Book of specifications

From

Team

DEDALUS

Romain Biessy Renaud Gaubert Aenora Tye Erwan Vasseure

Contents

Chapter 1

Introduction

We are a group of E1 students, who already have some experience in IT. We came to know each other during the Caml TP. And thought about the game based on our common passion for Minecraft and other RTS[1] games like Starcraft.

Even though we did not start coding, we still had many group meeting where we did brainstorming about the project. And spent at least 15 hours speaking together.
However, the project's general gameplay was fixed unanimously on the first reunion. On the other meeting, we discussed about the scenario, the task planning and some code examples.

When we assigned the different task to our members, we had in mind the idea that everybody should know how most of the game works and not only one member.
Therefore, we had to split the project in a way that would help us later, modules. Thus even if we did not begin to code the game, we have some kind of base.

We also decided that we would make a 3D game at the first person. Also we thought about a mix between RTS and FPS[2].
Creating a game has never been an easy task but, the game we imagine could be described as ambitious. Nevertheless, our teamwork should not be underestimate.
We will code this game with the help of Mogre[3] which is a 3D engine using C#.

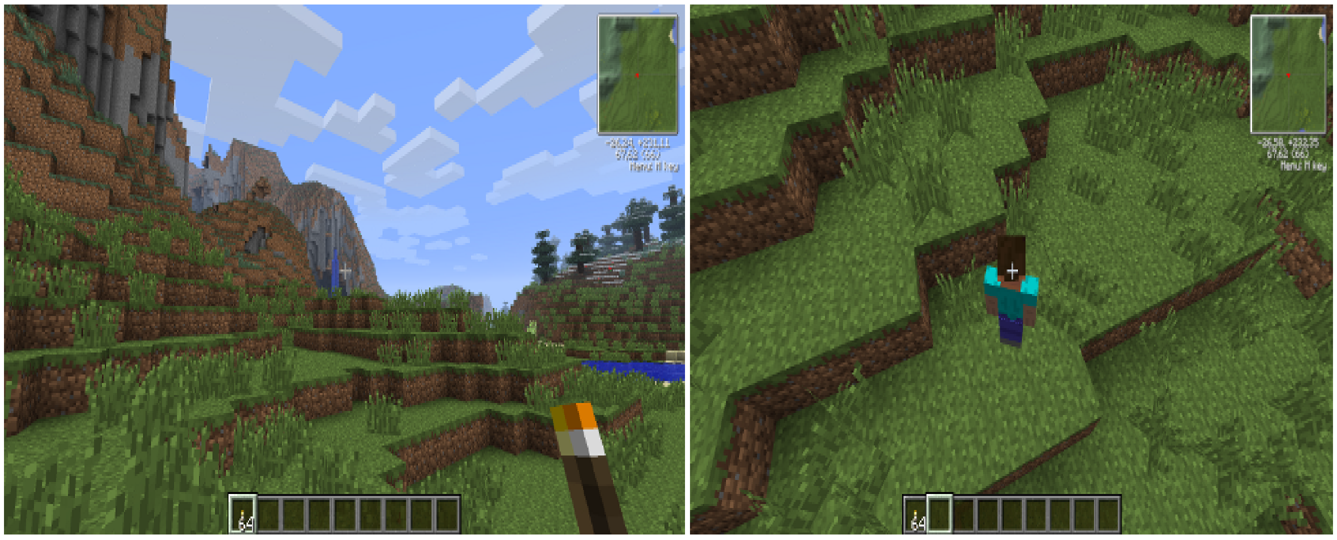
[1]Real Time Strategy
[2]First Person Shooter
[3]Managed Object-Oriented Graphics Rendering Engine

Chapter 2

Gameplay

2.1 RTS & FPS & minecraft-like

We decided that our game would be a mix of an RTS and an FPS on a minecraft-like terrain. What does that mean ? Basically, it means that you will be a following a character at first person. Besides Minecraft-like terrain means that he will be walking on cubes :



This images are from the game Minecraft

This character will have the ability to break those cubes or add others. But what makes it an RTS is the fact that he will be able to control other units, send them to combat and join them. Of course, for it to be an RTS he will have to face opponents that we be controlled by the AI or you could play against other players.

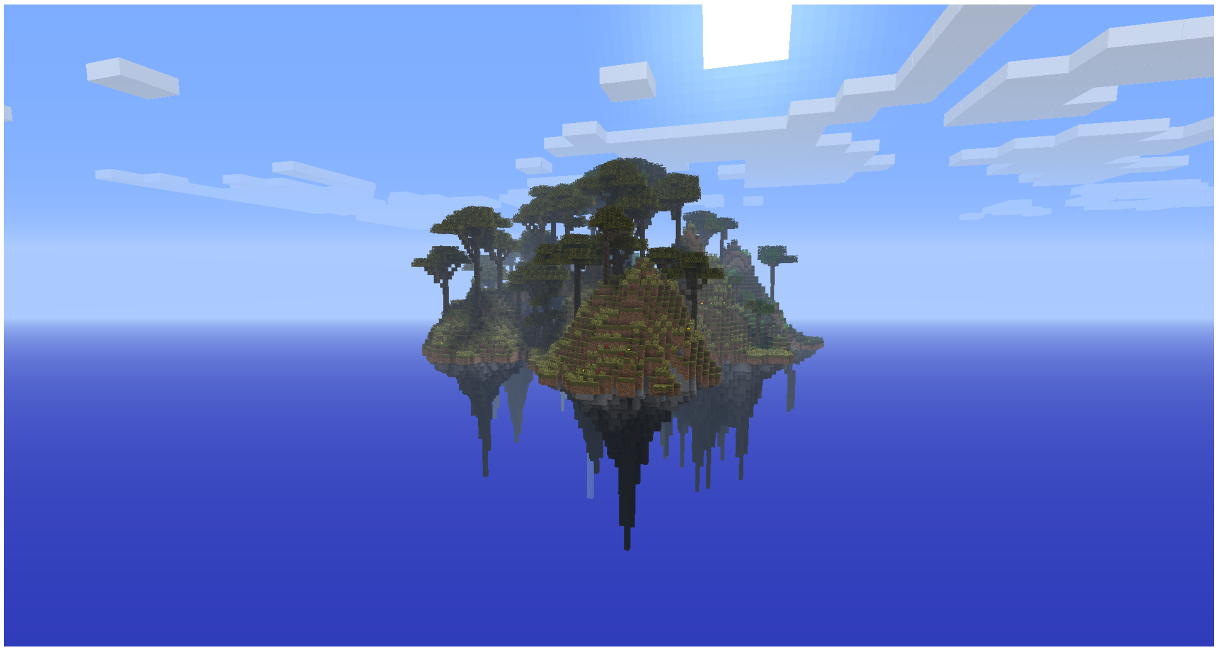
The character will also have the option to be able to add buildings that will be constructed by workers. Those will be the buildings that usually spawns units.
As you have already guessed the goal will be to eXplore, eXpand, eXploit, and eXterminate. But you will also have a scenario if you wish to play one.
Finally, you will also be able to face other players intense strategic battles.

2.2 Background

This is year 2432, humans have spread throughout the universe. But criminality rates have been exploding for a decade. Thus a decision has been made to create jail planets. Therefore, any human proven guilty of any crime shall be exiled to those planets.

During these past years, justice was harsh and you were one of the many sent to those planets that were innocent.

However the stellar ship that was supposed to drop you on the planet was hit by meteorites. Still, you survived unharmed using the escape pods. The planet where you were sent has a little particularity : it is composed of flying islands. You are now on one of the many islands which compose this planet.



This image is from the game Minecraft

Chapter 3

Goals & interest

3.1 Learning new languages

Since half of the group are inexperienced programmers, one of the major goal of the project will be to learn about **clean** computer programming.

Of course it will not be limited to C#, because of the fact that we are working with OO[1] and learning clean computer programming and to ensure that everybody can understand the code, we will be using UML[2].

Also learning about clean computer programming means that we will be using some OO well known principles such as inheritance and abstract class.

3.2 Learning about web

Since PHP[3] is a easy to use and well known language, we will be using it. But, we will use the Symfony2 framework with Doctrine (which is an ORM[4] that permits to represent data as objects).
As for Symfony, it is a PHP Web Development Framework. Thus it provides generic functionality and ”helps” you writing clean code.
What clean code means is the fact that symfony's own architecture helps organizing our code.

The power of symfony lies in the MVC[5] organization. Indeed Symfony2 is developed in an architecture that separates the representation of information from the user's interaction with it.
Thus when the user will try to access to a website URL, this URL will be submitted to the router, which will then call the right PHP function. After processing the data relative to its purpose, this function will then send variables to a TWIG page which will then be displayed.

[1] Oriented-Object
[2] Unified Modeling Language
[3] PHP : Hypertext Preprocessor
[4] Object-Relational Mapping
[5] Model View Controller

3.3 Learning about 3D

As you might have already understood, we are using Mogre which is a scene-oriented, real-time, flexible 3D rendering engine 3D library.
The class library abstracts the details of using the underlying system libraries like Direct3D and OpenGL and provides an interface based on world objects and other high level classes.

Thus, the user can choose between Direct3D and OpenGL without us having to rewrite the entire code for each libraries. As its name states, OGRE[6] is ”just” a rendering engine. As such, its main purpose is to provide a general solution for graphics rendering. Though it also comes with other facilities (vector and matrix classes, memory handling, etc.), they are considered supplemental. It is not an all-in-one solution in terms of game development or simulation as it does not provide audio or physics support, for instance.

You should also know that having an experience with 3D is generally far more interesting than having an experience with 2D. Thus we decided to do some 3D.

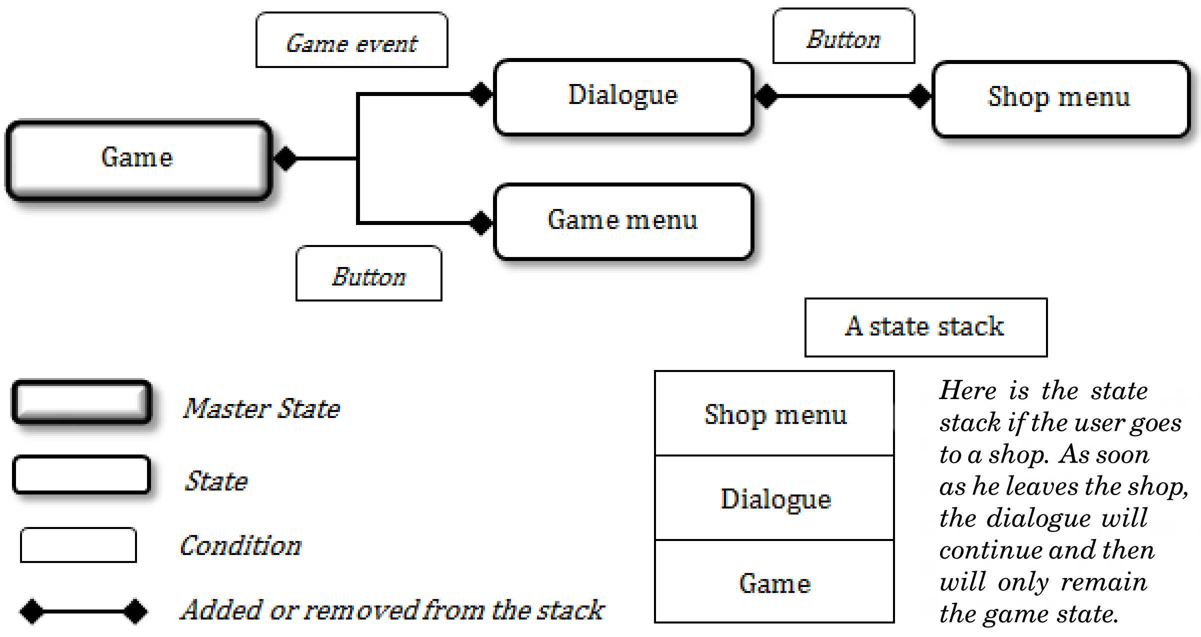
[6]Object-Oriented Graphics Rendering Engine

Chapter 4

Behind the scene

4.1 Language & Game design pattern

Our project will be coded in **C#** .NET 4.0 mainly because of the OO paradigm and also for the large choice there is about 3D engine and libraries. The OO let us use the **game state** pattern in order to structure the code. Basically, we consider our game as a stack of different states. Here is an example of a game state:



Here is the state stack if the user goes to a shop. As soon as he leaves the shop, the dialogue will continue and then will only remain the game state.

There is always a master state at the beginning of the stack. Then other states may be added or removed; only the state at the top of the stack is updated and drawn. Depending on some condition as an input entry or a game event, each state can call any other states via a state manager.

This pattern is advised for RTS games, among others. As we are working in a group, it is important to keep homogeneity throughout the program. Besides, a well-structured code will be easier to understand and to edit.

4.2 3D Engine & Library

The 3D engine we will use is **Mogre**. Initially, this is an API[1] coded in C++ called OGRE; Mogre is an advanced .NET 2.0 wrapper for OGRE. We have chosen this API since it is known as an effective, handy and well-documented 3D engine in C#.
Our GUI[2] will be implemented with the library **MyGUI**[3]. It is very flexible since all parameters are settable directly in the XML's files.
Since our 3D engine does not handle sound we have to use an audio library. We will use **NAudio** - an open source .NET library - because of its simplicity and completeness.

4.3 Softwares

Our IDE[4] will be **Visual Studio 2010 Ultimate**. We will also need **Blender** so that we can create our own meshes and then use them for our game with the script Blender2Ogre. Basically, all of our 3D objects will be created with Blender except for the terrain's cubes which are generated directly in the source code. The 3D engine we will use is **Mogre**. Initially, this is an API coded in C++ called OGRE; Mogre is an advanced .NET 2.0 wrapper for Ogre. We have chosen this API since it is known as an effective, handy and well-documented 3D engine in C#.

There are two key points in our projects which will have to be implemented using some well-known algorithms:

- **Terrain generation**. We want our land to be generated pseudo-randomly in order to create realistic islands - apart the fact that they are suspended in the sky. We want our islands to have mountains, jungles and rivers but we will definitely not build them ourselves. What we need to implement is the algorithm of **Perlin noise** in 3 dimensions. The idea for one dimension is to generate a list of random points, then to create a function which goes through these points. We repeat this step many times with an amplitude between the points getting smaller and smaller and then we add all the functions obtained. The concept is the same in 3 dimensions.

- **Pathfinding**. This is part of the AI[5] for the NPC[6]. The goal is to find the shortest way from a point A to a point B. A common solution for this problem is the **A*** (A star) algorithm. Basically, it tests all possibilities of path and then remembers the path which is getting closer to the arrival. This method is efficient and quick as long as there is no intricate labyrinth that is why we think A* is adapted for our project.

[1]Application Progamming Interface
[2]Graphical User Interface
[3]Multilayer and overlappable GUI System
[4]Integrated Developement Environment
[5]Artificial Intelligence
[6]Non Player Character

Chapter 5

Planning

We decided to split the work into functional tasks. Thus we first decided to ask each member's preference. After asking them about it, we decided to divide the game as the following table :

First oral	**Renaud**	**Aenora**	**Erwan**	**Romain**
Terrain generation	+	+		
Player			+	+
Sky			+	+
Web Site	+	+	+	+
Menu		+		+

Second oral	**Renaud**	**Aenora**	**Erwan**	**Romain**
NPC with simple AI	+			+
Mouse actions			+	+
Advanced menu		+		+
Advanced terrain generation	+	+		
Buildings			+	+

Third oral	**Renaud**	**Aenora**	**Erwan**	**Romain**
RTS		+		+
AI	+		+	
Save	+	+		

Fourth oral	**Renaud**	**Aenora**	**Erwan**	**Romain**
Multiplayer			+	+
Map & story editor	+	+		
Story	+	+	+	+

Chapter 6

Conclusion

We hope that this project will help us in multiple domains of computer science such as 3D, OO, web, . . .
Also we are glad to sail for this great adventure that is realizing a 3D game.

We acknowledge the difficulty of developing a 3D game but we are still motivated. Besides, we know that there will be ”rush” periods. Still we will try our best to reduce those periods.
We are confident that our objectives will be achieved for the fourth oral presentation.