Magrathea Engine

Document :	**Activity report**
Version :	1.0
Group leader :	Serge Lages
Group members :	David Callu Louise Jacquot Laurent Marchal Nicolas Remia Fabien Rioli
Date :	02/03/2006
Recipient :	Flavien Astraud

Abstract

The aim of this documentation is to give an overview about the progress of this project. The Magrathea Engine is an environmental engine designed with powerful tools to provide a complete solution for managing realistic simulations.

© Copyright 2005 – PFE EPITECH
pfe@epitech.net

Table of content

1. Engine progess.....3
1.1 - Development of the general engine's structure..... 3
1.2 - Development on the terrain's LOD algorithm (level of details)..... 4
1.3 - The streaming engine..... 5
1.4 - Atmosphere..... 5
1.5 - Terrain's file generation..... 7
1.6 – Web development..... 9

2. Engine Gantt..... 10

3. Tool progress..... 11
3.1 - Formation and preparation..... 11
3.2 - Starting core implementation..... 13

4. Tool Gantt.....14

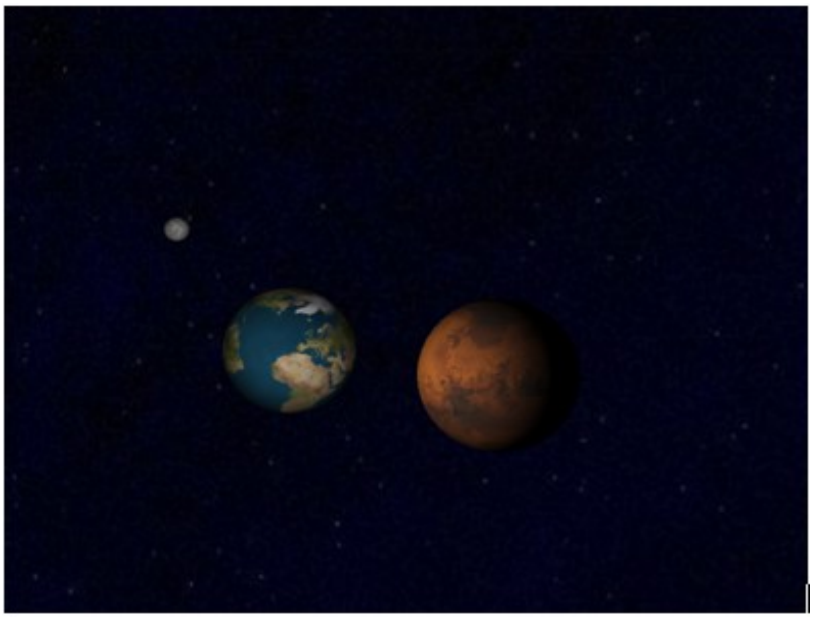
pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

1. Engine progess

1.1 - Development of the general engine's structure

This first step started with the development of several important parts of the engine:

- All configuration files of the engine are in the XML file format, so we needed to add to the engine features to load them. It has been done with the Expat library, which allows a fast and reliable parsing of XML files. But this library is written in C, so we need to wrap it into our engine with some class allowing a simple use of XML.
- The next step was to build the planet's rotation and movement system into the engine. It is based on OpenSceneGraph, so it was simple to add these features by using the Node system offered by this engine. Each planet is driven by a rotation speed around the sun, a rotation speed and an angle around itself, and a list of children planets which turn around it.



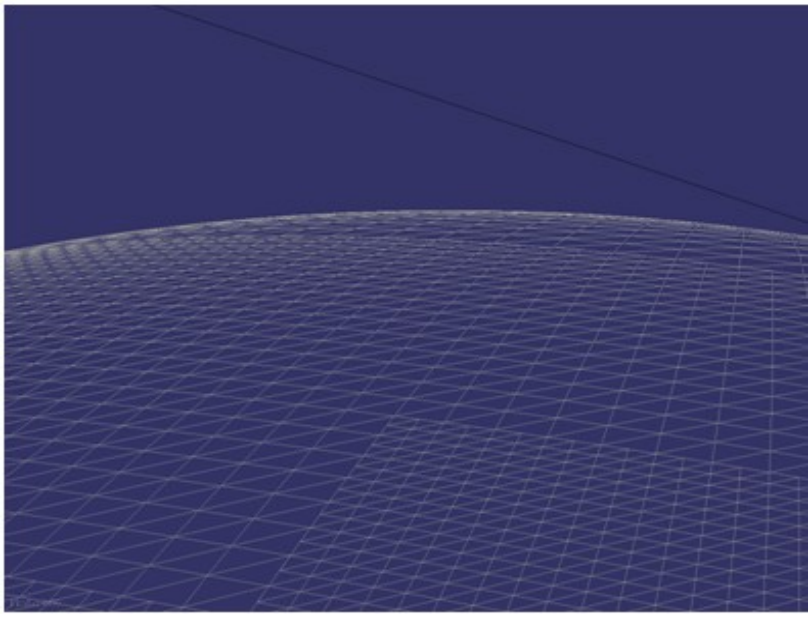
- Then we developed a camera system adapted to universe navigation. We used OpenSceneGraph features to reach it. So, we manage to detect the proximity of planets from the camera, and if a planet is near enough, we apply the same rotations and movements to the camera than the planet. So we can easily visit the planet.

pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

1.2 - Development on the terrain's LOD algorithm (level of details)

To render very large planets without drops in the frame rate, we needed to develop a LOD algorithm. It is based on the construction of a level of detail's tree, and according to the camera position, it select appropriates leafs to render. We made two implementations of this algorithm into the engine:

- The first version worked but with significant frame drops during the execution. Indeed, it was not taking profit of the OpenSceneGraph's features and was building the entire level of detail's tree into one Node of the engine. This version was abandoned.
- The second version is based on the first, but it was implemented using the OpenSceneGraph's features. I.e. each chunk of a level of detail is a node of the scene graph, and the tree is traversed by OpenSceneGraph with for example an automatic culling of invisible leafs. This version also uses the asynchrony loading system of OpenSceneGraph, and each chunk of terrain is loaded in memory only if it is needed. These loading are done in a separated thread which takes care to not drop the frame rate.



pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

1.3 - The streaming engine

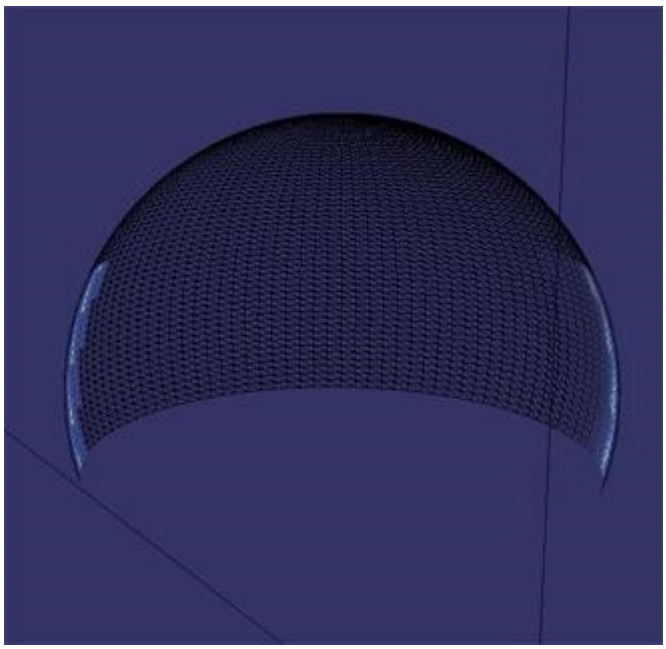
To load objects when rendering, we need a streaming engine. At the begining we though to make it one from scratch. But OpenSceneGraph provides a performant streaming engine, more adapted to the rendering engine. The main goal is to be able to do tasks in background but keeping a good frame rate, so we choose to use the OpenSceneGraph Streaming engine to load our objects.

1.4 - Atmosphere

For the Magrathe Engine, we need an atmosphere for the planets, to have a nice effect from space, like from the ground. To do it, we follow the algorithm of Sean O'Neil (more informations about it in the wiki). To use it in our engine, we have written a xml format to save parameters.

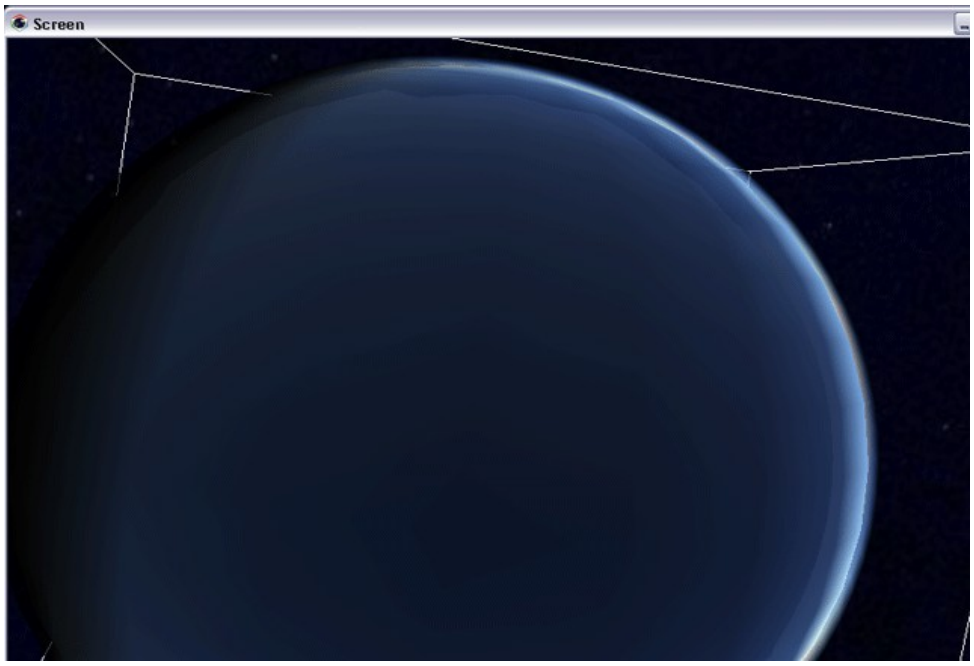
```
<Atmosphere Activated="True">
   <Parameters Kr="0.0025" Km="0.0015" G="-0.85" ESun="15.0" Width="120"/>
   <Color Red="0.650" Green="0.570" Blue="0.475"/>
 </Atmosphere>
```

The atmosphere is made of 2 spheres, one for the extern halo around the planet, and an other above the planet. The algorithm calculate the color for every vertex of the spheres. In this system we can configure some parameters: the colors of the sky, the width of the atmopshere, and some parameters about the light.

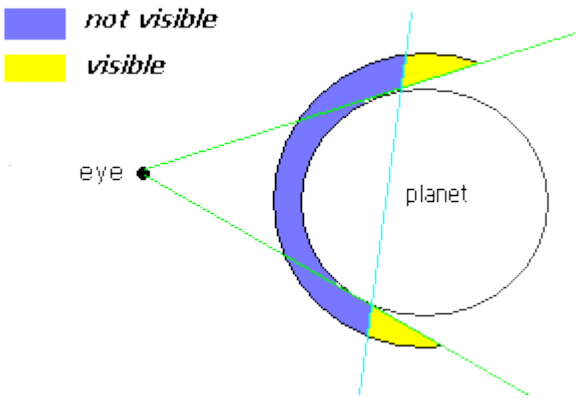


pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

I have written the begining of an article about our atmosphere in our wiki. How we do it, and how we optimize it.



To optimize it, there are a lot of issues, like calculate color vertex only for visible vertex:



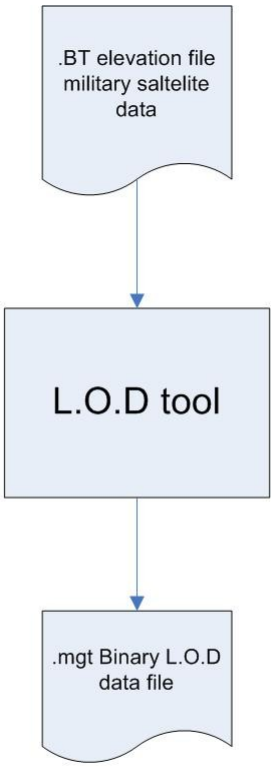
It stills some stuffs to do to accelerate this algorithm, but the next step will be to add clouds in the atmosphere.

pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

1.5 - Terrain's file generation

Loading the elevation map, spherise it and generate the levels of details is a very important part of the software we are working on.

In first we have worked on the loading of the elevation map. We decided to load ourselves a specific file format the ".bt" one and to convert it to a binary style file.



Our simple convertion tool

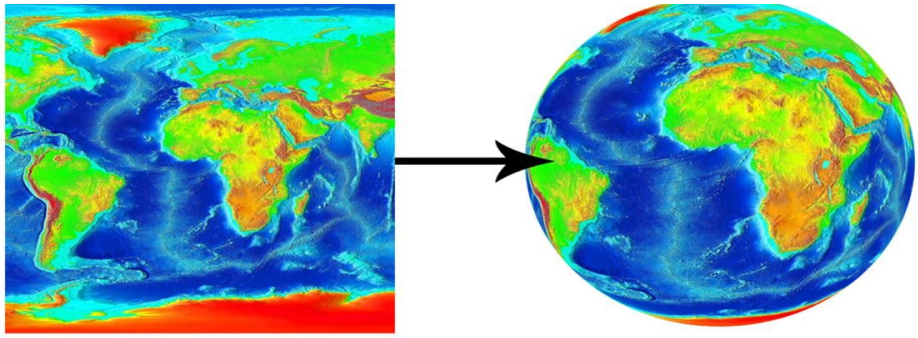
This file format is a military data format and contains satellite elevation data. We directly implement the LOD system on the data extracted from the file.

By the way it was the first manner to test the validity of the firs LOD displaying algorithm. After that we developed the spherisation formulas, the different aspects of passing form a "flat" map to a sphere is to:

- Look for alls the most different maps fundable over the web and search the type of the dominant projection style.
- Apply a reverse deformation to the vertex.
- Good management of the memory to process big and high level precision maps.
- Insure a realistic scale of the spherised result of the operation.

pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

The quadratic formula appears to be the most used model of projection.



The spherisation formula permits to transform flat topological to a spherical volume

After many tries we find a final formula:

For each final x, y and z coordinates Fx Fy Fz we can express with the original coordinates (Ox, Oy, Oz (given from the loaded data), the radius deduced from the total height of the loaded map, and the maximum values of Ox and Oy.

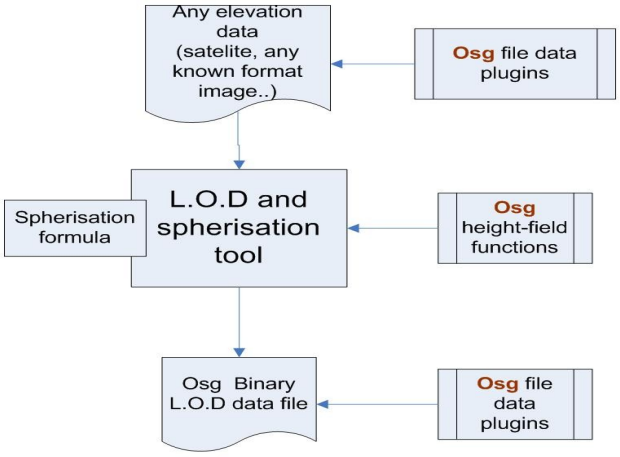
ry = (ry - (maxY / 2)) * 2;

Fx = sin(((Ox * PI) / maxX) * 2) * (sqrt((maxY * maxY) - (Ry * Ry)) / maxX) * Oz

Fy = cos(((Ox * PI) / maxX) * 2) * (sqrt((maxY * maxY) - (Ry * Ry)) / maxX) * Oz

Fz = Oy

We decided to change the output file to the provided one from OpenSceneGraph. By implementing OpenSceneGraph into our spherisation tool we took the chance of using other file formats of elevation file, and use the already made contributions for OpenSceneGraph existing plugins.



OSG provide plugins and data treatment to open our tool to others users of osg engine

pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

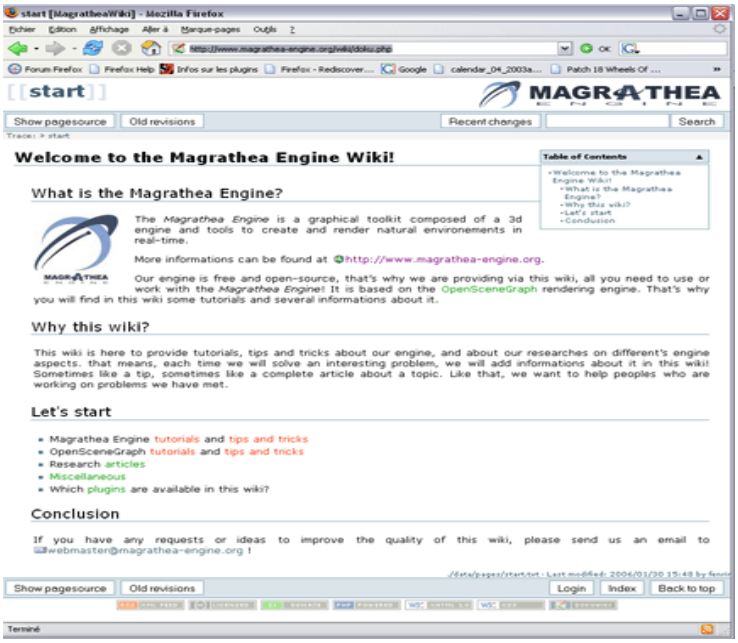
1.6 – Web development

The website : http://www.magrathea-engine.com



On this website you can find informations about the Magrathea Engine, what is it, screenshots and videos made in the development, libraries and binaries of the Magrathea Engine, documentations and links.
This website is also the main way to present our work and make people be interested by it.
This website is developped in php. The provider is one.net.

The wiki : http://www.magrathea-engine.org/wiki/

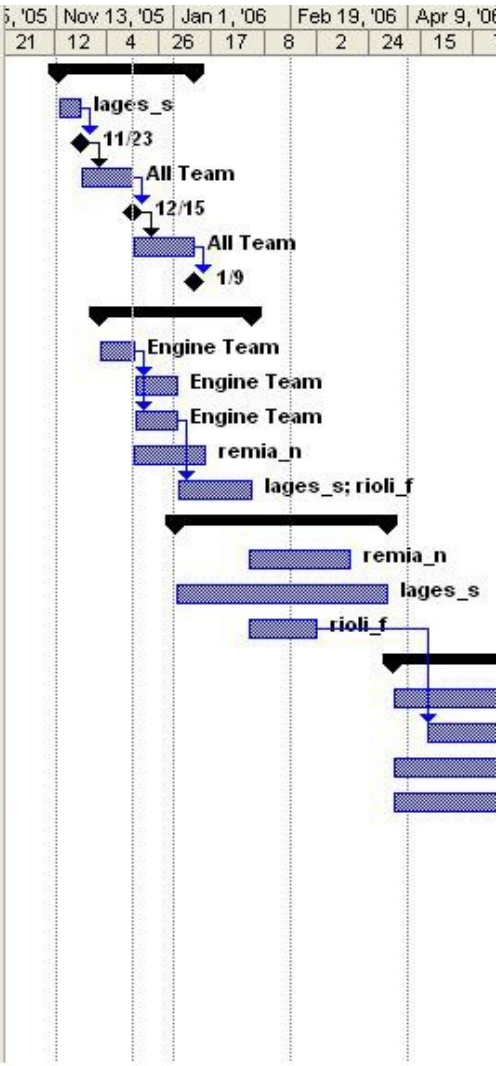


A wiki is a perfect way to provide articles, tips, and tutorials. For our wiki, we used the open-source wiki: DokuWiki.

pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

2. Engine Gantt

	Task Name
1	**Validation Phase**
2	Redaction of the project sheet
3	Validation of the Project Sheet
4	Redaction of the detailed etud
5	Validation of the Detailed Etud
6	Redaction of the specifications
7	Validation of the specifications
8	**Research and preparation phase**
9	Formation on the used technologies like OpenSc
10	Development of the general engine's structure
11	Definition of the terrain's file format
12	Development on the streaming engine
13	Development on the terrain's LOD algorithm
14	**Core development phase**
15	Starting the atmosphere and the weather's file f
16	Development on the terrain's procedural genera
17	Definition of the vegetation's and fluid's files for
18	**Rendering development phase**
19	Development on cloud rendering
20	Texturing and starting vegetation rendering
21	Starting development on fluid's rendering
22	Development on engine specific's edition plug-in
23	**Optimization and finalisation phase**
24	Development on weather effects
25	Advanced vegetation
26	Working on fluid's flow rendering
27	Optimization and testing Phase
28	Edition plug-ins functional
29	Tuning



done
working

pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

3. Tool progress

3.1 - Formation and preparation

Formation on the used technologies like QT and OpenSceneGraph:

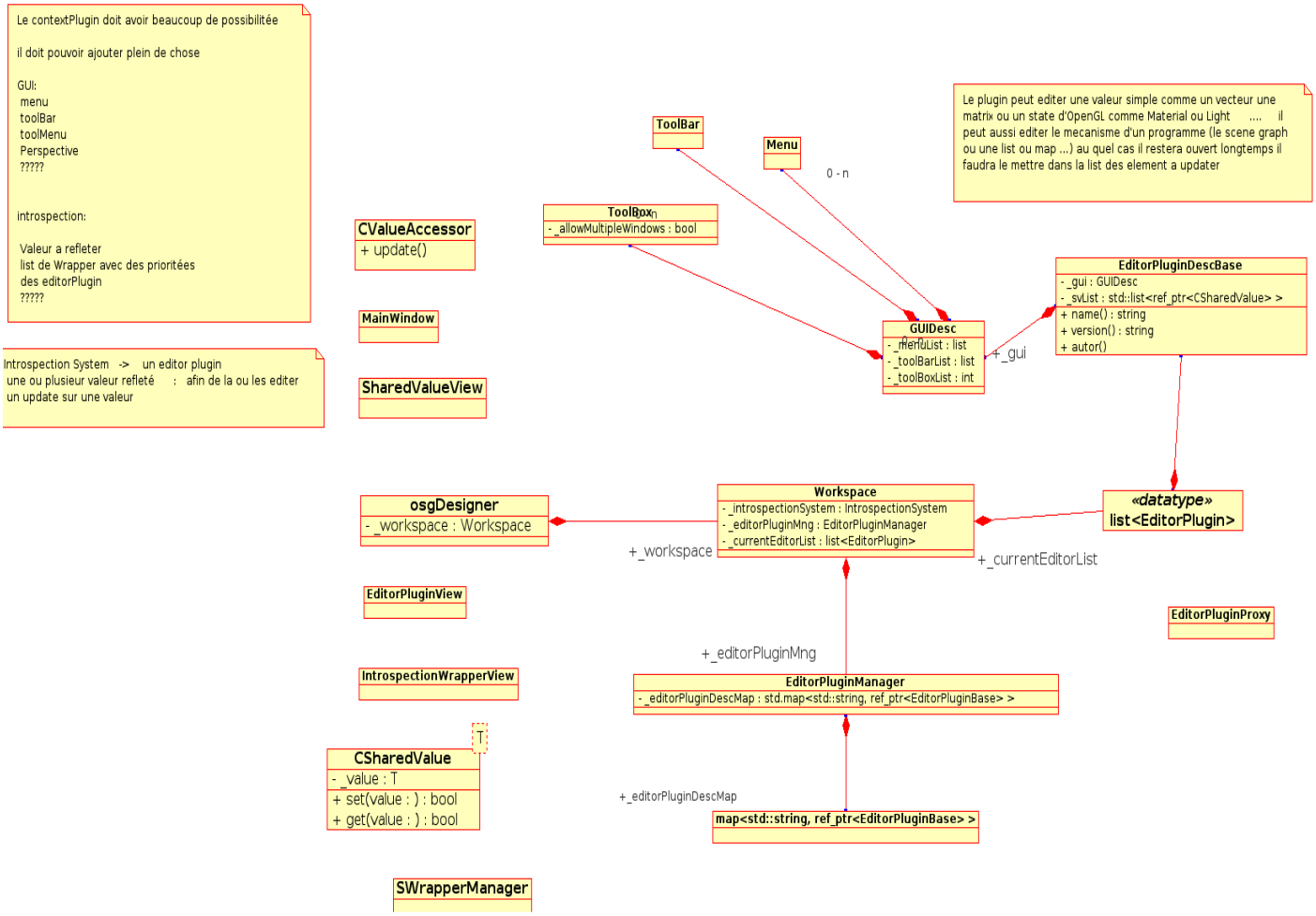
First we had to install our working environment. OpenSceneGraph works well on Windows XP and on all unix platforms. QT works well on all unix platforms too but for Windows XP pro it needs the MinGW compiler.
The project is based on a complex technology named Introspection recently integrated at the engine OpenSceneGraph, it was hard to learn, so we waste time here.

A big part of osgDesigner used QT technologies and nobody of the team have already used it. So we had to learn how it goes. The Qt documentation support by Trolltech helps us. (www.trolltech.com)

Design UML:

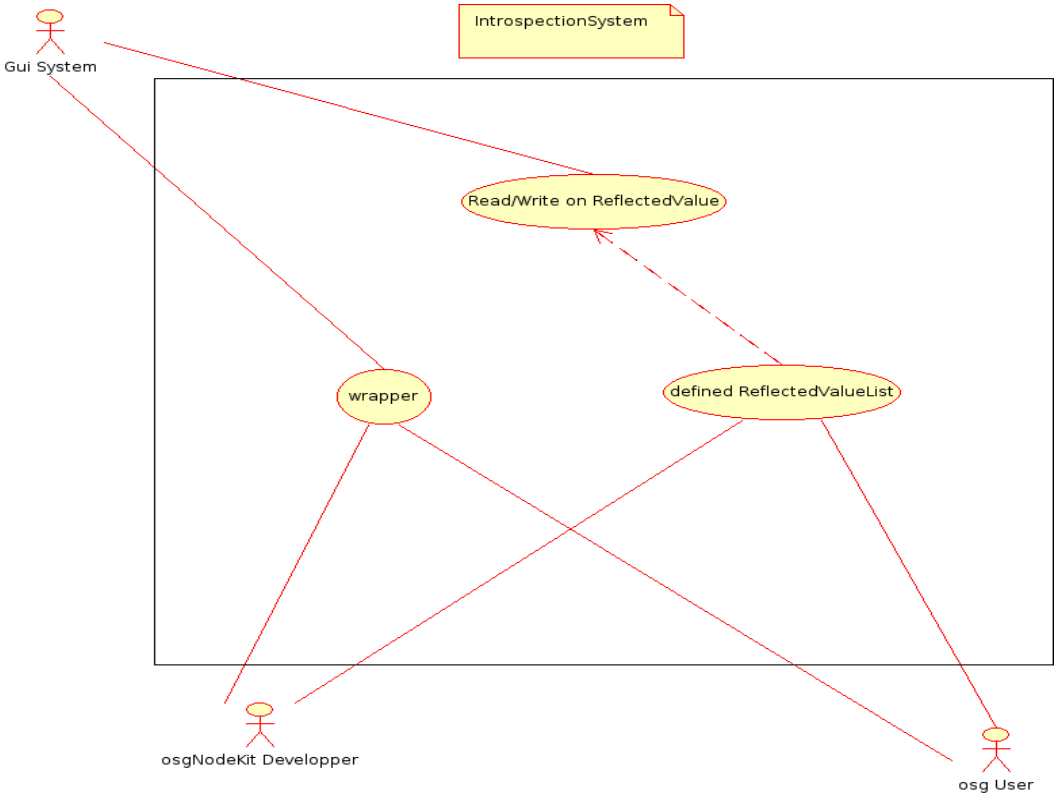
In order to win time, we wanted to design our project via UML. So we did several class diagrams like:

- a general one

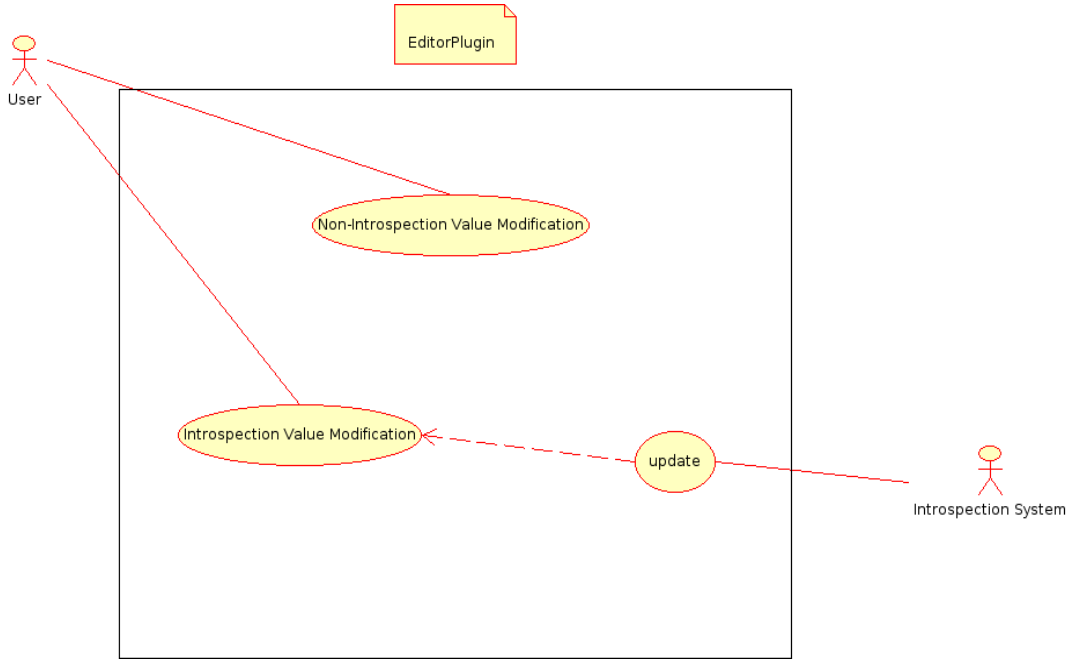


pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

- one for the Introspection



We also did a sequence diagram to design the Value updater.



pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

Reliability study:

In the first time, we would make an editor where each OpenSceneGraph users must write a ContextPlugin for do the link between his application and osgDesigner. After more reflection, we have decided to create the osgDesigner in a shared library, the developper juste need to load the code and instantiate an osgDesigner Class, do the modification, and if he have finish with the modification, delete the osgDesigner Class and unload the library. In a Real Time context the introspection is not a good idea and the fact to load and unload the osgDesigner code is very important.

We realised while learning QT 4 that we could use a new functionality of it: the Model/View architecture. So we begin a new class diagram for the mainWindow, with this archtitecture.

We also started to think about our PAQ and how will we provide quality.

A first architecture of the Web site has been placed on a wiki (wikiMedia). So anybody can complete it.

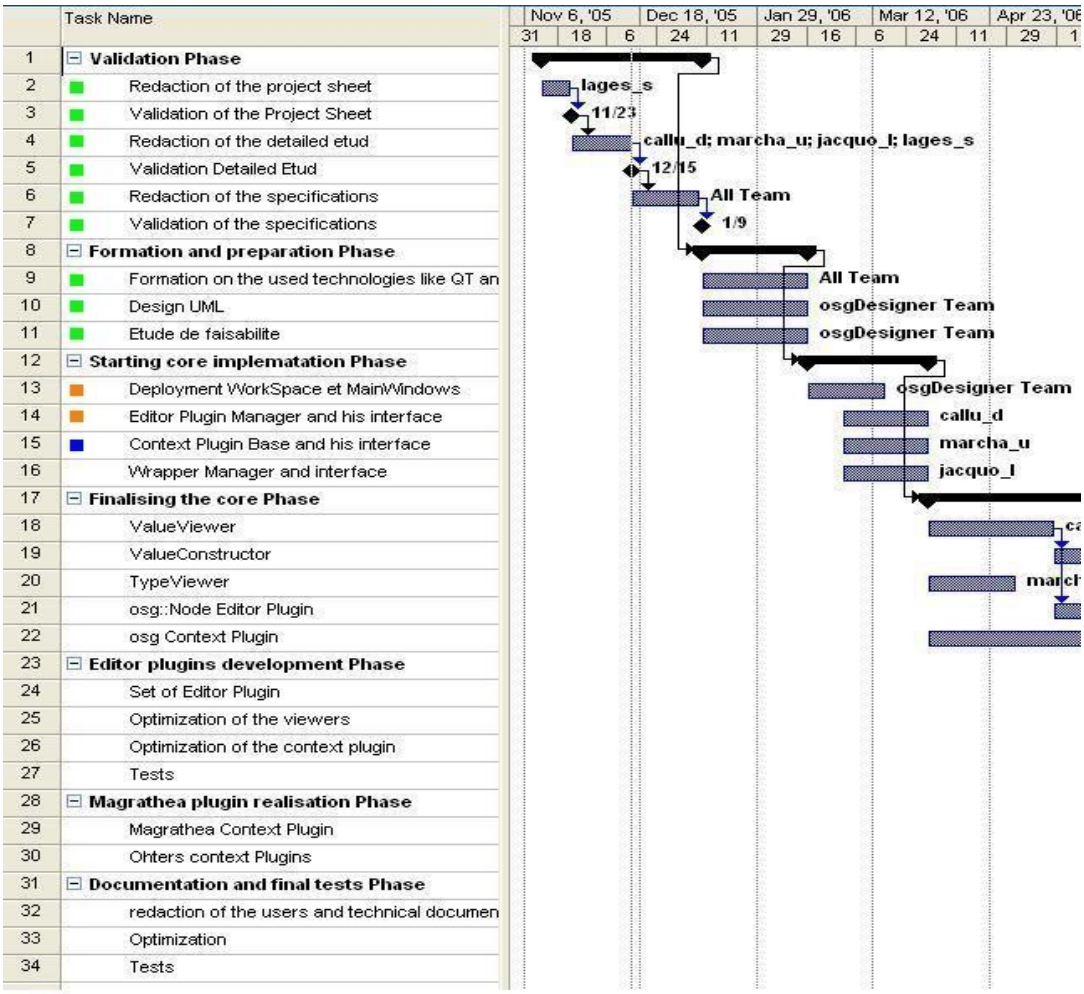
3.2 - Starting core implementation

Deployment Workspace and MainWindow:

The biggest part of workspace is the Introspection ToolKit (itk), after the first design and the first implementation, the design have been modified and the second implementation is in progess, it is probably not the final API but the global architecture don`t change in the future.
Implementation of the MainWindow started, but it's not finish yet. A lot of functionalities need to be developed. The class diagram needs to be redesign and validate by the team at the next week meeting.

pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU

4. Tool Gantt



pfe@epitech.net
Director : Flavien ASTRAUD
Technical responsible : Jérôme LANDRIEU