Defense Report 1 - Year Zero

Bearth Studio

March 13, 2019

timothee.ribes nicola.brankovic

enguerrand.vie axel.ribon

Contents

1 Introduction

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 artistic direction will be 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 will be quite epic with big catchy themes. But he will also juggle with calmer and more melancholy themes to recall the desperate condition of humanity. The epic aspect being purely playful and the dramatic aspect serving to reinforce the intentions of the scenario.

For this first defense we made sure that the important bases of the game were ready in order to leave only the artistic parts and the arti fi cial intelligence for the other defenses.

2 Development on distributed tasks

2.1 Timothy

My work on this project is spread over several areas of the game which are diverse. This is mainly the tutorial of the game, the sound aspects and help with the design of the website.

2.1.1 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.

2.1.2 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.

2.1.3 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

The actions of the units themselves will have to be developed for the next support at the same time as the internal graphics of the game. In addition, the units will have voices which respond to the orders given. 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.

2.2 Nicola

I am going to present here my work carried out on this project, the first thing that was given to me to do was the scenario, in fact even if the theme and the atmosphere of the game had already been decided upstream, the scenario allows to develop the play in one direction and achieve a certain harmony between the general style chosen, the campaign and the site.

2.2.1 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. Taking into account the fact that the site is not obligatory until defense 2, this is only a sketch of the site, it will surely be remodeled later.

2.2.2 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,

incomplete or missing gameplay, but also some implementations that will be necessary to continue the campaign.

2.2.3 Scenario

In the future, albeit distant, mankind has finally overcome the di ffi culies it has encountered; war, pollution, energy crisis, overpopulation ... The world lived in peace. But nothing is eternal, and man's thirst for space conquest and the discovery of extraterrestrial life has turned against him. Returning from a long mission to the outskirts of the solar system and after having visited many moons and planets, an astronaut team would have brought back an extraterrestrial parasite, taking control of the body of its host, making it into one. empty shell, free from any desire, or more precisely, than any desire which could have previously animated them. The infected people only thought about spreading the parasite in as many hosts as possible, and once their numbers were sufficient, they took up arms. We must not believe that humanity has stood idly by in front of their sick comrades. But the parasite spread very quickly, and resources quickly became limited. And the only cure that could be developed was a vaccine, preventing the parasite from implanting in the host but powerless against the infected. A big step forward in the face of this threat, but a little late, a large part of humanity was already under the in fl uence of the parasite, there remained only a handful of resistance fighters, ready to do anything to recover their resources. land, to fight against their brothers, who today behaved like common ants. But every day mankind was retreating, until one day they were forced to leave earth, forced to survive in mankind's last spaceships. Today, humanity is brought to its knees and the present inhabitants of the earth are chasing them and gradually recreating their military arsenal destroyed during the war. Humanity can no longer let it go, it is, armed to the teeth, in vessels worthy of the best science-fiction, that a handful of humans, will have to fight, for one day, return to earth and mark the start of a new era, Year Zero.

2.3 Enguerrand

My work so far has mainly been done in Photoshop. Considering that the graphic and artistic aspect of the game constitutes the major part of my work on Year Zero, I have given myself the means to achieve beautiful things, integrating as well as possible into the universe that we have created. Besides the artistic creation on Phosothop, I also had to order all the

elements of our game within Unity so that the visual rendering is pleasant and pleasant, as well as ergonomic and practical. For the next defense, I will focus in more detail on the import of 3D models of the various units and buildings, as well as on the generation of the map and the combat system.

2.3.1 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.

2.3.2 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 1 - Police Orbitron Black

Figure 2 - Police Space Age

2.3.3 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.

Figure 3 - Main menu

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.

Figure 4 - Testing some buttons

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.

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 5 - Button texture

Figure 6 - Button for a fill bar

Figure 7 - Square button with rounded edges

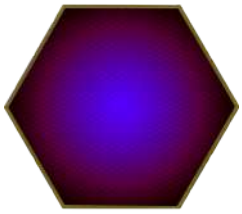

Figure 8 - Hexagon shaped button

Figure 9 - 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.

Figure 10 - Main menu

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 here, it is only visual.

2.3.4 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 ...

2.3.5 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.

2.3.6 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 11 - Interface image

Figure 12 - In-game interface

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).***

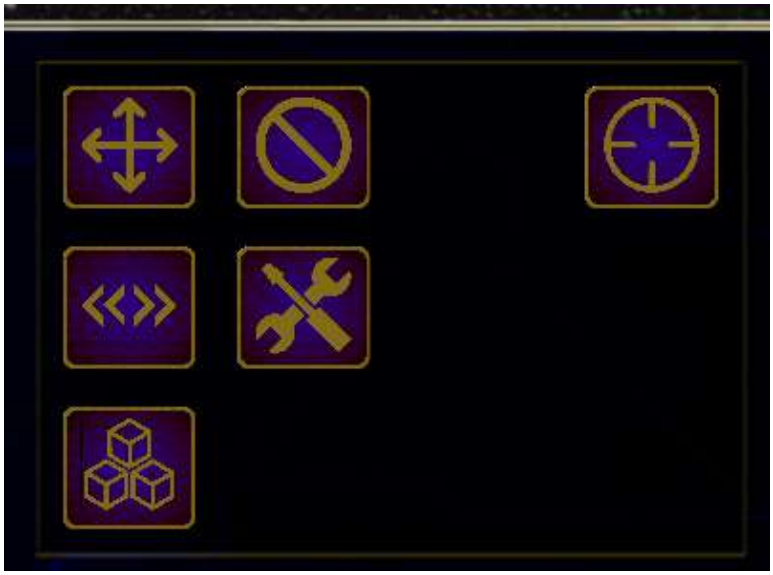

Figure 13 - Buttons with icons

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.

Figure 14 - Buttons with icons

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 15 - Resource buttons

The minimap buttons are also present, also formed by their texture and an icon, as for all the others made by me on

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

2.4 Axel

I am going to present here each functionality that I added to the game, my feelings, the problems that have arisen as well as my forecasts for the next presentations.
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.

2.4.1 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, a right click 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 Raycast principle, this Unity feature 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

I developed this algorithm quite recently because it 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 to move its units in formation, for the moment only the rectangular formation is implemented. 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. jostling endlessly to reach their destination. The algorithm is quite simple and defines 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, indeed 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 as 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.

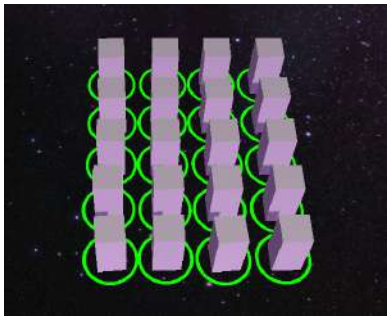

Figure 18 - Training

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.

2.4.2 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 which 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.

2.4.3 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 square on the Map and will observe each of its Raycasters, if at least one of them shows up as not available for construction, the starting square colored green 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.

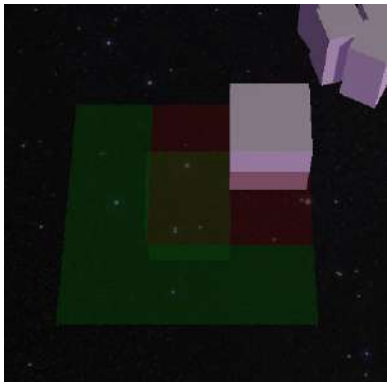

Figure 19 - Construction system

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.

2.4.4 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 (still absent) and quick game mode.

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: Adjusts the volume of the game

I reproduced more or less the same menu during the game, they only miss a few options like changing 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

The loading times were added recently and are still unnecessary for now. Indeed, since the game consists of only simple polygons and solid textures, 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.

2.4.5 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.

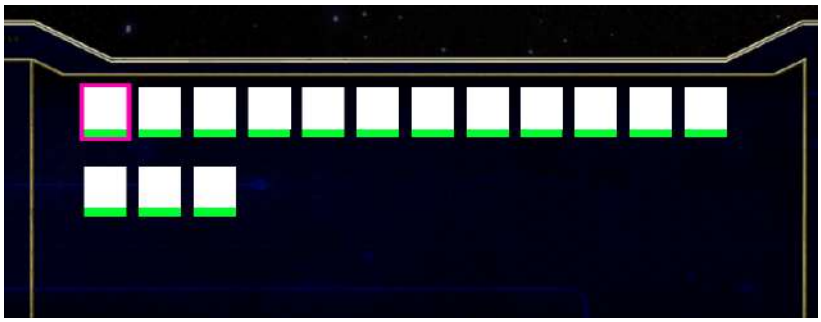

Figure 20 - Cards

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 which appears when selecting 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 21 - 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 22 - Help bubble

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 a unit and only displays its name for the moment, the final purpose being to display other information specific to that unit. For now, we only see the quantity and type of resource transported by a manufacturer.

Portrait

The portrait shows the life of the underselected unit and 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.

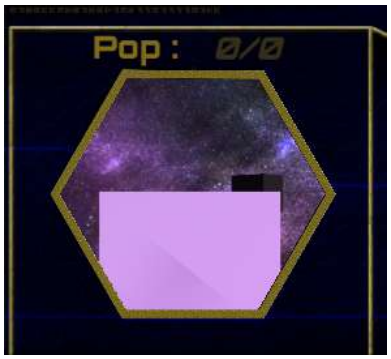

Figure 23 - Portrait

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 refers to 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.

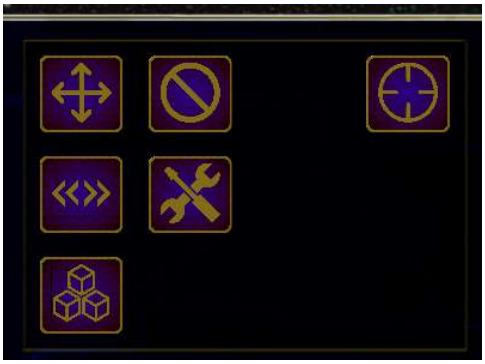

Figure 24 - Tools

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.

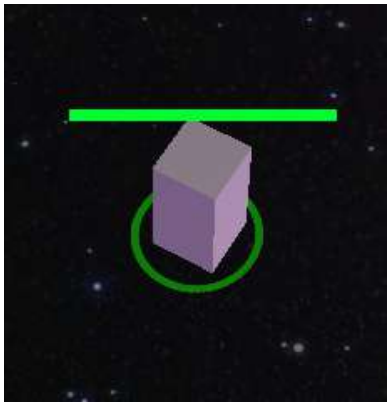

Figure 25 - Floating Life Bar

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 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 the players of 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 26 - 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 area will display temporary messages to justify the player why he cannot yet perform an action. For example if he tries to build a building when he does not have enough resources.

2.4.6 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.

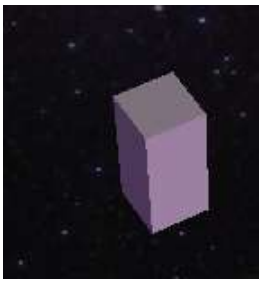

Figure 27 - Normal ted

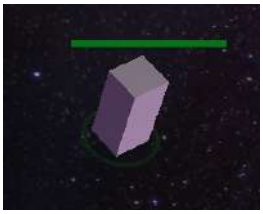

Figure 28 - Highligh-

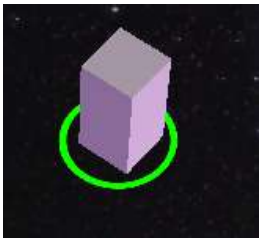

Figure 29 - Selected

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 the life of a unit is defined, where 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.

2.4.7 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 will drop off

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 must not be mixed up otherwise the unit would have had a chaotic behavior). 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 be hacker of other (this makes them change camps), but also many other functionality which will be unlockable in the tree but which are not yet incorporated.

Basic troop

Combat Troop by default its attack and defense are balanced.

Light Troop

Combat troop, its attack and defense are 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

Unique 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 instantiation buildings of ships. That is, the ships created will appear at the level of the cruiser.

2.4.8 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, model, minimap icon are disabled, the unit is invisible.

This system gave me a lot of problem because it uses the shaders that I

know very little and who do not code 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 30 - Fog Of War

2.4.9 Waitting room

This part will detail the functioning of the Waitting 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 he enters 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. At the start of the game,

2.4.10 Skill Tree

Beyond this, thanks to the construction, we wanted a notion of progress to be present in the game and what is more that allows players to differentiate themselves so that the winner is not only determined by the strategy in combat but also throughout the course of the game. We have

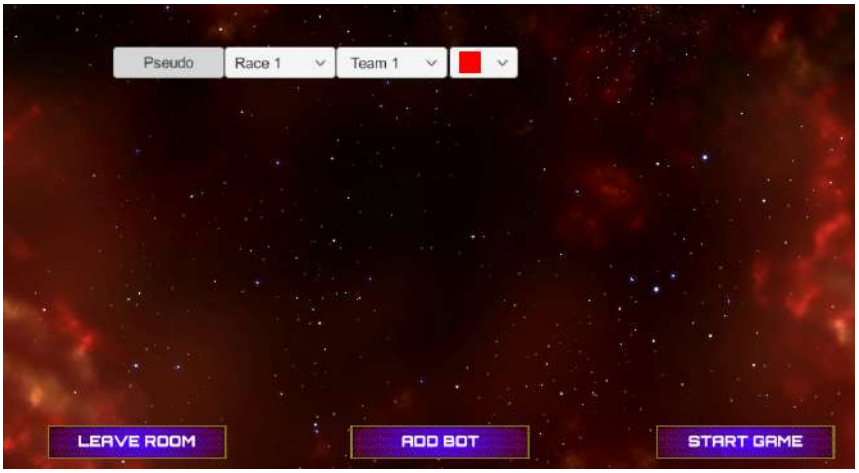

Figure 31 - Waiting Room

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.

2.4.11 Online

Photon

To manage multiplayer in our game, we have chosen to use the Phon PUN 2 asset in particular for its accessibility but above all because the tool provides us with a server hosted on the Photon servers which o rent

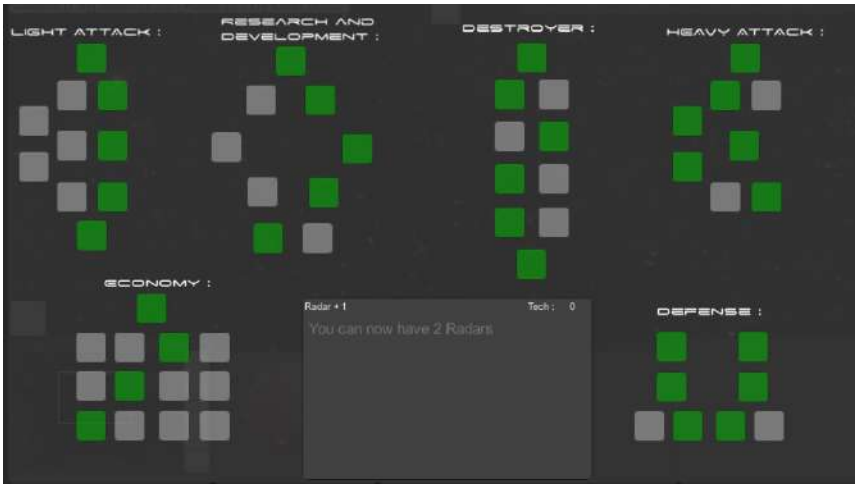

Figure 32 - Skill Tree

much better stability than if we had hosted it ourselves.
I followed the advice of InfoSpé by implementing multiplayer very quickly. PUN o ff 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 all of them. instances of the same object.

2.4.12 Assessment and Outlook for the next defenses

I am very satisfied with my work, compared to the schedule I am ahead, a calculated advance because my work rate on the project will have to drop as the exams approach because we will have a lot of work. The idea was to have developed a very large majority of the game mechanics for this defense in order to leave only the combat system and especially the artificial intelligence for the others. The latter will prove to be a very big challenge for us since we have no experience or knowledge in the matter. So for the next defense, I plan to have done the combat system as well as to have started well in Arti fi cial Intelligence, at least from a theoretical point of view.

3 Conclusion

To sum up, the basics of the game are there and the next defenses will see the arrival of artificial intelligence as well as the textures, models and various visual e ff ects of the game. From the first, we took advantage of the start of the second semester to work as much as possible so as to have less work when the other classes get tough.