then one or more builders can be sent to repair it.

InConstructionUnit

It also inherits from DestructibleUnit, each ConstructedUnit has an InConstructio- nUnit, it is the same building except that it is the version under construction, this makes it possible to manage scripts with totally di ff erent operations independently and therefore more simply.
It has roughly the same repair system as the ConstructedUnit but for the construction of the building itself.

Production unit

This building will produce units, its speci fi city and have a rally banner, ie the player will place a banner attached to this building and the units produced will automatically try to join the banner once instantiated.

House

This building increases the maximum population.

Radar

This building can be built a limited number of times (this number can be increased in the skills tree). It does nothing in particular except warn the player when an enemy unit enters its line of sight. A voluntarily larger field of vision than other buildings.

Space station

It is the main building of the base, it is the place where the miners go to deposit their resources. It is he who makes it possible to produce builders.

EnergyFarm

This building continuously generates Energy.

Farm

This building produces food which must then be collected by the builders and brought to the space station.

Asteroid

This unit inherits from ResourceUnit, it contains a de fi nite number of ores (one of the game's resources) and self-destructs when everything has been mined.

Laboratory

This building continuously generates technology points (for the skill tree).

Movable unit

This class inherits from DestructibleUnit and unlike the building it is characterized by the ability to move. It can also be hacked (The unit now belongs to the hacking player).
Each mobile unit is equipped with the patrol system, thanks to a Tools, a mobile unit can be told to go back and forth continuously between its current point and the place pointed by the mouse.

Builder

This mobile unit has a lot of features:

- Harvest: She goes back and forth between her place of harvest and the nearest space station, bringing back resources each time

- Construction and Repair: She can build a building or repair it (the more constructors who build / repair a building, the faster it goes)

This unit took me a lot of time and thought to organize myself well and that the various associated behaviors do not influence each other at the risk of creating bugs. (For example, if a builder was ordered to construct a building while repairing another, the 2 behaviors should not be mixed up otherwise the unit would have had a

chaotic deportment). Each module is simple but managing all these modules is more complex.

Mobile Medical Station
This unit heals surrounding mobile units.

Hacker
This unit can by de fi nition hack another (this causes them to change sides), but also throw electromagnetic bombs as well as reveal an area for a period of time.

Basic troop
Combat Troop by default its attack and defense are balanced.

Light Troop
Combat troop, its attack is powerful and its defense weak, but it is very fast and consumes very little population.

Bomber
Slow but robust combat unit that sends out very powerful missiles but at a low rate.

Cruiser
Single unit very slow, very resistant, but which improves the surrounding allied units and once the right skill is unlocked, it acts as a relay for the ships instantiation buildings. That is, the ships created will appear at the level of the cruiser.

4.6.8 Graphics

Fog of War
The principle of fog of war is simple, the game map is covered with a fog, each unit has a field of vision in the form of a more or less large circle and which will dispel this fog. This system is actually made up of 2 systems, one is purely graphic and the other is only programming oriented.

Graphically it is about a layer which darkens the game and thanks to the shader system integrated in Unity, we can simulate that an area is lit. Technically, when an enemy unit is not in the field of view of any allied unit,

then its textures, its model, its minimap icon are deactivated, the unit is invisible.

This system gave me a lot of problem because it uses shaders that I know very little about and which are not coded in C # but Cg HLSL which I do not know. In addition, very few forums and tutorials exist about the fog of war. It took time and research for all the visual part, the code part being more traditional.

Figure 41 - Fog Of War

4.6.9 Waiting room

This part will detail the functioning of the Waiting Room, that is to say the waiting room in which the players wait for others to join them or for the creator of the game to launch it. This room will independently manage each player from the moment they enter the waiting room. First of all a parameter panel visible to all players will be displayed, only the player who owns it can modify it, this panel will allow the player to define his class, his team, his race and his color. . The creator of the game can add Bots (Artificial Intelligence) and start the game after all players have checked the box ready. Once this box has been checked, a player can no longer modify his properties. At the start of the game,

4.6.10 Skill Tree

Beyond the construction we wanted a notion of progression to be present in the game and what is more that allows the players to differentiate themselves so that the winner is not only determined by the strategy.

Figure 42 - Waiting Room

in combat but also throughout the course of the game. We therefore opted for a skill tree. This is divided into several distinct sub-trees each dedicated to a particular sector which are:
- Light attack
- Heavy attack (bomber)
- Research and development (Hacker)
- Cruiser
- Economy (Construction, resources)
- Defense (turret)

Each of these trees therefore makes it possible to unlock units or skills but also to improve certain statistics such as the construction speed or the damage per second of a unit. In addition, these trees are restrictive, that is to say that when several skills are at the same height of the tree, choosing one of them will block all the others. The goal being that at the end of the game, there is very little chance that the players will have made the same choices and end up with the same base and the same army. Purchasing skills costs technology points.

4.6.11 Online

Photon

To manage multiplayer in our game, we have chosen to use the Phonone PUN 2 asset, in particular for its accessibility. I followed the advice of InfoSpé

Figure 43 - Skill Tree

by implementing multiplayer very quickly. PUN o ers several methods to synchronize the various elements of the game online, first of all we can choose to send a stream which contains various variables to be synchronized, otherwise we can use the RPCs which allow to execute a function on the same instance of an object in other players.

4.6.12 Preparation for AI

Before I started the AI core, I first had to prepare the game for it to arrive. Having never developed an AI for this type of game, my algorithms were not designed for it. In fact, until then, the identification of the player to which a unit belongs went through the system implemented for multi-player, but it is not possible to create “dummy” players corresponding to the IAs. The host of the game (the one who created it) must therefore contain all the AIs locally. I then had to implement methods to identify if a unit is player-owned or AI and react accordingly.
Then I had to implement some tools to be able to allow the AI to control his game. Indeed, features like unit instantiation or building construction were originally designed for the player who is equipped with a mouse and a keyboard in particular. I then developed di ff erent module for each type of action:

- The Army: This object will allow to know if the army of the AI is ready for the attack or not, and thus send it to attack.

- Builders: This module allows you to retrieve builders according to their current actions (does nothing, mine or build) as well as distribute them equitably among all possible tasks.

- Construction: Allows the AI to simply issue the order to construct a building. The position of the building generated in the shape of a snail. That is, buildings are constructed around the initial Space Station to form a square that will become ever larger as the building is constructed. However, the AI checks before building that the terrain is suitable. Thus, if an ore or a building is already present it will automatically move to the next position.

- Instantiation: While the player must select a building to ask him to instantiate a unit, the AI will ask for a unit, and then, it will deduce in which building it is ideal to initiate the task.

- Mining: Allows you to send a constructor to mine. The algorithm will then determine the optimal asteroid or farm to send the troops to. To determine it, the algorithm will assign a score to each compatible resource building and take the one with the lowest score. This score is established by the distance between the Space Station and the resource building as well as by the number of builders already mining on it.

The purpose of these tools is to allow the AI to control its part, we can image these tools as the body of the Bot, and the AI as its head. Once these tools were completed, I was able to move on to the development of AI, that is to say, use them intelligently.

4.6.13 IA

Before talking about how AI works, I will focus on the reflections and the path that led me to this vision of artificial intelligence.

History:

Having no experience in artificial intelligence, and faced with the di ffi culty of developing an AI for a strategy game, we wanted to make it fairly modest in terms of reasoning faculty. Indeed in a strategy game the AI must know how to manage a lot of things, it must destroy the opposing teams but for that it must first develop its base, manage its resources, create enough builders, discover new resources and

construct such or such a building to achieve its ends. So I favored making a simple but functional AI to the detriment of a more ambitious AI at the risk of failing.
My first idea was to mix scripts and reactions, indeed I had no idea how to make the AI manage its resources well and make the right choices. Concretely, this had to result in a series of actions that the AI would have to perform in a scripted manner with a few small liberties. The main lines of its behavior had to be scripted, in particular the construction of this or that building, while other actions such as the creation of new builders had to be automatically managed by the AI. The first flaw of this method is to make this AI last over time, as long as it has not won it must continue to improve its base as well as to train armies, but if the AI follows scripted actions it sooner or later would have stopped. To remedy this I decided that the action sequence that the AI should take would be very generic and that once all the actions had been performed, the AI would start following the same actions again (with a few exceptions. near). However a major flaw persisted, we would have had to create this action list. This is possible but very complicated, it had to be very well thought out so as not to restrict the AI. To overcome this di ffi culty, I wanted to give the AI more autonomy, when it would have run out of resources, to determine whether it needs to create a new unit or to wait or better. allocate, and when this occurs, defer the current task to the end of the list. To push this principle even further I wanted to ensure that, if the AI wanted to create a unit, she must have built the right building. It is easy to automate and that's when I understood where this automation was going to lead me. In reality it is quite simple for the AI to deduce from itself what it needs, this is how I changed my way of seeing things to arrive at the AI that I am about to present.

IA:

Here is how the AI works: at the start of the game, it is given an objective, that of forming a predefined army. The AI will then use all the means in its possession to accomplish this objective; If she does not have the necessary buildings she will build them, but for that she needs a builder which she will then create. If it lacks resources, it may or may not ask for new builders, or even better distribute them, or even wait.
Operation is simple, every task the AI needs to perform is stored in a stack. When the AI completes a task it pops up to perform the task.

coming out of the stack. If she can't do it yet for some reason, she'll stack the task again and then create a new one that will meet the needs of the initial task.

Each of these tasks somehow gets into error when they are not performed, this is detected by the AI and, depending on the error, a suitable task will be created. Once the AI has finished its army, it will detect which enemy base is closest and attack it. Once the assault is launched, only the death of the troops will stop it. Indeed, if an army destroys an opposing base it will not return but will attack the next until victory or death. Meanwhile, in the base, the AI is already preparing its new army as the previous one sees its number of troops decrease.

The only predefined elements are:
- The armies: We define ourselves what the AI will have to create as troops. When an army is finished, the AI moves on to the next, and once the last one has been created, the AI will create that same army again until the end of the game. - The maximum number of buildings: In fact, to prevent the AI from building as many houses as possible at the start of the game, or from building a new battle station each time the others are full, I have set a maximum number for each building, and this number will increase each time the AI completes an army. This helps ensure that the AI base grows continuously and steadily.

4.6.14 Combat

For this defense I also took care of the combat system. The principle is simple, but two major di ffi culties must be taken into account, the behavior and the multiplayer.
Since troops are ships, it is normal for them to defend themselves by shooting at each other with projectiles. Each combat unit observes within a certain radius around it, if an enemy unit enters its perimeter it will attack it if it does nothing else. Unit behavior is tricky when an enemy approaches, should they attack them? continue what she is doing? to run away ? She will just fight back if she does nothing and if not, continue what she is doing.

When a unit attacks another, it will instantiate projectiles at regular intervals towards the enemy which will damage them if they hit them. The troops always stay in front of their enemy and in case of flight follow them as long as the distance between the two is sufficiently small. The firing frequency, as well as

damage per projectile varies by unit, and by skill tree. One of my di ffi culties was to manage this in multiplayer, first of all it is not very optimized to instantiate the projectiles of each troop in multiplayer and to follow their positions, in particular during big battles. So, since balls only move in one direction, and never change, I made it so that when a ball is instantiated in one player it is instantiated in others but the player owning the ball also communicates the force and direction of the ball to other players who will then move it locally to each other. This has only a purely visual purpose since it is the owning player who will destroy or not the ball for all the others when it hits an enemy player. The disadvantage of this method, unlike the one which consists of synchronizing the positions in line, is that in the other members of the game this ball may not hit any unit visually but still do damage because of the latency in multiplayer. However, if the ball position was synchronized online, it would consume such bandwidth that the latency would be even higher.

4.6.15 Debugging

A very large part of my working time has been dedicated to correcting bugs, especially with the arrival of AI and the combat system, and even more so when it comes to multiplayer.

4.6.16 Miscellaneous improvements

Many details have been improved or added. They are neither essential nor very complicated, they are generally help messages in the menus in case the player has for example not given a name for his game or if he opens the allies panel. , or a little message " ***waitting for players. . .***
»In the waiting room to remind the player that he is in the waiting room and his goal.

4.6.17 Group work

Our tasks having been well distributed, we can work a maximum each on our side without needing one or the other. We regularly organize meetings to take stock of what has been done and remains to be done as well as to discuss the direction the game will take, we discuss how to implement certain features, in short, we de fi ne what we want the game to be.

Concretely it works rather well there have never been any major di ff erents and our di ff erent areas of expertise allow us to trust each other.

From a technical point of view we are trading on the Discord software and we have initialized a repository on GitHub.

4.6.18 Optimization

For the last defense, my work was in particular to optimize the game so that it could be launched by a maximum of di di erent computers with variable technical speci fi cations and in particular the less powerful ones. The game was well optimized until then and could run on a lot of machines well. This can be explained by the fact that the game does not use particularly greedy graphics effects and those that are more greedy like shadows, ambient occlusion and anti-aliasing are deactivated if you lower the graphics options of the game.
Game.
In general, RTS type games require a relatively light con fi guration by default. However, it is a type of game that does not lend itself very well to fi xed con fi gurations. Indeed, the number of units that can exist at the end of a game can become very high, and even more so if there are a lot of players in the game. Thus this kind of game goes from very little demanding in terms of resources to very demanding when the number of units is substantial. Our game is no exception to this problem because of only one element of the game which is very demanding and which is the detection of other units. Each player's unit has a detection radius which is used to display or not the enemy units (see fog of war) or to attack them.
From about 80 units on the screen, the number of frames per second on a computer equipped with a mid-range processor will drop below
60. However, if these same units are more scattered and not on top of each other then performance goes up very quickly. However, it is very rare to exceed even 50 units on a single screen, so this is very rare and in all other cases the game runs very well on very light configurations.

I still tried to optimize this, and this is how I realized thanks to the Unity "pro fi le" (tools allowing to see what consumes the most resources in our game) than the components. unit to detect the collision is much more efficient using spheres than boxes (3-dimensional rectangle). The improvement is slight but present. Also the best optimization I could do was to replace my collector system.

read by another. Until then I used the components of Unity called “colliders” which allow very simply to detect when another collider comes into contact with it, when it leaves it and when it stays there. However I do not need to detect when an element enters the zone but only when it stays there or leaves it. Unnecessary calculations then weighed down the game. Moreover, in order for the colliders to be able to detect each other, the object which contained them had to also have a "rigidbody" which is used to calculate the physics of an object. However, in a strategy game in space it is useless to calculate the physics (all the units remaining at the same height in the game and not colliding directly). So I found another method called "overlapsphere" which eliminates the need for a rigidbody and only detects units that are in the collision radius and not when it enters or leaves. With a simple algorithm I can tell when it comes out. But this new system has greatly improved the performance of the game.

After optimization we were able to test that with a mid-range processor (Inter Core i5-7300HQ) the game could support 150 units before seeing its number of frames per second drop below 60 and 220 units for a high-end processor (i7-7700).
Given that the maximum population per base is 100 and that combat units cost between 2 and 10 population points, we can consider that a full army at the end of the game has about 30 to 40 units maximum and therefore end up with more than 150 units in a very small playing area would require a minimum of 4 players, which is the maximum number of players possible during a game.

Moreover these results come from tests carried out with the graphics parameters at maximum, at least the performances are approximately 15% higher.

4.6.19 Minimum and Recommended Con fi guration

Minimum Con fi guration:
OS: Windows 7.8 or 10
Processor: Intel or AMD Dual-Core at 1.70Ghz
Memory: 2GB
Storage: 2GB
Graphics card: NVIDIA or AMD DirectX 11 compatible card with 1.5 GB of memory

Recommended Con fi guration:

OS: Windows 7.8 or 10
Processor: Intel or AMD Dual-Core at 1.70Ghz
Memory: 2GB
Storage: 2GB
Graphics Card: NVIDIA GT1040 or higher | AMD RX 540 or higher

Tests performed using MSI Afterburner Benchmark software.

4.6.20 Balance sheet

For the first defense I developed a maximum of functionalities for the game first of all because I had to do an artificial intelligence for the next one and that I have no experience in this field but also so that between the second defense and the last one I have for task only to polish the game, to optimize it, to debug it. Our goal was to provide a game reaching a certain standard of quality. We wanted anyone to be able to see our game and think that it is a real game intended for marketing and not an educational game developed in 6 months by 4 students. We believe that this standard has been achieved.

5 Conclusion

After many hours of work, our game is fi nalized and all features have been implemented. We knew how to work in the detail of the game and to refine it, and it is therefore with pride that we present to you, for this third defense, Year Zero.