1st Oral

From

Team

Romain Biessy Renaud Gaubert Aenora Tye Erwan Vasseure

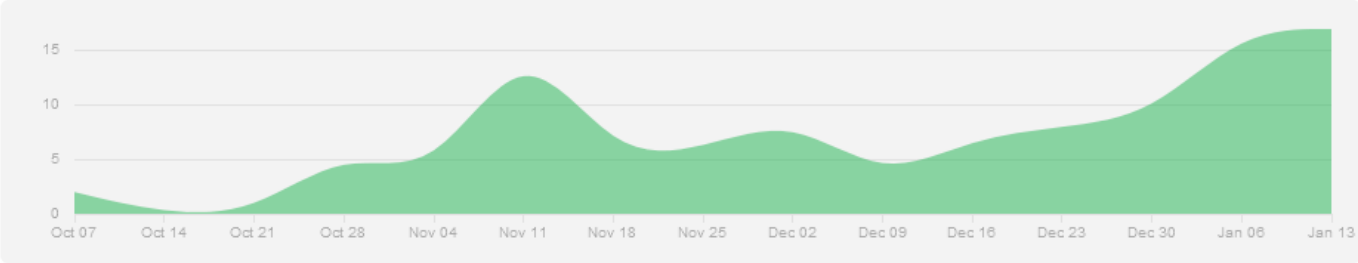
Contents

Chapter 1

Introduction

An ambitious 3d game such as ours requires a lot of work, to give you an example we spent not less than 100 hours each on the project since October also our project actually counts 120 000 characters and 15 000 lines of codes with not less than a 100 files.

This game also required a constant load of work from ourselves :



Of course this would be meaningless if we wrote bad code, even if we don't have the pretension to say that we write good code we have the pretension to say that we do note write bad code.

Throughout this report we will try highlight the fact that our work has payed and we already have a good base to proceed without worrying about the development of our game.

Chapter 2

Scripts

2.1 First script

As time went on, we thought of a better scenario than the one elaborated for the book of specifications. And we came up with two scenarios of our own :

During the second half of the XXIVth century war has been declared between the human race and the Kabool empire.The whole universe has become a large battlefield, not providing any safe shelter. As war goes on, the Kabool empire has finally taken over the Gaia systeme, your homeland and the very core of the universe. They plan to destroy the Genesia, a purelight gemstone that keeps the universe balance.

If the Genesia were to be destroyed, the whole system would blow appart and create a black hole that could end up swallowing everything. The gemstone is kept on a secret planet named Utopia, made from flying islands floating thanks to the mystic power of the Genesia.

As the threat over it grows larger, the islands are beginning to crash due to lack of the power from the Genesia being taken away. You embody a legendary soldier send to Utopia to retrieve the gemstone under the enemy eye and chase the Kabool empire away from the planet.

You will have to conquer the planet, island after island by developing a colony with your soldiers and discover new technologies that allow you to travel from one island to another and will help you win again the Kabool and save the universe. You finally land on the planet, and here your journey begin.

2.2 Second scipt

In the second part of the XXIVth century, the World Confederation, the galactical government faces a high amount of criminality from every planets of the federation.Some planets soon became populated by rebels and pirates.

You are a miner from the planet Mangaria 2, sent to work into a mine on a crystal planet. The flight was supposed to last one year, but after 4 months and 15 days the alarms started to ring, a pirate ship started to attack you, you ship was lost, and you barely managed to escape with an escape pod. Your pod was programed to go to the nearest planet; the pirate planet.

Chapter 3

State management

3.1 What have been done

The state management system has been implemented by Romain. The idea is simple and efficient. There is an abstract class State which can basically be started up, updated and shutted down. Then two classes inherit State : MenuState and GameState with their own attributes. The most important class is the StateManager, it has a stack of State which can be easily popped and pushed from other classes using reflexion - we just have to specify the type of the state to the StateManager in order to push a state. Each frames, the StateManager updates only the first State of the stack so that the other States aren't destroyed but are liked paused since not updated.

This system has the advantage to structure the code. Besides, it makes the change from one State to another very easy.

3.2 What must be done

There isn't much work anymore for the state management. All we need now is to implement some other States such as an OptionState. Then it could be launched from any other existing State

Chapter 4

Menu

4.1 What have been done

For now the menu is represented by three buttons, ”Play”, ”Exit” and ”Options”. The play button will simply launch the terrain generation and after a few seconds put us into the character skin. The exit button does what you think it’s doing. The options button does nothing for now. The background has been made into Minecraft since it’s our graphic model; and our game is not fit enough for the task.

4.2 What’s next

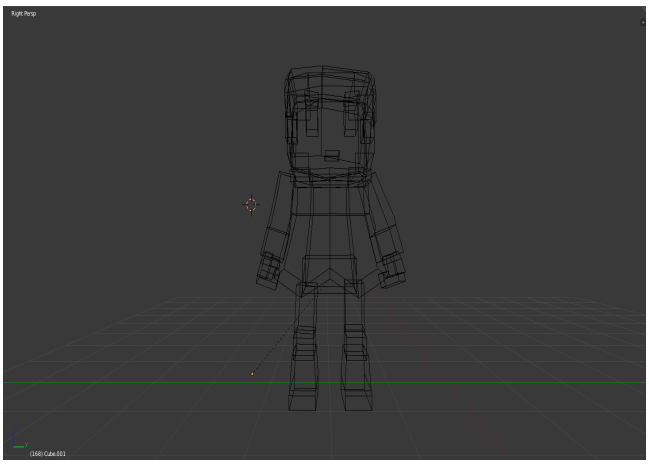
33 Our goal for the next oral is to make a functional option button, with few options like, the screen resolution, full-screen-windowed mode, the vertical sync, the language choice. We will also add an in-game menu.

Chapter 5

Character

5.1 Mesh

At the beginning we were using the Mogre[1] mascot called Sinbad. It was useful since it has predefined animations; the drawback is that it doesn't have the right size, and it isn't made for being integrated in a cubic world. After that Aenora and Erwan made a few different characters in blender, a 3d software. We encountered difficulties while trying to implement the new mesh file and its animations into the Mogre engine. Some of the character textures were made with the UV mapping technique.



5.2 Move

We can move our character with the directional arrows or the WASD keys. Animations start by pressing one of these keys. There are four animations; two for the walking loop and two for the idle loop. Erwan integrated the keys manager, inspiring himself from the MOIS manager provided by Mogre.

An issue Erwan encountered is the rotation of the player because we have a 3D world. If we needed some advanced rotations we would had to use quaternions, a powerful but non-intuitive mathematical tool similar to matrix in 4 dimensions. Fortunately an easier way is to decompose the rotation within the 3 axes (x, y, z). This is way simpler since we only need to rotations : yawing the player (around the

[1]Managed Object-Oriented Graphics Rendering Engine

y-axis) switch the horizontal movement of the mouse and pitching the camera (around the x-axis) switch the vertical movement of the mouse. Erwan implemented the rotation as well.

5.3 Animation

Romain integrated a fade for the transitions of animation. Basically, animations have a weight between 0 and 1, 0 is no animation and 1 is the full animation, 0.5 would be that animation, but with a lower amplitude. The fade is made by decreasing the old animation's weight to 0, while increasing the new one to 1. This method guarantees a smooth transition between all the animations even if they are stopped before their end in game.

5.4 Collision

The collision wasn't planed for this presentation but we have done the beginning. The hit-box of the character is represented by the 8 points of a parallelepiped rectangle. To test collision we look at the type of block around the character. For instance if we want to test the feet collisions, we get the type of the block under the character, using the 4 lowest points. For the next presentation, Romain will have to finish all the collisions test. The collision is still a little bit buggy and some work needs to be done. For sure, it will be fixed for the next presentation.

5.5 Camera

We can switch among two cameras in our game the first person one and the debug one. The first person camera is following the player, it just has to be pitch depending on the mouse. The debug camera isn't supposed to be seen by a player, thought it is very useful for us since it's a free camera detached from the player. Thus, we can see the whole world and the player with its animations. They have been both implemented by Romain.

5.6 What's next

Once the collision is fixed, we would like to implement the jump of the character. It's an essential feature that any FPS[2] game should have.

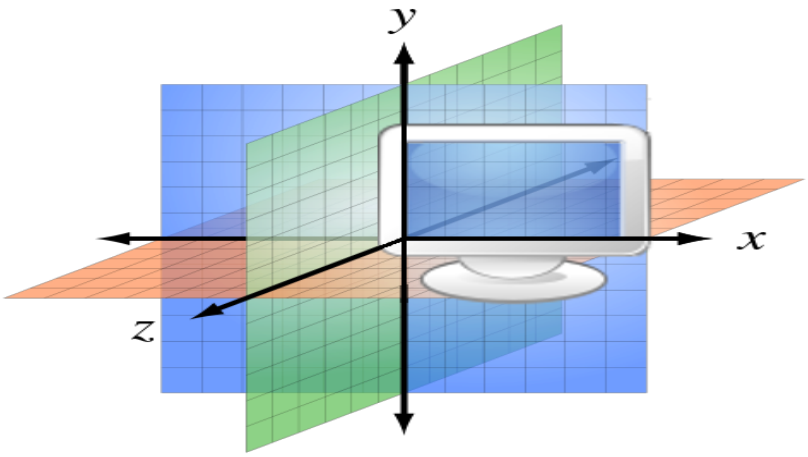
[2]First Person Shooter

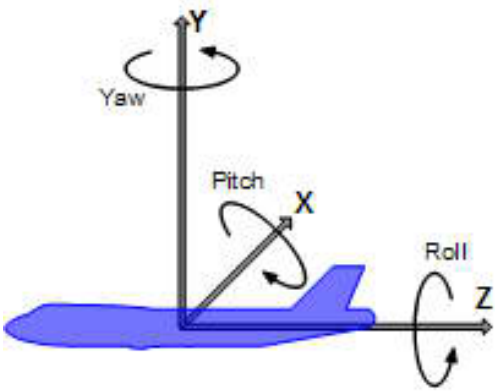
Chapter 6

Graphic engine and terrain

6.1 Ogre3D

A 3D world is at first sight more complex than you could imagine.The player moves via an FPS (Fisrt Person Shooter) type camera. While its position in space is given by a Vector3 (3 floats). The orientation of the camera uses rotations around the X and Y axes: this is called respectively the yaw and pitch (the roll, rotation around the Z axis is not useful in our cases).



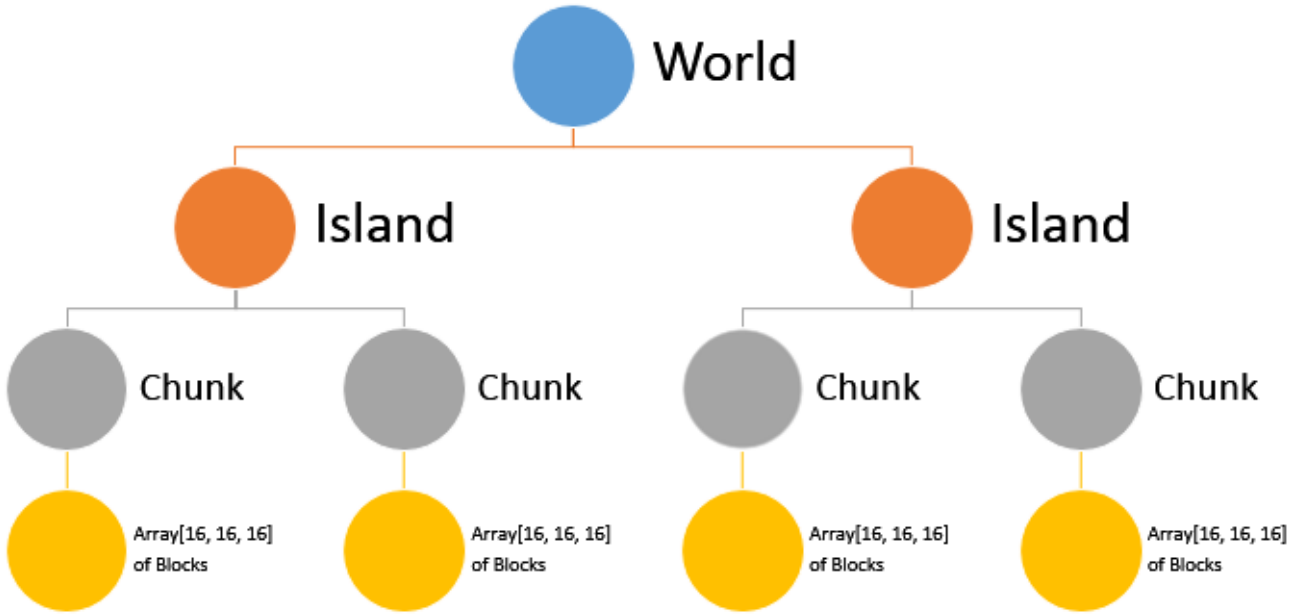


A trick to find if the rotation you want to apply is a positive rotation or a negative rotation is the following :

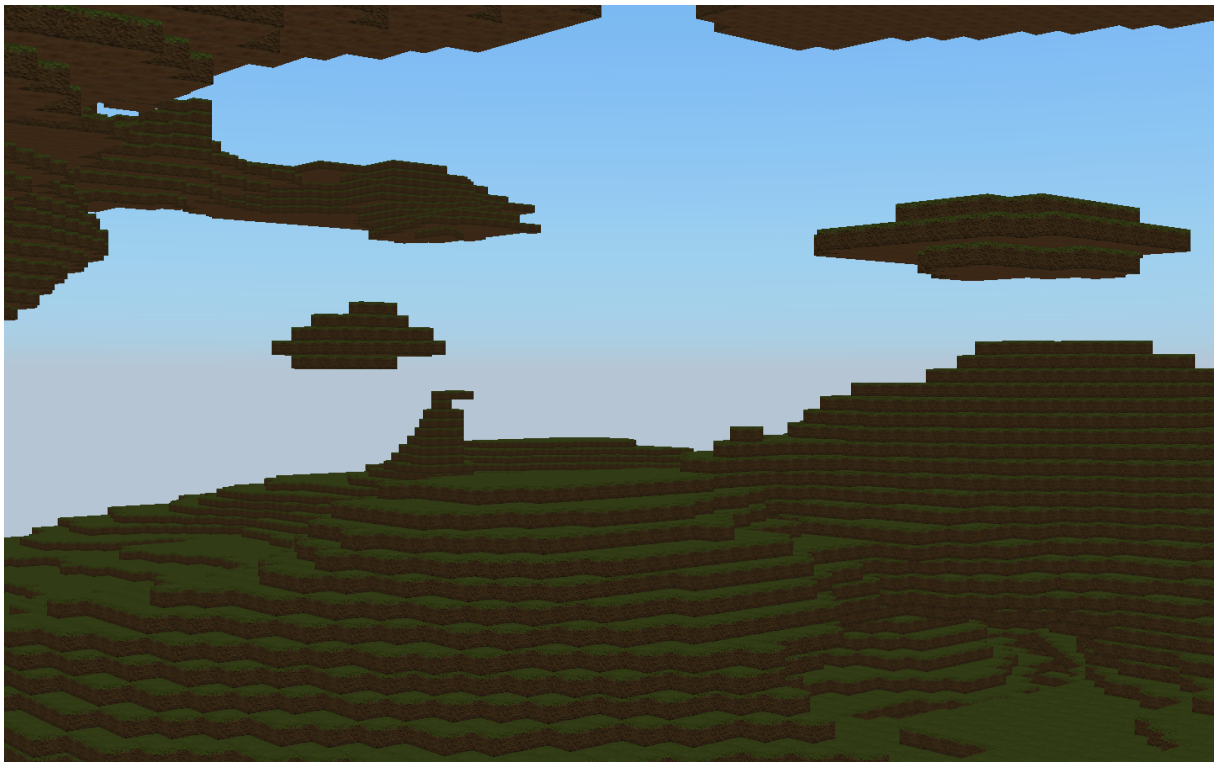
Using your right hand as a guide: point your thumb in the direction of an axis, curl your remaining fingers. The direction of the curl matches the positive rotation around that axis

6.2 Terrain Generation

When creating the terrain architecture we had the goal to separate the terrain in multiple arrays which we called chunks. Thus, when we will be able to save the terrain, we won't save it in a single file. Also this permits us to later, dynamically load the terrain chunks by chunks and not as a single entity.
The terrain Hierarchy is as follows : World contains a few islands which themselves contains a dictionary of chunks which contains a 16 * 16 * 16 array of blocks. It follows this chart :



The terrain itself is generated when we create an island, using a fairly complex perlin 3d algorithm. The basic idea of the terrain generation is to go through all the blocks in each chunks of an Island, and with it's position, using a 3D array of perlin noise, we compute the Block's noise value and following it's value we choose the block's type : air or grass.



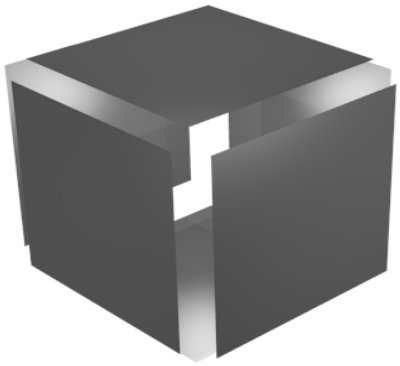
A randomly mountain generated

A dome with random trees

6.3 Terrain Display

Of course, displaying blocks where grass is, would've been to easy ! Indeed, even displaying visible blocks (to the player) cause a lot of lag, most of the time, with this method, the Frame Per Second count was below 1.

Thus we had to imagine an other way to display blocks. The solution we have chosen is the following :

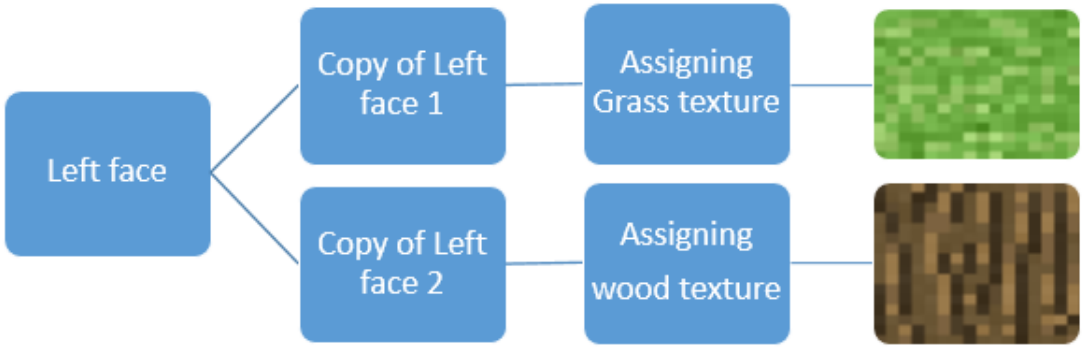


We could have imported directly a cube from a mesh file. But we needed to have a specific texture for each faces. That's why we created the cube directly in our code face by face. The principal benefits are the fact that we can change the size of the faces from one variable.

However, this still puts a large amount of entity on the screen, providing a number of FPS near 30. A good FPS count would be 60.

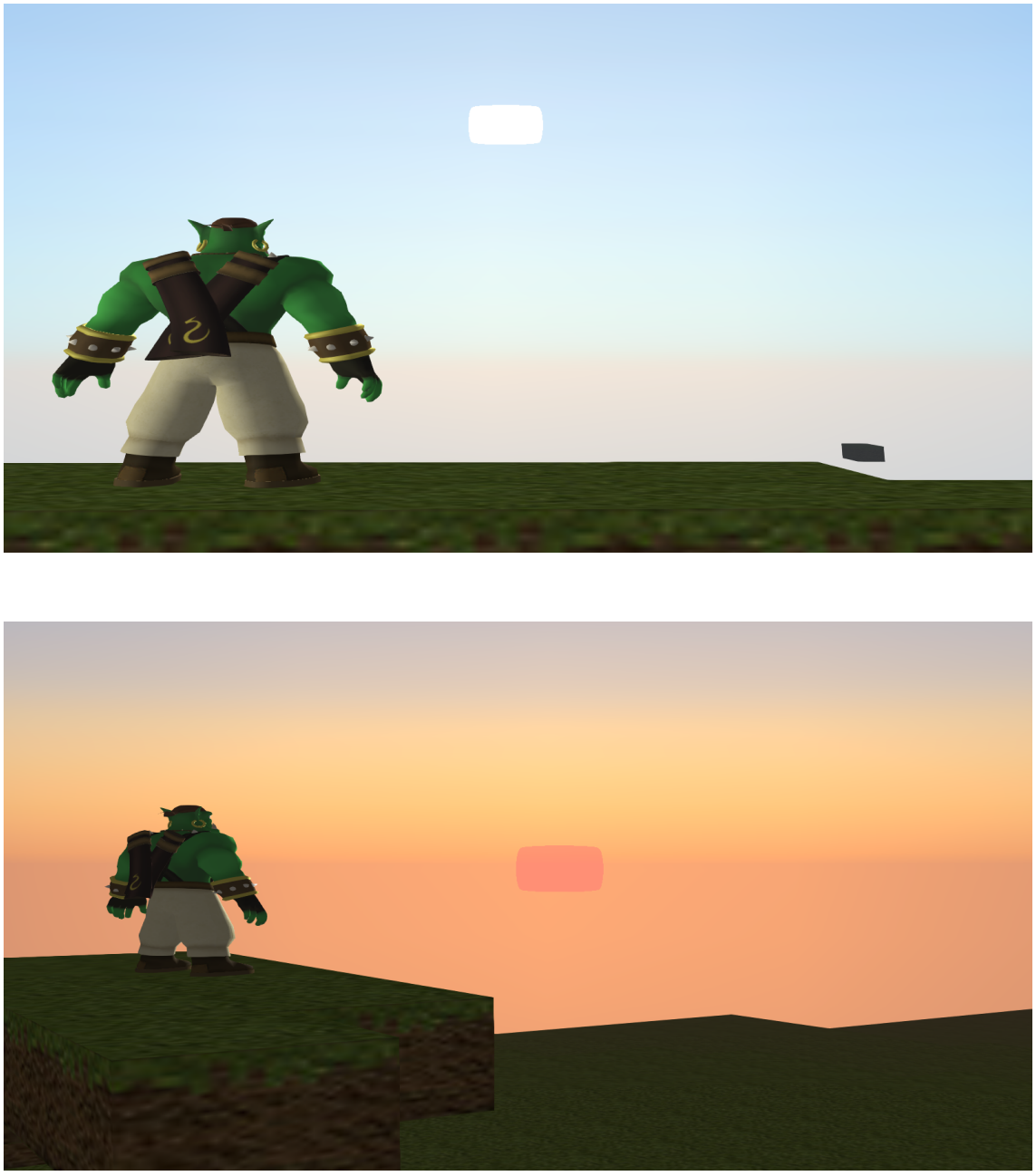
We exploded the cube into faces

We exploded the block in 6 different faces. The basic idea is to create in the code the faces, copy them and assign them different textures :



6.4 Sky

The sky is generated by an open source library : Caelum. The main difficulty was to build the lib ourselves related to the actual version of Mogre we're using. It has been done by both Erwan and Romain. Then we just had to configure it.



Dusk time on SkyLands.

This lib is very useful, it automatically handle the position of the sun, the moon and the stars. Moreover it position the stellar objects according to the actual date and hour. We can easily speed up or down the time with the mouse wheel. Note that it won't modify the speed of the character.

6.5 What's next

As we've already said, displaying faces has a major impact on the FPS count, thus the goal for the next oral presentation will be about having an other way of displaying the terrain (Vertex) or displaying not just faces but 'multi-blocks faces'. Either way, we need a better FPS count.

Also you might have noticed that the terrain's borders seems to have been cut as if they were part of a bigger terrain. For the next oral we will have to cubically speaking round those borders.

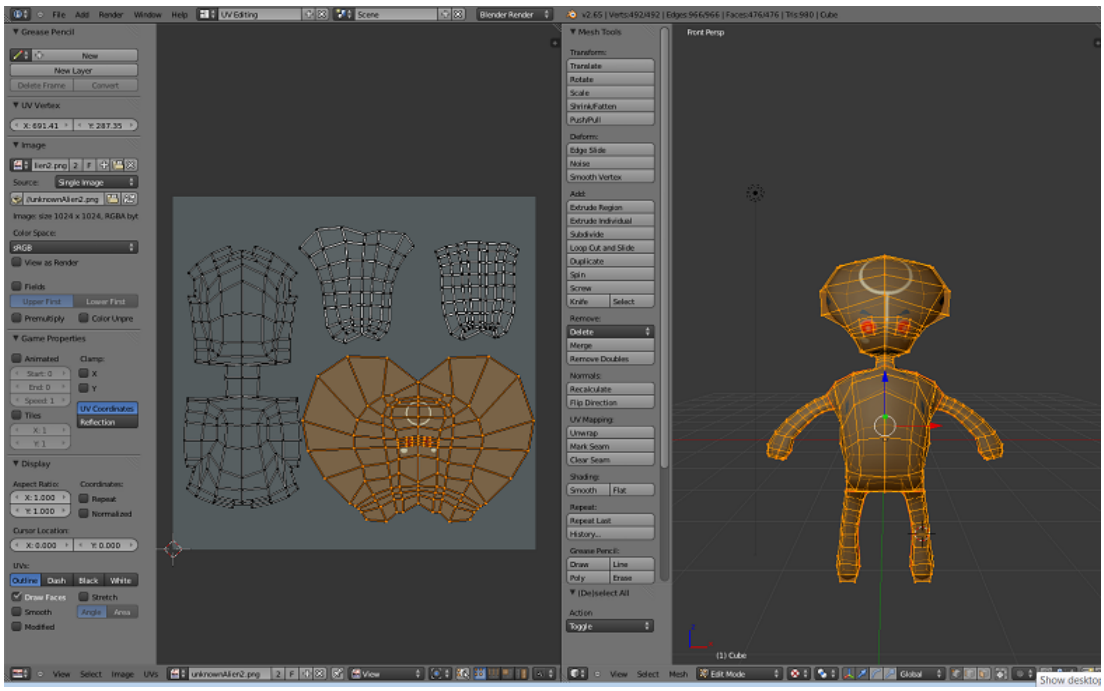
Chapter 7

Graphics

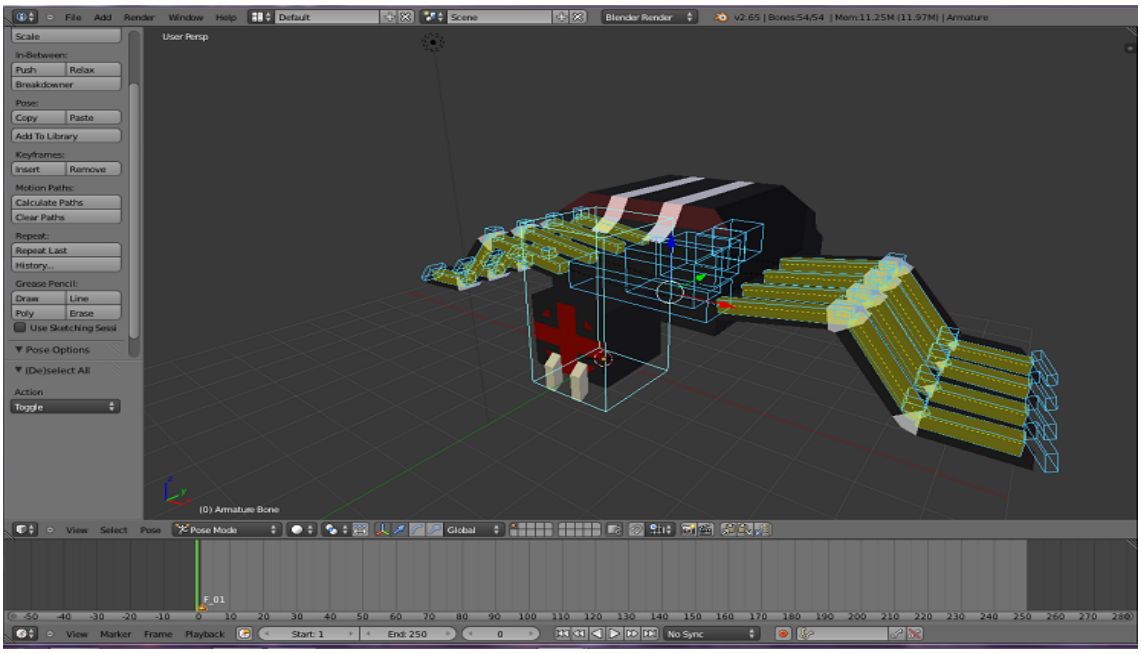
7.1 Software choices

7.1.1 Characters and 3D model:

To create the main character and enemies we used a 3D modeling software named Blender, that Erwan and Aenora already used in the past years. Moreover, it is a free software and easy to use thanks to the large amount of on-line tutorials. Blender provides a broad spectrum of modeling, texturing, lighting, animation and video post-processing functionality in one package. Through its open architecture, Blender provides cross-platform interoperability, extensibility, an incredibly small footprint, and a tightly integrated work flow. Despite being a free software, Blender provides the user an almost professional render. Edit mode lets you design the shape of your 3D model starting from a cube, a sphere or even a monkey head, then using UV editing mode, you can design texture for your shape using paint or photoshop or other similar programs.



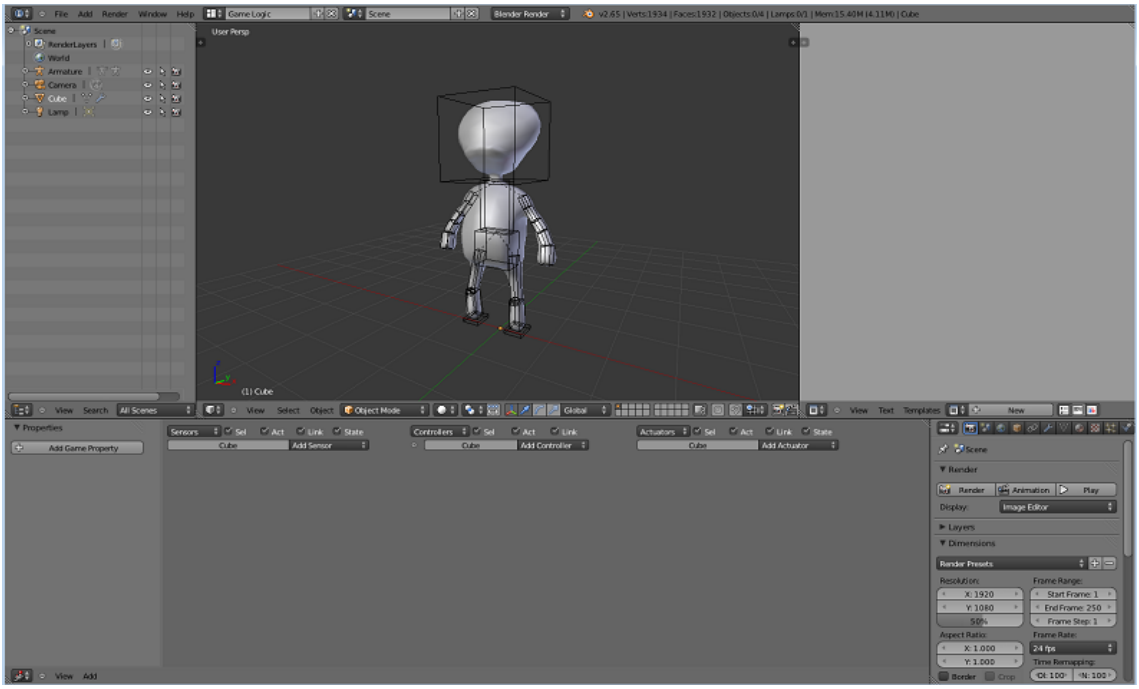
You can place bones on your model to give it a structure and then animate it through the animation mode.



Then link key-frames with the animations using the game engine.

7.1.2 Texturing and illustrations:

As for the pictures and textures Aenora used photoshop since she already worked with it last year. The textures are 16x16 pixels images so that it takes the smallest place as possible. There are at least 20 different textures for all the different materials.



We finally fully created a player, enemies, weapons and 3D structures and animated them using blender.

Chapter 8

Website

8.1 PHP and JS

As explained in the book of specifications we are using the framework Symfony2. Which provides us a flexible OO interface. Of course symfony is not WordPress and we still have to work and code all the PHP but it provides useful components such as : easy URL rewriting, Templates, Form components, and automatic cache.

As for the data storage Symfony2 uses Doctrine which uses a traditional SQL base and adds a database abstraction layer to the database. Thus with Doctrine, tables are class.

As for the JavaScript, we used jQuery which is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animation, and Ajax interactions. Thus, we could made animations in a few lines when using plain Javascript would've taken us hundreds of line.

Even thought we used a couple of library to simplify our work, keep in mind that they do not do the work for us, we still have to work with them to create the website.

8.2 Design

When creating our design, we wanted a maximum compatibility with the different browser (yes even IE 8 !), and even though the design isn't the same with a recent browser (such as Opera, Chrome, Firefox, Safari) than with an older browser which does not support CSS3, the deign does not appear ugly !

Using the new CSS3 properties and images created by our group members, we built our website by hand, during the winter holidays. As you might have guessed it was no easy task. Only one member of our group had done web coding before and was not a CSS expert.

However, the result we came up with was surprisingly good looking :

Chapter 9

Conclusion

For this oral presentation, we've reached the goals we fixed in the book of specifications. Actually we did a bit more than we intended to do at first : Perlin 3d generation and Physics which required a lot of work !

Our game's future seems bright, at the very beginning, when we thought the game, we believed that our project might be a bit ambitious. However seeing our current progress in the game, boosted up our moral and we are now convinced that we can finish this game !

Still we better not lower our guard, the work we still have to do is enormous , and now that the base is done, we must not rest on our laurels.