CAHIER DES CHARGES

From

Team Dedalus

Islands

Romain BIESSY Renaud GAUBERT Aenora TYE Erwan VASSEURE

Table of contents

Chapter 1

Introduction

Even though we didn't start coding, we still had six to seven group meeting to discuss about the project. And spent at least 15 hours speeking together.
However, the project's general gameplay was fixed unanimously on the first reunion. It will be a 3D RTS at the first person. Thus a mix between RTS and FPS. Creating a game has never been an easy task but, the game how we see could be described as ambitious. Nevertheless, our teamwork should not be underestimate.

When we assigned the differents 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. Therfore, we had to split the project in a way that would help us later : modules. Thus even if we didn't begin to code the game, we have some kind of base.

Chapter 2

Gameplay

2.1 Section 1

2.1.1 Sub 1

Chapter 3

Goals and 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 computer programming.

Of course it won't be limited to C#, because of the fact that we are working with OO, to ensure that everybody can understand the code, we will be using UML.
Also we will be discovering and thus learning about the following languages :

- HTML5/CSS3 and PHP/MySQL with maybe some JavaScript for the website;
- XML because this is how the GUI layout will be written.

We will be using some OO well known principles such as inheritance and abstract class. We also might use Symfony2 (with Doctrine) for the website.

3.2 Learning about web

Symfony is a PHP Web Development Framework. Thus it provides generic functionality and "helps" you writting good code. What i mean with good code is the fact that because of its own architecture organize your code.

The power of symfony lies in the MVC organization. Indeed Symfony2 is developped 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 submited to the router, which will then call the right PHP function which will then send variables, after processing the data relative to it's function, to a TWIG page which will then be displayed.

3.3 Learning about 3d

As you might've 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 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 doesn't 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. Since we knew about Mogre,

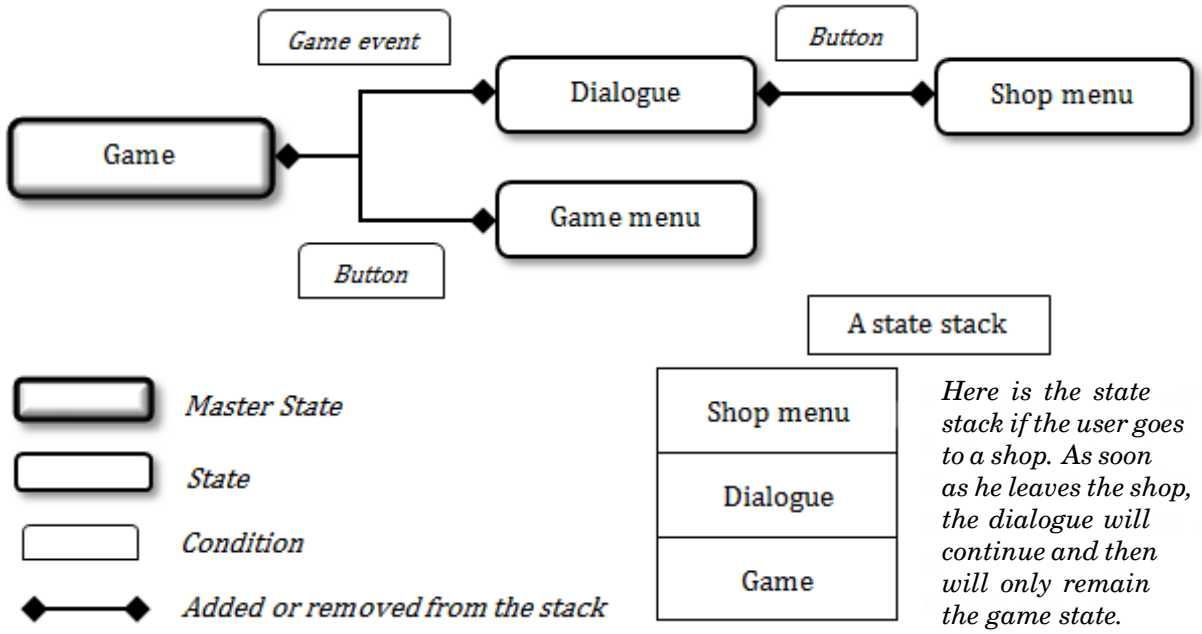
3.4 Goals

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 **OOP!**[1] 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:



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!**[2] games, among others. As we are working in a group, it's important to keep homogeneity throughout the program. Besides, a well-structured code will be easier to understand and to edit.

[1] **OOP!**
[2] **RTS!**

4.2 3D Engine & Library

The 3d engine we will use is **Mogre!**[3]. Initially, this is an **API!**[4] 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!**[5] will be implemented with the library **MyGUI!**[6]. It's very flexible since all parameters are settable directly in the XML's files.
Since our 3D engine doesn't 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!**[7] 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's 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!**[8] for the **NPC!**[9]. 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 's why we think A* is adapted for our project.

[3] **Mogre!**
[4] **API!**
[5] **GUI!**
[6] **MyGUI!**
[7] **IDE!**
[8] **AI!**
[9] **NPC!**