

CAHIER DES CHARGES

From

Team Dedalus

Islands

Romain BIESSY

Renaud GAUBERT

Aenora TYE

Erwan VASSEURE

Sommaire

I	Project presentation	3
1	Gameplay	4
1.1	Section 1	4
1.1.1	Sub 1	4
2	Goals and interest	5
2.1	Learning	5
2.1.1	Sub 1	5
II	What must be done	6
3	Behind the scene	7
3.1	Language & Game design pattern	7
3.2	3D Engine & Library	8
3.3	Softwares	8
3.4	Algorithm	8

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 speaking 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 different tasks 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 didn't begin to code the game, we have some kind of base.

Première partie

Project presentation

Chapitre 1

Gameplay

1.1 Section 1

1.1.1 Sub 1

Chapitre 2

Goals and interest

2.1 Learning

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 have to use XML with the GUI

2.1.1 Sub 1

Deuxième partie

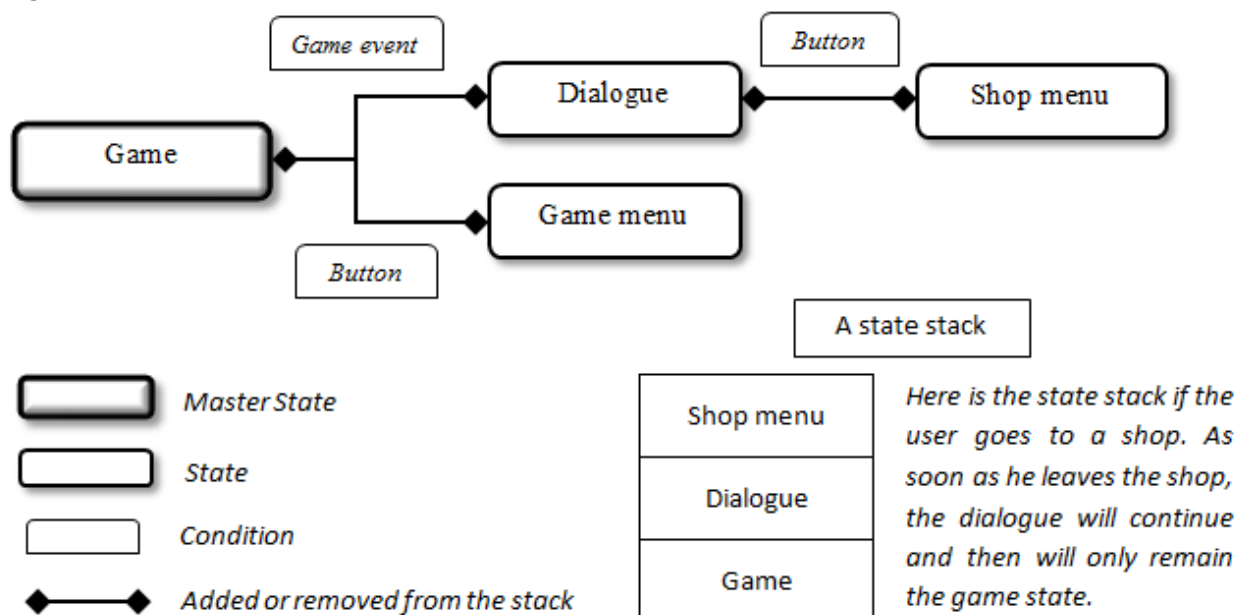
What must be done

Chapitre 3

Behind the scene

3.1 Language & Game design pattern

Our project will be coded in C# .NET 4.0 mainly because of the OOP¹ 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 RTS² 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

1. Oriented-Object Programming
2. Real Time Strategy

understand and to edit.

3.2 3D Engine & Library

The 3d engine we will use is **Mogre**³. Initially, this is an API⁴ 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 GUI⁵ 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.

3.3 Softwares

Our IDE⁷ 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.

3.4 Algorithm

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 AI⁸ for the NPC⁹. 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***) 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's why we think (**A***) is adapted for our project.

3. Managed Ogre

4. Application Programming Interface

5. Graphical User Interface

6. Multilayer and overlappable GUI System

7. Integrated Development Environment

8. Artificial Intelligence

9. Non Player Character