PRESENTATION OF THE THEOLITH PROJECT

www.theolith.com

A role playing video game in 3 dimensions developed in C++ with the DirectX 9 SDK.

4th September 2010

Entirely produced by Richard GESLOT (www.richard-geslot.com)

1- INTRODUCTION

In order to go beyond a simple letter and resume, I would like to present to you a video game that I have produced. I did it alone, during my spare time. Of course, I will limit this presentation to a brief study of a few pages so that it can be read in a few minutes. This makes no claim to be a document going into the technical details of programming 3D real time effects.

2- GENERAL PRESENTATION OF THE GAME

Theolith is a RPG and I have drawn much of my inspiration from the graphics of the Warcraft universe. The game has about 25000 lines of code. It has been produced in C++ with the DirectX 9 SDK.

I urge you to watch videos on theolith.com which show the strengths of my game. You will also find a playable beta version. Some items listed here are not present in this beta at the time of writing this presentation. But these are real functions which have their source code and work properly, only needing further debugging and being correctly incorporated into the story of the game.

Here are the highlights of the game that will be developed in the presentation:

- I have developed in parallel **a map editor** to allow easy creation.
- **Dynamic characters**: thanks to the possibility of assigning script in the map editor.
- The player may loot the corpses of his victims to improve his equipment. My mathematical knowledge allows me to establish a balance between the bonus and the level of the object. Creatures have an object board with their probability of occurrence.
- All data is encrypted with my own technique, even textures and 3D shapes. Depending on the types of data, I use either a system of indexes in the file's header or a linear storage. The overall aim is to optimize the balance between storage size and loading time.
- An **Interactive interface** with a system of movable windows.
- **Day/Night in real time** with management of shadows.
- A powerful **particles system**: adding and setting up an effect is very easy.
- The possibility of piloting planes, cars, boat through the presence of a real **physics engine** respecting Newton's laws.
- A quest system (running on the same idea as the famous World of Warcraft).
- Ability to control multiple heroes, switching from one to another.
- Heroes can send animated spells thanks the particle engine.
- Landscape is composed with 3D shapes that can be imported from Warcraft III or Starcraft II with a program I created that converts formats of these games to an encrypted architecture only readable by Theolith.
- Advance rendering techniques such as **vertex and pixel shaders**.
- A **deferred rendering** engine (video of 100 dynamic lights on theolith.com).

Having used OOP to make this game, I propose a study of each object. Here are the main objects of Theolith:

2.1- GAMEOFF

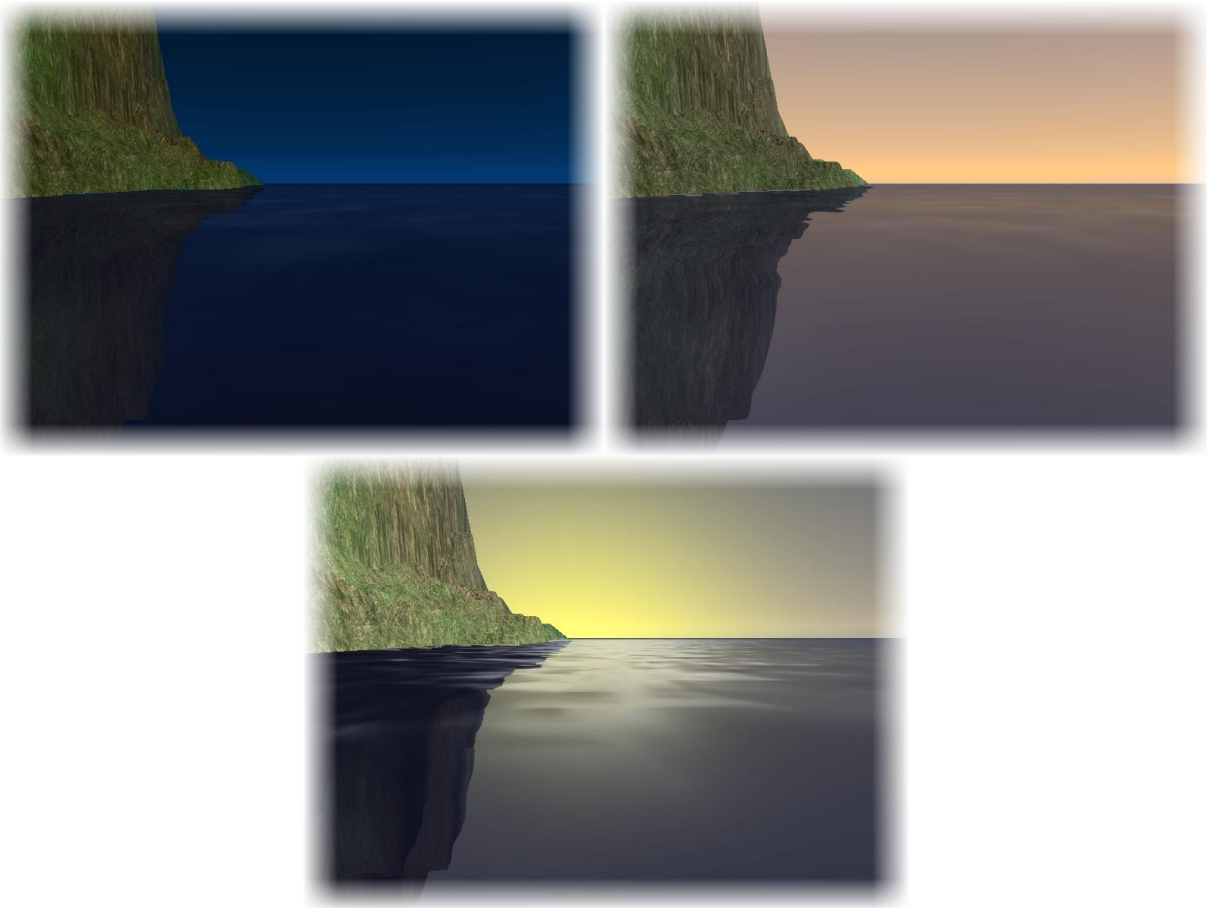
The Gameoff is the start menu. It is modeled on that of Warcraft III. The menu animation is pleasant to use: buttons light up when hovered over by the mouse and the window containing those rises and sinks when the menu changes.

The cave was made with the POV-Ray program (by cleverly placing 3 textured plans and applying a heightmap to them). The sea is of course in motion, turning red inside the cave and merging into a dark fog. Finally you will notice gray clouds moving quickly, as if to warn the players of a strong wind outside…

2.2- MAP

This object manages the landscape of the game. Initially, it will load terrain from a file of which I have defined the architecture. For rendering, I used the tiles technique because it facilitates computes of collision, memory storage and division of land. As shown in the picture, I used a system of transparent layers, which allows continuous variation of the vegetation. The land is fully managed through the Shader technology.

2.3- SKY



This object manages the sky. Firstly, I should make it clear that I have established a system of day and night in real time with the landscape's shadows adapting to the sun's position. I chose the model of the Skydome. Thanks to this, it's very easy to change the color of the sky in real time because I just have to change the diffuse parameter of all vertices by means of the Vertex Shader. This allows me, without loading any texture, to have a bright blue sky with a light pink line on the horizon gradually varying towards a dark sky with a light turquoise blue line at the horizon.

It may be noticed on the picture that the water uses advanced rendering techniques of distorted reflections of land and sunlight. These are achieved through the Pixel and Vertex Shader.

2.4- DOODAD

Doodads are all static objects placed on the map: trees, houses, grass… They can be directly imported from Warcraft III or Starcraft II because I developed another program that converts Warcraft III format (MDL) or Starcraft II format (M3) to an encrypted format that can only be understood by Theolith.

2.5- PARTICLE

In Theolith, you can meet many particles: spell effects, effects on the ground, the exhausts of flying machines… The large number of particles is due to an engine that I have created. With the call of subroutines together with some arguments, they can be easily create and destroy.

Technologically, these particles are simple quads, often drawn in billboarding, on which position and the 4 components (Red Green Blue, Alpha) can be varied.

2.6- CHARACTER: HERO & NPC

Character is a virtual class that is the source of 2 subclasses: Hero and NPC. Heroes are the group of characters controlled by the player. In Theolith, the player can control up to 5 heroes simultaneously by switching from one to another. NPCs (Non-Player Characters) are characters controlled by the computer with which the player can interact.

In order to have access to a lot of 3D animated characters, I import them from Warcraft III and Starcraft II. One of the most difficult things I encountered was to manage the various characters' animations. In fact when I extract them from Warcraft III and Starcraft II, I just have a simple binary file. This required that I study the animation of a 3D shape: the skeletal system, interpolation of translation, scale and quaternion matrices between each frame. I have published a sample of my source code that import Starcraft II models on this website: http://code.google.com/p/libm3/wiki/RenderM3

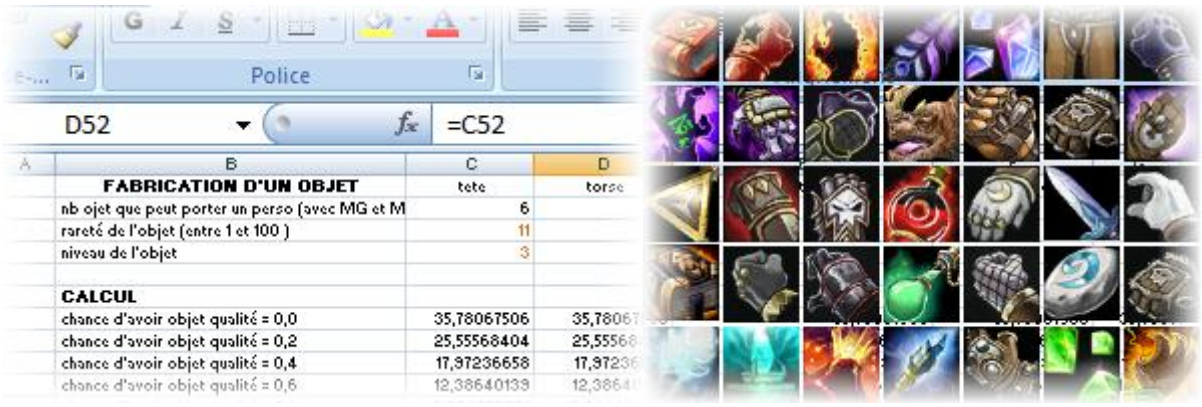
For each character, I can assign a script listing the actions of my choice. This method can make the game much more dynamic. During his quest, the player will encounter creatures that move, interact and fight each other... This makes a very lifelike world. But of course, every NPC has a default configuration that is to defend, attack, but also help a friend.

In Theolith, characters are not necessarily humans. They may also be machines. In the picture above, for example, you can see a boat and a flying machine. They are there to impress and immerse the player in the world. A physics based on Newton's laws is of course present to manage the movement of these weapons that the player, in the following versions, will be able to control.

Sound effects are very important for a game and are therefore present in Theolith. They make swords, spells launches… more living. An interesting effect to observe is the aircraft engine the frequency of which rises with the increase in speed.

Naturally, all is saved in files. The map editor writes on these files and the game reads them. A character has many parameters. Files dedicated to their backup possess intelligent architecture which manage to combine rapidity of reading (via indexes), storage size (the memory of a character does not have a fixed size) and security (thanks to an encryption algorithm that is efficient but does not change the file size).

2.7- OBJECT: OBJECTLOOT & OBJECTSPELL



The Object class is divided into two subclasses: ObjectLoot (objects looted from corpses or purchased from the store and usually causing no action) and ObjectSpell (objects causing spell casting).

The objects are the interface between the hero and the world. When the player wants to attack, he will use his ObjectSpell in order to kill the NPC before picking up from his body new items of equipment that allow him to improve his skills. He will also find items needed for his quests or salable items to make money in order to buy new, more efficient ObjectSpells. This cycle allows the game to have a long life, because the player will constantly seek improvement.

One of the challenges I have met but which is one of the strengths of a role playing game is achieving balance. The size of bonuses for the player or of penalties for the enemy should not be determined randomly. As shown in the Excel window above, I have done much research to balance up the price, quality and characteristics of the object based on the probability of its being obtained and the level of the NPC possessing it.

2.8- QUEST

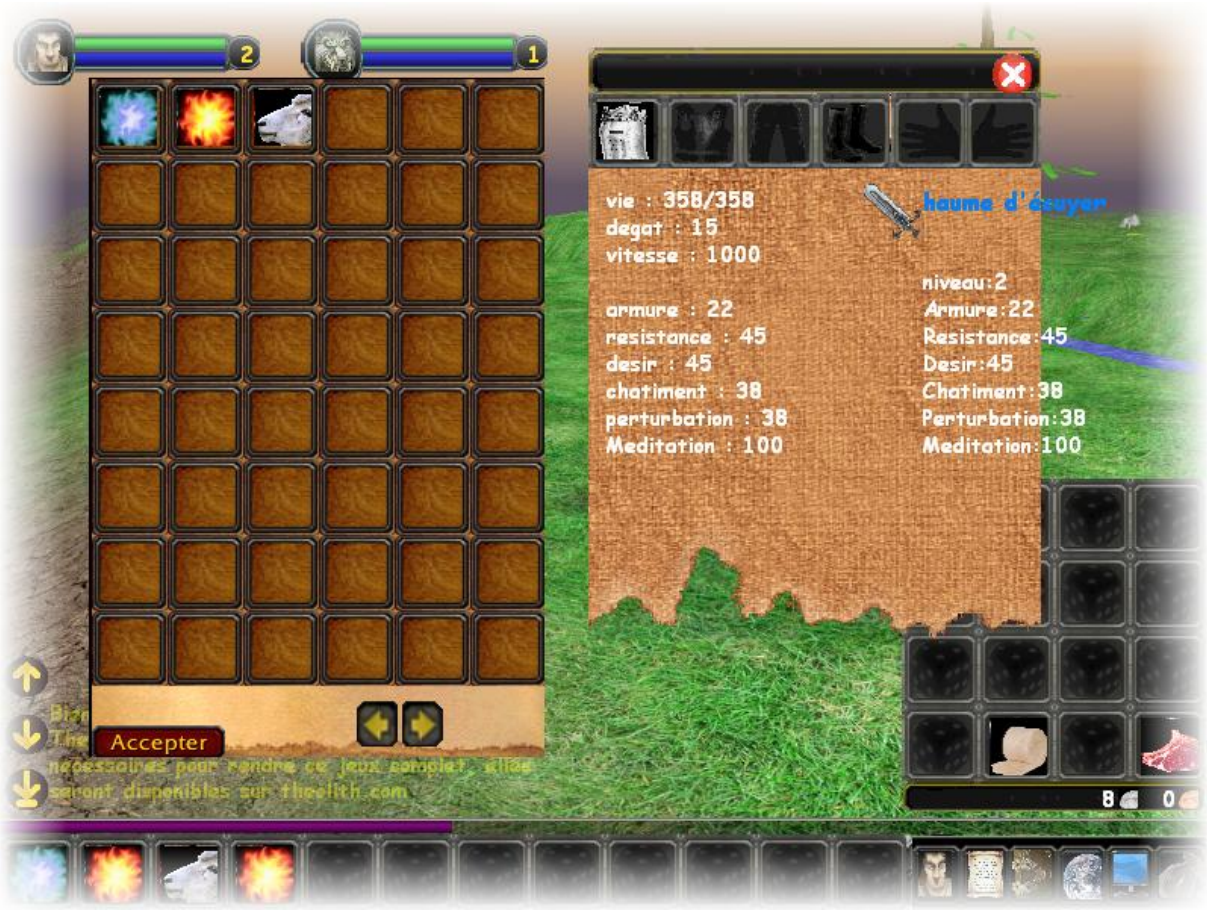
Quests work on the same principal as that of World of Warcraft. The player will meet NPCs with an exclamation mark on their heads, he can talk to them and the NPC will set him a problem, with a coherent story in the game. This will cause the player to explore unknown lands and to challenge his abilities. All his efforts will be rewarded with items, money and experience points. Of course, when the player leaves, everything is saved, both achieved quests and the current state of ongoing quests.

2.9- SPELL

The spells are there to give the game a magical dimension. For a better rendering, spells use the particle engine. As explained in the section about objects, the characteristics of spells increase with the level of the player. However, the equations are made in such a way that the higher the level of the NPC the greater the necessity for the player to send spells in a dynamic manner. This allows a gradual increase in the difficulty of the game.

Spells have been designed to require N successive steps, N increasing with the complexity of the spell. The most commonly encountered lay-out comprises a first step of initialization and verification: Is the Target an enemy? Is it alive? Has the player enough energy? ... If the first step is validated, we go to the second, the casting of the spell: The hero is immobile during the casting time and this time increases with the size of the spell. Next comes the step where the spell will move through space towards the target, often using particles. The spell reaches the enemy and the sub-routine associated with the action of the spell is called. Finally we arrive at the final step, the destruction of the spell and dumping it from memory.

2.10- INTERFACE



The interface of Theolith is very dynamic thanks to a complete GUI engine able to click on buttons, write in dialog boxes or move windows. But these capabilities do not stop there. In a role-playing game, the interface is certainly one of the most complicated elements to devise (unlike a FPS for example). A large number of tasks have to be accomplished by the interface and I prefer to present them in the form of a list:

Left click management:

- **Character selection**: To determine if the player has clicked on a character, I use 4*4 matrices: MatLookAt, MatPerspective and MatWorld. The program will compute the equation of the line starting from the eye of the camera to the mouse pointer and will examine if this line passes through the mesh collider attributed to the character.
- **Object transfer**: The interface controls the management of objects by the user. For example, an object triggering the casting of a spell cannot go into an inventory containing loot objects. On the other hand, a utensil purchased from a merchant can be placed wherever the player wants in his inventory.

- **Window management**: to move windows, click a button... All these actions allow the player to feel comfortable with the program.

Right click management:

- **Interaction with a character**: An action on a character could have many meanings, and it is the interface that will take care of this first layer, which is to decide, depending on the characteristics of the target, if it is an attack, a purchase, a quest request, etc.
- **Spell casting**: If the clicked object is a spell, the interface will create a spell.

3- MAP EDITOR PRESENTATION

However good a game may be, if it has no editor, developers will never be able to make their creation concrete. Therefore, before concluding, let me say a few words about this centerpiece. I spent a lot of time developing this editor along with the game. Thanks to this I have a simple means of testing all my ideas.

Here are its features:

- Paint the card by choosing the size and texture of the brush: grass, rocks, snow... You could paint several layers one on top of the other for progressive environment transitions, through a process of transparency.
- Add or delete a Doodad.
- Position particle effects, indicating their density and their lifespan.
- Add or delete a NPC (Non-Player Character).
- Editing a NPC to change his name, what he says, his aggressiveness, his faction ...
- Set the loot table of a NPC, associating it with objects and the chances that this object will appear when he dies.
- Adjust the heights of land. This is done from a black and white bitmap: a heightmap.
- Add areas of water.
- Manage quests offered by NPCs: add or remove them, set presentation and victory texts as well as rewards.
- Editing AI scripts that are associated with NPC s. For this purpose, the editor opens a dialog box where you can write lines of script code.

4- CONCLUSION

Thank you for having taken the time to read my presentation. I think this project shows my motivation and my ability to initiate and carry through a project down to the smallest detail. Moreover, it enabled me to gain a lot of experience in video games. I should now like to apply these skills at a professional level.

I remember reading an article listing the various jobs related to video games. Having created this game by myself has enabled me to acquire skills in each of these areas:

- 2D engine developer (because of the interface).
- 3D engine developer, of course.
- Physics engine developer (because of vehicles).
- I have also held the architect's role in dividing my program into objects.
- AI programmer by making scripts and behavior for NPCs in their dealings with enemies and friends.
- Special effects programmer with my particles engine.
- Sound management.
- I also did very substantial scriptwriting work in writing my story and creating the world of Theolith.
- Marketing Manager by improving the presence of the game on the internet.
- I was tools programmer by creating my map editor.
- Data programmer by creating optimal and encrypted architecture and data backup.
- I put into practice one of the qualities of the graphic artist which is to establish attractive and consistent graphics using the minimum of resources.
- And of course, I have had to fix many bugs and this has certainly made me a very good debugger!

Theolith is a project that is important to me and I intend to continue working on it. To follow its progress, I invite you to visit its official website: www.theolith.com.

I wish to thank the people responsible for the websites cppfrance.com, magosx.com and gamedev.net from which I learned a lot. Thank you also to Laurent Testud and Wolfgang Engel who wrote very interesting books about real-time 3D.

Richard Geslot