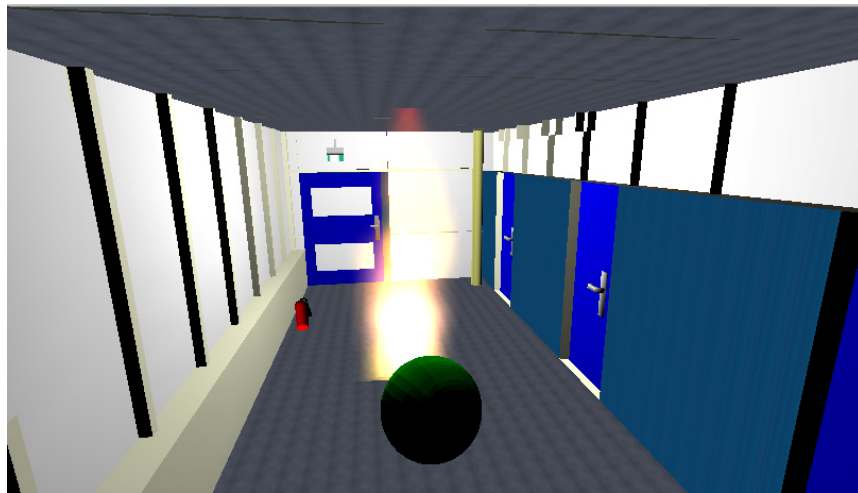


Serious Gaming



27th June, 2007
Stephanie Rowlinson



Index

| | |
|-----------------------------------|----|
| Index..... | 2 |
| Abstract..... | 4 |
| Acknowledgements | 6 |
| Introduction..... | 8 |
| Related work | 10 |
| The Delta3D engine..... | 11 |
| Kaneva | 13 |
| Blender Game Engine..... | 14 |
| Modelling and Implementation..... | 16 |
| The Delta3D engine..... | 16 |
| The Kaneva engine | 20 |
| The Blender Game Engine..... | 21 |
| Conclusion | 24 |
| Appendix A..... | 26 |
| Appendix B..... | 30 |

Abstract

This paper is a report of the assignment given to me during my internship at the TU Delft. The assignment was to:

- 1) research different game engines
- 2) to simulate a fire in the corridor of the 12th floor of the EWI building at the TU Delft using one or two different game engines

In order to complete this assignment I first researched useable game engines and decided to try a total of three game engines.

- The Delta3D engine, which has been developed in order to be used for modelling and simulation games. This engine uses C++ as its main language and has been developed in Microsoft Visual Studio .NET 2003, though it is also capable of running on Linux. It comes with a particle editor, a viewer (for 3D objects) and STAGE a 3D level editor. Games cannot be run from the editor, for this you must write separate code using the Delta3D API.
- The Kaneva engine, which has been developed in order to make quick and easy development of Massive Multiplayer Online (MMO) games possible. It comes with a world editor from which games can build and run. All the networking tasks have already been dealt with and the development kit even comes with an out of the box game server to test the games on.
- The Blender Game Engine, this game engine is an integral part of the Blender 3D modelling software. It has been designed for easy use by people who have little to no programming experience. The engine uses logic blocks to get things to work, though further game logic can be added by using Python scripts.

I then proceeded to try these three engines. First of all a model had to be build of the corridor on the 12th floor. Due to my lack of modelling skills, this became a model of only part of the 12th floor corridor. The next step was to start programming using the game engines mentioned above.

The Delta3D engine proved to be difficult to use. Its documentation was incomplete and outdated, though the support forum somewhat compensated for this. After having made a basic scene, in which a camera can be moved through the model, any further attempt to add functionality failed. This was due to a configuration error, the cause of which was never found. The Kaneva engine turned out to be the wrong kind of engine for me, as I was not building a MMO. The Blender Game Engine was user friendly and in this I finally managed to make a working demo.

All in all it must be concluded that if the goal is to develop a MMO game, the Kaneva engine is very well suited for this task. For the development of a serious game that has network play, it is better to use a more professional engine as the open source engines require a lot of extra knowledge to get them to work. If the development team is experienced with the required programming logic and the theory and practice of things such as network programming, using the Delta3D or Blender Game engines shouldn't be too much of a problem. Otherwise go for a more developed and better documented engine.

Acknowledgements

My acknowledgements go to Leon Rothkrantz and Mohammed Abd El Ghany for their support and help with the assignment. I would also like to thank the various people on the Delta3D and Blender artist forums who helped me out when I was stuck. Wouldn't have been able to do it without you.

Introduction

In the last few decades computers have increasingly become a part of people's lives. Where at the start of the last century nobody even knew what a computer was, nowadays the average mobile phone has more computing power than the first computers did. Computer games in the form of simulations have been around almost as long as computers. As early as 1947 an interactive missile simulator was created by Thomas T. Goldsmith Jr. and Estle Ray Mann. The game was inspired by the World War II radar displays and used a cathode ray tube as a display and analogue circuitry to make the necessary calculations.

As computers made their way in to people's living rooms so did video games. Where at first games had been written and played only by people who worked with computers, games now became commercial entertainment products. Nowadays there are few people left among the younger generations who have not at some point played a video game. The immersive nature and realistic graphics of modern videogames has led to a lot of criticism from parents who fear that their children are being negatively influenced by these games.

However, it has also made a lot of people think about the educational value these games could have. With so many of their students playing games, teachers have been looking for ways to incorporate games in their lessons. This has led to some schools setting up buildings in Second Life and giving lessons there. Though not everybody agrees with this. The virtual world it offers is not free of controversy, nor is it purely educational.

The TU Delft has come up with a proposal of a purpose build virtual university. This university would allow students from different countries to interact with each other and listen to lectures from teachers of different universities. The virtual campus will not only serve an educational purpose. A lot of focus will be put on social interactions between the different users. It will also be possible to generate events, such as fires or other disasters, and use these to test the knowledge that was gained through lectures about these subjects.

There is another kind of serious game. This type of game is usually referred to as a simulation game, as it simulates real life situations. As mentioned above these games have been around for years. As terrorist threats increase and the scenarios for disasters become increasingly bigger and more complex, the need for emergency services to practice how to respond to different situations has increased. However, real life simulations are becoming more and more difficult. For instance, how do you simulate a nuclear disaster or a flood? You can of course use things like smoke machines and a few proper make up artists help to create a sense of realism, but there are a lot of things that have to be imagined, like fire and water. This is one of the reasons why not everybody takes the exercises seriously. There have already been several incidents where passers-by have disrupted the simulation.

All in all it would be easier, safer and cheaper to do these kinds of exercises in a virtual world. With the current physics and graphics engines it's possible to create virtual, yet realistic worlds. Of course in these worlds you can "really" blow up the nuclear power plant. You can then try out your latest emergency plan and see how well it does or doesn't work. If it's a failure then you can reset the game and start over. There are already several programs that allow the user to do just that.

The problem is that the different emergency services all have their own programs. So the fire brigade can practice extinguishing fires, but their practice doesn't concern itself with situations such as whether the ambulances can get through or if the police has managed to secure the area. It also doesn't simulate the coordination needed if multiple fires were to break out simultaneously. So the bigger and more complicated scenarios described above can't be simulated with these existing programs. The Dutch ministry of internal affairs has proposed that a program, which will satisfy all these needs, be developed. For this purpose the Decis lab and the TU Delft have been contacted. Decis have already made a demonstrative program in Second Life. However, the ministry was not very impressed by this. So the TU Delft is looking at the possibility of using a different game engine.

My assignment for my internship was:

- 1) to research different game engines
- 2) try to simulate a fire in the corridor of the 12th floor of the EWI building at the TU Delft using one or two different game engines

To accomplish this, I have first of all researched several game engines and identified three candidates: Delta3D, Kaneva and Blender Game Engine. I then tried to get the test program as described in point two to work. This being the best way to see if the engines were indeed suitable.

Related work

A game engine is the core software component of any game. The game engine provides functionality which enables easier game development. Without a game engine every game would have to be written from scratch. Functionality usually provided by game engines includes: a 2D or 3D rendering engine for the graphics, a physics engine or collision detection, sound, animation, A.I., networking, scripting, streaming, memory management, threading and a scene graph. By providing all these things the engine allows the game developers to focus on what they want to achieve. Choosing the right game engine can save a lot time and money.

In order to complete my assignment I tried three game engines: the Delta3D engine, the Kaneva game engine and the Blender game engine. All these engines are free for non commercial use, though the Kaneva engine is developed by the Kaneva company and is not open source.

Each of these engines has a different approach to game development. The Delta3D engine focuses on modelling and simulation games. The Kaneva engine has been build to allow easy MMO development and the Blender engine focuses on 3D modellers who want to start making games. These different approaches lead to three very different game engines, which will be discussed below in the following paragraphs.

The Delta3D engine

The first engine is the Delta3D engine. The engine has been developed by the American Naval Postgraduate School. Its focus lies on making serious simulations games that can be used for things like fire fighter training. The engine has been developed in Microsoft Visual Studio, but was written in C++ and is there for cross platform. Although only Visual Studio .Net 2003 is officially supported, there have been quite a few people who have managed to get it running under various distributions of Linux.

The Delta3D engine comes with several tools to help develop a game. These are the particle editor, the viewer and STAGE. The particle editor allows you to make simple particle based objects, such as fires or smoke. The viewer does exactly what its name says: it allows you to view 3D objects that you want to use in your scene so you don't have to create a new scene and run a test game every time you want to see what something looks like. STAGE is the most important tool you get. The name stands for Simulation, Training and Game Editor and it's an editor that allows you to build your world. You can use the standard actors such as skybox and camera or add your own actors. These actors can be assigned 3D meshes which represent them. However, you can't run a game from the editor. For this you must first load the map into game, this is done by coding using the Delta3D libraries.

If you want to get anything done in Delta3D you have to write a piece of code to make it happen. Ready made methods are provided to aid you. For instance:

```
dtCore::RefPtr<dtCore::Object> mText;  
  
mText = new dtCore::Object("Text");  
  
mText->LoadFile("HelloWorld.flt");  
  
GetScene()->AddDrawable( mText.get() );
```

will create a new drawable object, i.e. an object that can be drawn in a scene by the Delta3D render engine. It then assigns a file, in this case a 3D object of the type .flt, to the object and finally adds the object to the scene. (This can also be done by using STAGE, you then only have to load the map created by STAGE.) However, what the above code does not do is add a camera which allows you to view the scene, nor does it give you any control. This illustrates both Delta3D's strength and weakness. You can do just about anything with this game engine if you write the appropriate code, but on the other hand even the simplest of programs requires quite a bit of coding. This isn't really a problem, except for the fact that you're using somebody else's engine. This means that the methods were written by other people and whilst the use of these methods might make perfect sense to them, it's not always clear to the user.

Added to this is the fact that the Delta3D engine is an open source engine, which means that the documentation is optional and largely dependent on volunteers. As a result the tutorials are often outdated and incomplete. The API is also a mess, a lot of function or class descriptions are blank or merrily state what they do, without giving any explanation as to how this happens. For example a constructor may require five variables which have somewhat mysterious sounding names. The tutorial doesn't explain which variable does what, just gives you a bunch of values to punch in, so you decide to look in the API. However, you find that the API also only states that these variables are required without telling you what they do. The

support forum is the best place to get your 'why' questions answered. The engines developers are also present and their expertise is a great help.

The engine is geared towards first or third person simulation development. So it comes with a well balanced physics engine and an out of the box first person shooter collision model. Network play is supported, but there is little documentation available on the subject and though an MMO is a theoretical possibility, all the modules to support it will have to be written by the developer. The Delta3D engine incorporates other open source projects such as, Open Scene Graph for the graphics and OpenAL for the sounds. These modules have been integrated in the API, which means that if you want to draw something in a scene you first have to learn how to use OSG. The differences in syntax make it difficult to integrate the OSG components with the Delta3D ones.

All things considered this is a powerful engine which is not suited for beginners or people with little patience. A lot of the problems you come across are solvable, but require a lot of previous knowledge about things like network programming. As the engine is still being developed there are many modules which are buggy or incomplete. Thus, this engine requires a lot of previous knowledge which I for one do not have.



Figure 1
A simple program made by using a tutorial

Kaneva

This is an engine which I haven't had much time to look at. The engine is specifically geared towards massive multiplayer online games and handles things such as login menu's for the developer. The hosting and the network side are covered by the engine. The engine comes with an editor which is fairly easy to use. As mentioned before, Kaneva is not an open source project. Rather it has been developed by the Kaneva company, which also has its own MMO social simulation/game which was developed using the platform.

The goal of the game engine is to allow starting developers to make MMO games more easily. The engine is free until you start to use it for commercial purposes. Kaneva then collects an licence fee of \$50.000,- and also collects 50% of the net fees generated by the games you have developed using their platform.

Because the engine has been developed by a company and with the above purpose in mind, it is highly user friendly. It was made with fps and rpg games in mind and as such it comes with ready made modules for NPC's, bank areas and a quest system. The result is an easy to use editor with which you can create a new world, including environmental effects such as fog, with just a few clicks. The world is also highly customisable as you are free to import your own models and many variables. For instance the density of the fog can be customised. The company also states that you can script/program using Lua/Python and the C/C++ API. I haven't had a chance to try this out yet.

The documentation appears to be in order. The beginner's tutorial I tried was easy to use and correct; however, I haven't had a chance to look at the further documentation yet. One thing that must be noted though is the rather obscure website design. It's not easy to find the information you need.

This engine is very well suited for all your MMO development needs, but even though there is a licensing option for stand alone games, it's really not suited for the purpose of my assignment. As it requires little previous knowledge it's ideally suited for hobbyists and other people aspiring to create the next World of Warcraft.

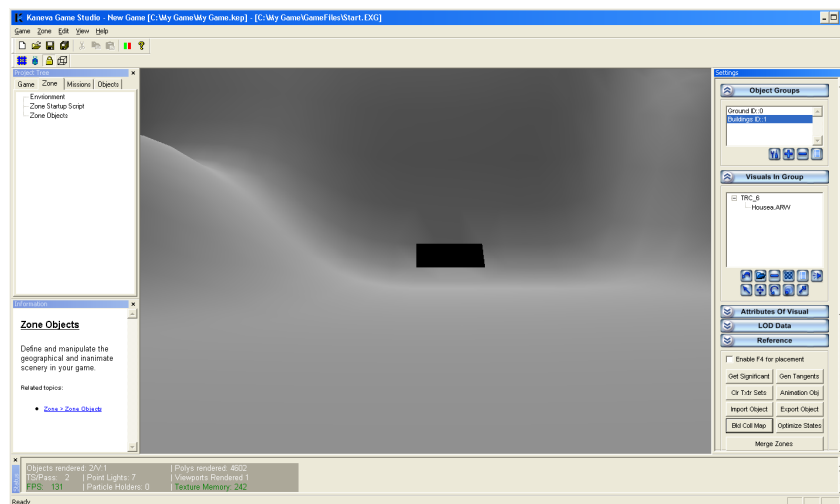


Figure 2
The Kaneva editor in action

Blender Game Engine

Blender is a well known open source 3D modelling program. It was originally an in house for a game development company, so the game engine was a feature from the start. However, when after the bankruptcy of the original company the source code was released the game engine took a back seat to the 3D modelling function. It wasn't until version 2.41 that a stable version of the game engine was included again. The Blender game engine uses the Bullet Physics Engine for rigid body dynamics, ragdoll and vehicle physics. Unlike Delta3D these functions have been integrated in a way which requires no previous knowledge of this engine by the user.

As Blender is still primarily a 3D modelling program, the game engine is aimed at 3D modellers who want to use their models to create games, but who have little to no programming knowledge. Therefore the game engine uses so called logic blocks to add logic to the game. There are three types of logic blocks: sensors, controllers and actuators. More complicated games require the use of Python, which has been fully integrated with Blender. The logic blocks can be set to respond to bits of Python scripting to create the necessary game logic.

The use of these logic blocks is what makes Blender such an easy platform to use. In order to make a sphere move forward, you only have to add a sensor block for the ↑ arrow, a AND controller (which is always triggered when there is input from only one sensor) and a motion actuator with the dLoc variable for the y-axis set to 0.50. If you want to make the sphere move as if it is actually affected by the laws of physics, you just make a dynamic actor of the sphere by pressing the appropriate button.

The physics engine is realistic enough to be able to create a serious game using the Blender game engine. The fact that you can instantly use the models you have created in your game is also a big plus. Due to the popularity of Blender it doesn't suffer from the lack of documentation as much as Delta3D does, though the documentation that's available for the game engine often assumes too much prior knowledge of Python to be comfortably used by someone who is still fairly new to Blender. The ones that don't use Python usually don't cover more than the bare essentials. It's also much harder to find documentation for the game engine than for the other functions of Blender. Another drawback is the lack of network modules. Though these can undoubtedly be written and made to work, it again adds to the workload and it requires a certain amount of expertise to be able to pull it off.

The engine can be used for the development of all kinds of games. Though with the edition of a specific Vehicle Engine, aimed mainly at the handling of cars, it would seem that Blender is mainly suited for the creation of racing games.

The Blender Game Engine is an easy to use and versatile game engine. The logic block system makes it very well suited for use by 3D modellers, the user group originally targeted by Blender, as these people often have little to no programming experience. The Blender Python API provides an easy way to add to the functionality of these logic blocks. However networking modules appear to be absent, making this engine unsuitable for the simulation games that were proposed in the introduction.

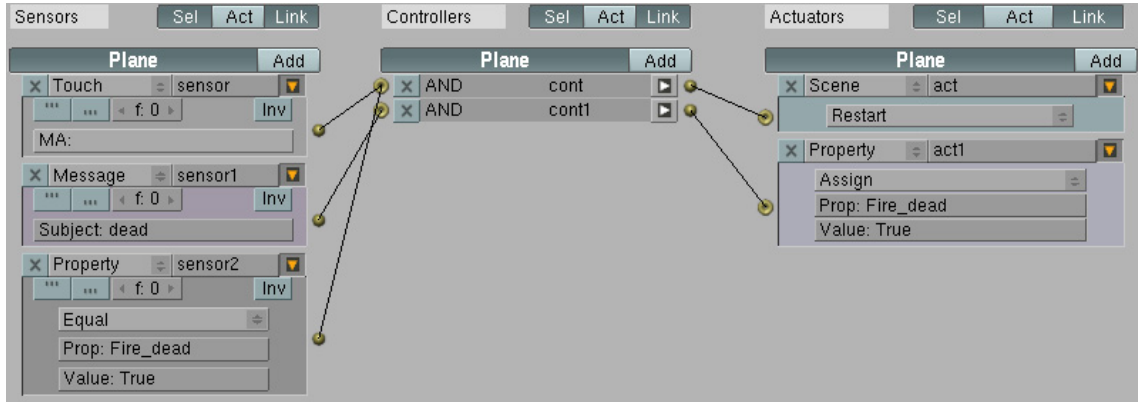


Figure 3
The logic blocks in Blender

| | | | | |
|--------------|----------------|--------------|-------------|-------------|
| Actor | Ghost | Dynamic | Rigid Bod | No sleeping |
| Do Fh | Rot Fh | Mass: 1.00 | dius: 1.000 | Form: 0.40 |
| Damp: 0.040 | RotDamp: 0.100 | Anisotropic | | |
| Bounds | | | | |
| Add Property | | | | |
| Del | Bool | Name:Prop | True | False D |
| Del | Int | ame:MyHealth | 100 | D |

Figure 4
The properties and actor buttons in Blender

Modelling and Implementation

In order to develop a game there are several steps that must be taken. First there must be a concept. This concept needs to be turned into a set of requirements which the game must meet when it is completed. The game itself consists of two things: the look and the feel, i.e. the graphics and the game play. In modern game development these two things are usually the work of different departments. So I decided to divide my game development into two separate parts as well. The first part was making a realistic model of the environment and the fire that would have to be added. Without the model there was little sense in writing the program. The second part consisted of adding the functionality. I settled on using Blender for the model and the three engines mentioned above for the functionality.

Based on the assignment I formulated the following requirements:

- The player should be able to walk around
- The player should be able to get burned
- The player should be able to extinguish the fire
- Extinguishing is only possible if the player has an extinguisher
- Extinguishing the fire before getting burned to a crisp should end the game
- The environment should be recognisable as the corridor on the 12th floor

The object was to use a model of one of the corridors in the EWI building. The twelfth floor corridor, which was the one that needed to be modelled, is rather long and complex, with doors, rooms and elevators on both sides. When I started on this project I had no 3D modelling experience whatsoever, therefore I decided to recreate only a small portion of the corridor and to use that as an example for whether or not an entire game was feasible. So I chose one end of the corridor and started learning how to use Blender. With a bit practice I was able to produce a model of the corridor which was recognisable. The first fire was created using the particle editor of Delta3D.

The Delta3D engine

At the beginning of the project I had spent a few weeks getting to grips with C++ and the game engine. So I was now able to fairly easily load the objects into a map and add a camera enabling the user to look around. This early version of the program contained no collision model or other functionality. In order to add further functionality the Delta3D game manager must be used. The game manager allows the developer to write custom actors, messages and components and handles all the managing tasks, such as making sure that the right actors get the right messages. Before starting to program I made a design. However, this design is Delta3D specific, which is why I'm adding it here and not at the beginning.

The design for Delta3D

The Actors:

Extinguisher: this is what the player can use to extinguish the fire. In order to use it, the extinguisher must first be selected from the user's inventory. Then it must be pointed at the fire and used by pressing the right mouse button. This only works if there is still some water left in the extinguisher. This is indicated by the variable contentsLVL.

Player: this is the most important actor. As this is a serious game it must be as realistic as possible. That is why the user sees the world through the "eyes" of the actor player. This creates a sense of actually "being there". It also means that there is no model which represents the player. The most important part of the player is the variable health, which represents how badly (if at all) the player is hurt. If this runs out the player is dead and the game ends. The health of the player is decreased by walking around in a space where there is still smoke and fire. Touching the fire also decreases the health.

Another important variable is the inventory. This stores pointers to the items the player has with him/her. Only if an item is in the inventory can it be selected and used.

Fire: this is a simple particle system. Whilst the extinguisher is being used it slowly decreases in size. The fire also has a health variable. When this becomes zero the fire is out.

Hall: this is the model that the player can walk through. It will have a collision model which prevents people from walking through the walls. It has no inherit behaviour.

The use case:

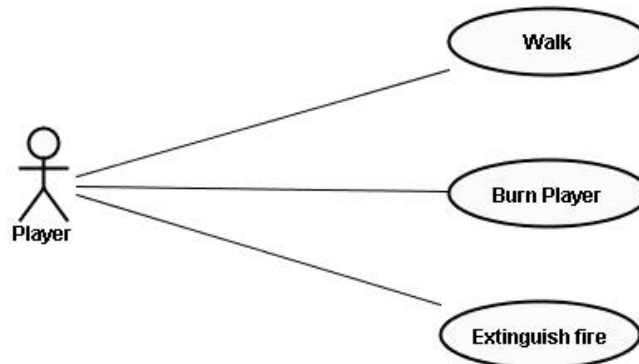


Figure 5
The use case.

The UML design:

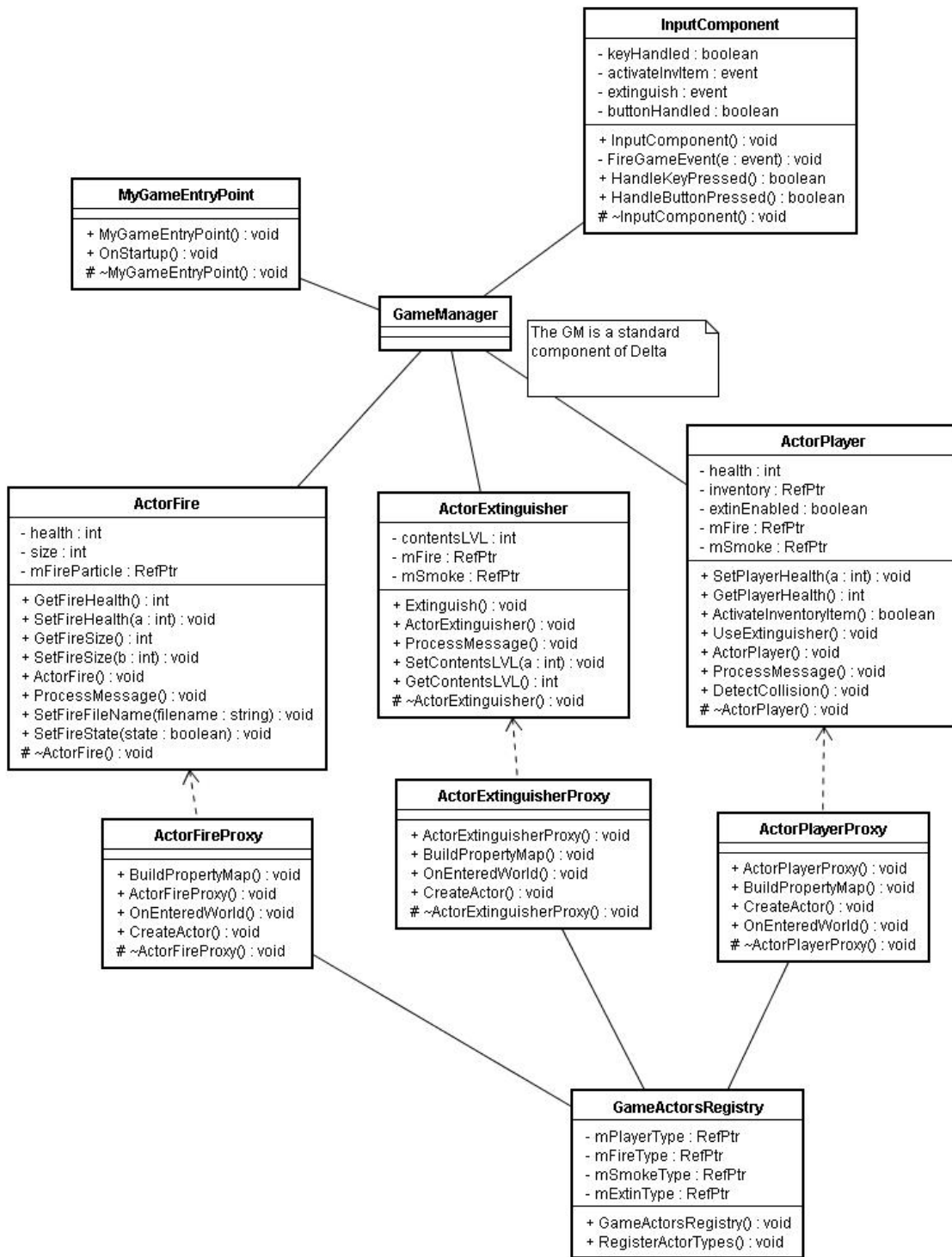
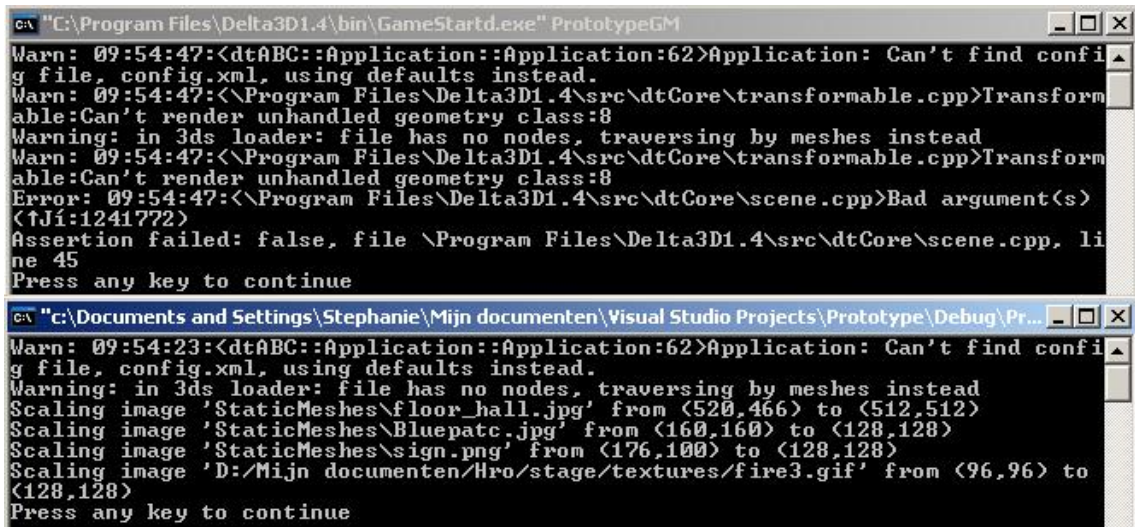


Figure 6
The UML diagram

The implementation of this design turned out to be harder than initially assumed. The custom actors having been written, these needed to be loaded into STAGE in order to be used. This took several tries to get it right. Also, due to lack of documentation the difference between the basic types of actors were unclear. This led to a lot of time being wasted on writing and rewriting some of the actors. However, the game refused to run due to some assertion that failed. After having chased this error for a few weeks and with the help of the game engine developers on the forum, it was concluded that it was probably a configuration error. I spent some more time trying to track down this error, but was unable to find the cause. Due to lack of time I then moved on to the next engine.



```
g:\ "C:\Program Files\Delta3D1.4\bin\GameStartd.exe" PrototypeGM
Warn: 09:54:47:<dtABC::Application::Application:62>Application: Can't find confi
g file, config.xml, using defaults instead.
Warn: 09:54:47:<\Program Files\Delta3D1.4\src\dtCore\transformable.cpp>Transform
able:Can't render unhandled geometry class:8
Warning: in 3ds loader: file has no nodes, traversing by meshes instead
Warn: 09:54:47:<\Program Files\Delta3D1.4\src\dtCore\transformable.cpp>Transform
able:Can't render unhandled geometry class:8
Error: 09:54:47:<\Program Files\Delta3D1.4\src\dtCore\scene.cpp>Bad argument(s)
<↑Jí:1241772>
Assertion failed: false, file \Program Files\Delta3D1.4\src\dtCore\scene.cpp, li
ne 45
Press any key to continue

c:\Documents and Settings\Stephanie\Mijn documenten\Visual Studio Projects\Prototype\Debug\Pr...
Warn: 09:54:23:<dtABC::Application::Application:62>Application: Can't find confi
g file, config.xml, using defaults instead.
Warning: in 3ds loader: file has no nodes, traversing by meshes instead
Scaling image 'StaticMeshes\floor_hall.jpg' from (520,466) to (512,512)
Scaling image 'StaticMeshes\Bluepatc.jpg' from (160,160) to (128,128)
Scaling image 'StaticMeshes\sign.png' from (176,100) to (128,128)
Scaling image 'D:/Mijn documenten/Hro/stage/textures/fire3.gif' from (96,96) to
(128,128)
Press any key to continue
```

Figure 7
The illusive errors

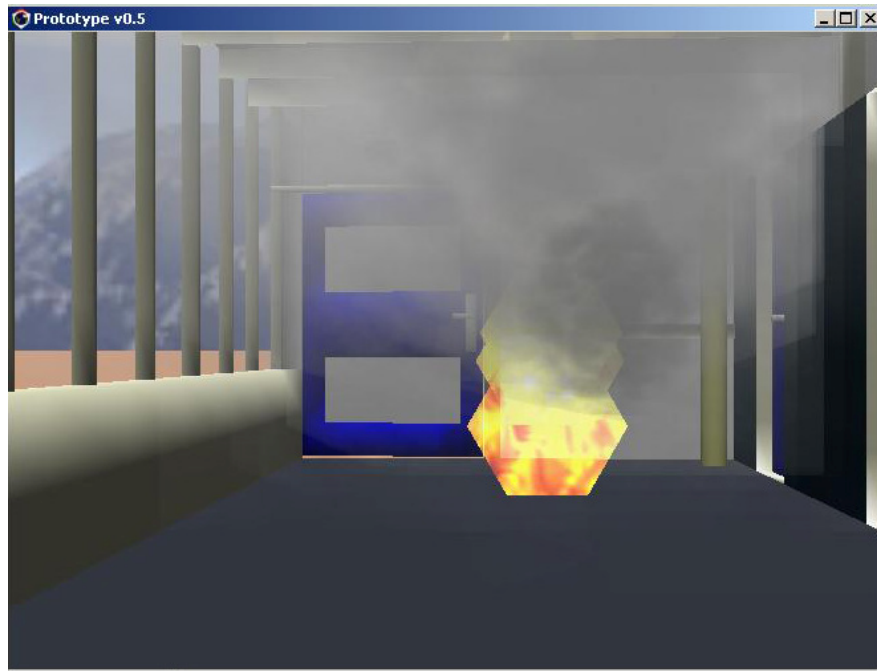


Figure 8
The only working version of the game

The Kaneva engine

One of the things I noticed whilst working with Delta3D was how difficult it was to create an MMO with the engine. So with the university community idea in mind I began searching for an engine geared towards MMO development. The Kaneva engine sprung out, as it was the only free engine which would run on my system. This meant that I could download it and start trying it straight away. In my opinion still the best way to get to know an engine.

The only thing I tried with the Kaneva engine was the beginners tutorial. The ease with which a basic scene, including only a house and some environmental things such as fog, were created made a good impression. The player's character is created at start up in the automatically generated login screen. This character can then be used in the scene that was made earlier.

Despite this ease it quickly became clear, that whilst this engine is very much suited for the development of MMO games, this wasn't what I needed., as I was not making an MMO game.

The Blender Game Engine

This engine has one big advantage over the previous two, it's also my modelling program so there's no need to import, or export the models. I decided to use the Blender engine for two reasons. Firstly, I wanted to have something working by the end of my internship and this seemed the easiest way to get it done. Secondly, after having listened to several professionals talking about game development, I was curious to try a game engine which is closer to the ones most companies are using. These game engines are often third party, high level game engines. With games having become more and more complex over the years, it's become impossible for the programmer to write a piece of code for everything the game designer wants. So the majority of game development companies use level editors which come with ready made logic blocks and require no programming skills. This way the game designers can make the levels that they want, whilst the programmers can concentrate on improving the rendering times and physics. The Blender Game Engine, being aimed at Blender users who are in the first place 3D modellers, offers exactly this kind of functionality.

Using the model that I had created for use with the Delta3D engine, I started turning the static scene into a game. First I added a fire, which is nothing more than a particle to which an empty object (an object that has no mesh) has been added. This empty object acts as an emitter, creating new particles in a way that resembles a fire. The difficult part was making all the particles move in the same upwards direction and to make sure that they weren't solid. At first the fire was more like a particle canon. Everything that got hit was eventually knocked out of the scene. Having fixed the fire, an extinguisher which can be picked up by the player was added. Finally, I added a simple sphere to function as the player. This because it proved to be rather hard to use a first person view, and character modelling, let alone animation, are still beyond my abilities.

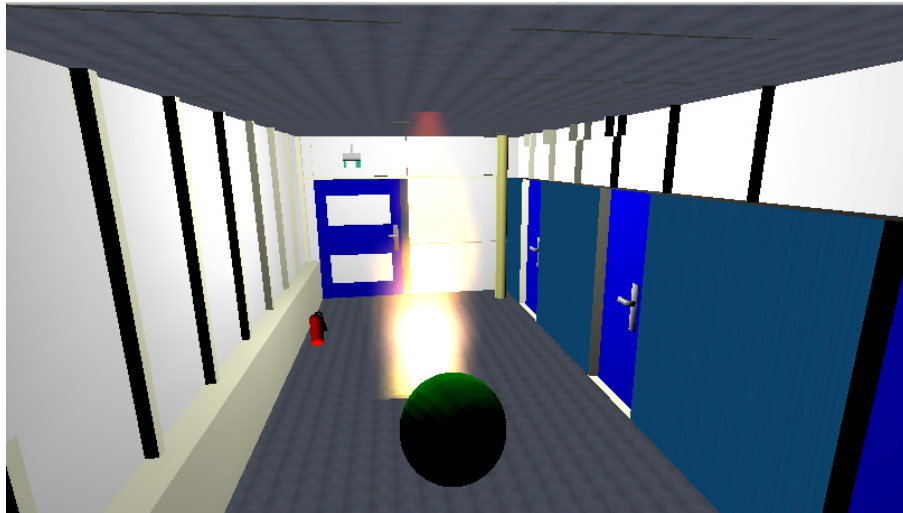


Figure 9
The game in action

The design for the Blender Game Engine

Having all the basic elements the next step was adding logic, the same basic idea as can be seen in the design that was used for Delta3D. The player starts the game with full health and a blazing fire in front of it. Next to the fire is an extinguisher which can be picked up. Having picked up the extinguisher the fire can be extinguished by pressing the “e” button. Once the fire has been extinguished the player can advance to the door, the touching of which resets the game. Throughout the entire game the player will lose health if he/she ventures too close to the fire. Losing all their health will result in the disappearance of the sphere and then thus the end of the game.

The logic block system doesn't really lend it self very well to UML adaptation, but the diagram below should give some idea of the functions that have been used to create the game.

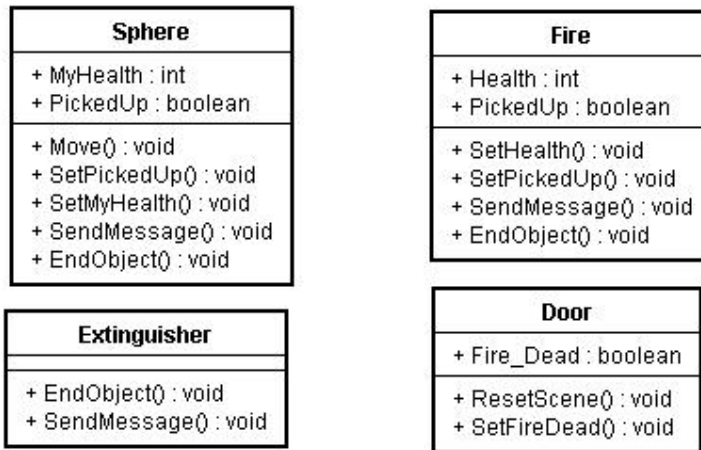


Figure 10
The design for the Blender version

Conclusion

The assignment was to make a proof of concept for a serious game situated in the EWI building at the TU Delft. Over the course of the past months I have tried several game engines and approaches. I would say that it is possible to develop a serious game at the TU Delft using the game engines that I have examined. The real question is exactly what kind of game do you want to develop?

If it's a game that's not unlike my demo, I would suggest a game engine similar to the Delta3D engine. Though I would recommend a more developed and better documented engine, such as the Quake or Half-Life engines. Because unless you happen to have a team of coders who are familiar with C++, OSG and have a lot of experience with game development, it's going to be difficult to get Delta3D to do what you want it to do within a reasonable amount of time. I had none of these things when I started this assignment and consequently spent the first three months trying to get something to work in Delta3D, which I managed to get working in Blender in two weeks. The biggest part of which were spent learning how to use the game engine and making the extinguisher and fire; about two hours were needed to add the actual functionality.

Checking to see whether your proposed engine has an easy to use editor is also a wise thing to do. A proper editor will save a lot of time when it comes to level design. The Blender Game Engine could also be used for this.

No matter which engine you use you'll have to make models and using your modelling program to make the game has its advantages. However, this engine has the same drawbacks as the Delta3D one. It's open source, not always well documented and a lot of functions will have to be written by the developer. Though the fact that Blender uses Python and has integrated its physics engine makes it easier to learn.

If you are set on developing the university online community, then Kaneva is your best bet. This engine already has all the ingredients that you need to make an MMO. Furthermore it's user friendly, which means that less time has to be spent figuring out how to do simple tasks and more energy can be put into actually making the concept work. The fact that it wasn't specifically developed for a certain game is also a plus. For instance, using an engine that was made by a company to use for their fps and then sold commercially is very nice if you want to make an fps, but not if you're looking for something to make an rpg with.

Appendix A

| Datum | Uur | Werkzaamheden |
|---------|-----|--|
| 12-2-07 | 8 | Begonnen aan stage. Opdracht gekregen en begonnen met het uitzoeken van wat er voor die opdracht gedaan moet worden. |
| 13-2-07 | 8 | Game engine is in C++ dus begonnen aan C++ tutorial. |
| 14-2-07 | 8 | C++ tutorial classes en pointers. |
| 15-2-07 | 8 | CMS03 opdracht aan 't uitwerken als oefening. |
| 26-2-07 | 8 | CMS03 werkend gekregen. Beginners tutorial afgerond. Gezocht naar tutorial die grafisch en netwerk programmeren uitlegt. |
| 27-2-07 | 8 | Begonnen aan tutorial over MFC classes. (De engine is hierin geschreven.) |
| 28-2-07 | 8 | Tutorial MFC classes. |
| 1-3-07 | 8 | Modal dialog windows in c++ |
| 5-3-07 | 8 | Modal dialogs afgerond, door met Modeless windows. |
| 6-3-07 | 8 | Modeless windows. |
| 7-3-07 | 7 | Nog even bezig geweest met toolbars. Daarna ben ik gestopt met de MFC classes en heb Delta 3D geprobeerd te installeren. Door aan houdende hoofdpijn ben ik een uur eerder naar huis gegaan. |
| 8-3-07 | 8 | Delta 3D geïnstalleerd. Hello world geschreven en met de camera gespeeld. Blender geïnstalleerd voor later gebruik. |
| 12-3-07 | 8 | Verder met het leren van Delta3D, bezig geweest met object motion. Ook bezig geweest met omgevingen. |
| 13-3-07 | 8 | Omgevingen en character beweging. Alle tutorials zijn outdated dus het creëren van werkende code is lastig. |
| 14-3-07 | 8 | Character beweging afgerond en begonnen met Terrain tutorial. |
| 15-3-07 | 8 | Terrain lijkt niet goed te werken op mijn laptop. De video kaart kan er niet mee omgaan. Begonnen met STAGE tutorials. |
| 19-3-07 | 8 | Verder met STAGE tutorials. Begonnen met actors. Gesprek gehad met stage begeleiders. Opdracht is nu eindelijk duidelijk. |
| 20-3-07 | 8 | Begonnen met Blender. Weet nu wat er nodig is, dus vuur gecreëerd. Rook zit standaard bij de engine. |
| 21-3-07 | 5 | Boek over Blender gehaald. Begonnen met modelleren. |
| 22-3-07 | 8 | Verder met modelleren. |
| 26-3-07 | 8 | Plan verder uitgewerkt. Idee voor een prototype uitgewerkt. |
| 27-3-07 | 8 | Prototype versie 0.1 gemaakt. Verder gegaan met modelleren |
| 28-3-07 | 8 | Verder gegaan met modelleren. Documentatie bijgewerkt. |
| 29-3-07 | 8 | Verder gegaan met modelleren, materiaal tutorials. |
| 2-4-07 | 8 | Begonnen met het modelleren van een stuk van de gang. Besloten om niet heel de gang in een keer te maken. |
| 3-4-07 | 8 | Verder met modelleren van de gang. Versie 0.5 af. |
| 4-4-07 | 8 | De gang materialen en texturen gegeven. Nooduitgangbord toegevoegd. Bij het exporteren naar zowel osg als 3ds gaat er informatie verloren. |
| 5-4-07 | 4 | Model aangepast voor exporteren. Versie 0.5 van het prototype gemaakt. Documentatie bijgewerkt. |
| 10-4-07 | 8 | Documentatie verder bijgewerkt. Begonnen aan UML ontwerp van |

| | | |
|---------|-----|--|
| | | het programma. |
| 11-4-07 | 8 | UML ontwerp afgemaakt. |
| 12-4-07 | 8 | Begonnen met het uitwerken van het UML diagram. |
| 16-4-07 | 8 | Verder gegaan met het uitwerken van het ontwerp. |
| 17-4-07 | 8 | Verslag naar het Engels vertaalt. Verder geprogrammeerd |
| 18-4-07 | 8 | Voor het eerst de code getest. Importeren in STAGE lukt. Actors aanmaken niet. |
| 19-4-07 | 8 | Actors aanmaken lukt nu wel. Documentatie bijgewerkt. |
| 23-4-07 | 8 | Bezig geweest met het game manager gedeelte van de code. |
| 24-4-07 | 8 | Ben tegen een error waar ik niet uitkom opgelopen. Ben nu documentatie aan het bijwerken in afwachting van een antwoord v/d community. |
| 25-4-07 | 8 | Documentatie bijgewerkt, UML geupdate. |
| 26-4-07 | 8 | Wat gerommeld met het programma, krijg er nu een andere error uit. De oplossing voor die error staat waarschijnlijk op het forum van Delta3D en die is de hele dag offline. |
| 7-5-07 | 4,5 | Op het forum gekeken, maar wat daar staat blijkt een andere error te zijn. |
| 8-5-07 | 8 | Nog steeds niet achter het probleem gekomen. Besloten om het programma gedeeltelijk te herschrijven. |
| 9-5-07 | 8 | Probleem opgelost! Had de .dll de verkeerde naam gegeven. Krijg nu een andere fout terug. |
| 10-5-07 | 8 | Fout van gisteren ook opgelost, krijg nu weer een error terug waarvan het antwoord wederom misschien online te vinden is, maar de site is weer down. Het gedeeltelijk herschreven programma werkt wel, maar dat doet weer niet wat het moet doen. En STAGE geeft nu ook een error. |
| 14-5-07 | 4,5 | Verder gewerkt aan de nieuwe versie van m'n programma. Heb het oude voorlopig even opgegeven. Bij de nieuwe library hebben de actors wel alle variabelen die ze moeten hebben. |
| 15-5-07 | 8 | Probeerde de map van TryGM te veranderen en dat lukte niet. Toch maar weer terug gegaan naar PrototypGM. Probleem in STAGE opgelost, maar nu weer een andere terug. |
| 16-5-07 | 8 | Het probleem wordt waarschijnlijk veroorzaakt door een configuratie error. Lijkt erop dat Delta3D hergeïnstalleerd zal moeten worden. |
| 21-5-07 | 4,5 | Gesprek gehad met m'n stagebegeleider. Uitgelegd dat er een aantal privé problemen zijn waardoor ik niet zo goed functioneer. En waardoor ik er niet altijd ben. |
| 22-5-07 | 8 | Weer bezig geweest met die errors. Ben er inmiddels zo gek van geworden dat ik toen ik thuis kwam eerst alle frustratie heb weg gejankt. |
| 23-5-07 | 8 | Begonnen aan een text only versie van het spel. Volledig in C++ |
| 24-5-07 | 8 | Studium generale over Girls en Gaming in Amsterdam bezocht. Was erg interessant. |
| 29-5-07 | 8 | Verder gegaan met de text versie van de game. Merk dat ik gewend ben geraakt aan de RefPtr type van de game engine. Heb moeite met de gewone pointers. |
| 30-5-07 | 8 | Ben blij dat ik dit ff heb gedaan. Merkte dat ik de laatste weken toch vooral met het verbouwen van tutorials bezig ben geweest. Het was |

| | | |
|---------|---|---|
| | | een fijne uitdaging om een gewoon C++ programma te schrijven en erg goed voor mijn programmeer vaardigheden. |
| 31-5-07 | 8 | Nog een keertje naar Delta gekeken en besloten om het op te geven. Verder gaan zoeken naar andere potentiële engines. |
| 4-6-07 | 8 | Op zoek gegaan naar een alternatieve game engine. Multiverse engine geprobeerd, werkt niet op mijn system. Kaneva engine tegengekomen. |
| 5-6-07 | 8 | Kaneva engine geprobeerd. Ziet er goed uit is goed te gebruiken, maar niet wat ik zoek. |
| 6-6-07 | 8 | Besloten om de blender game engine te gaan gebruiken. Begonnen met wat tutorials. |
| 7-6-07 | 8 | Gesprek gehad met stagebegeleider. Ga nu een werkende demo maken in Blender. |
| 11-6-07 | 8 | Verder gewerkt aan Blender. |
| 12-6-07 | 8 | Vuur gecreëerd met behulp van tutorial. |
| 13-6-07 | 8 | Om een of andere reden wordt het gebouw weggeschoten zodra ik er een actor van maak. |
| 14-6-07 | 8 | Begonnen met het maken van een extinguisher. |
| 18-6-07 | 8 | Extinguisher afgemaakt. Deze wordt ook weggeschoten door het vuur. |
| 19-6-07 | 8 | Opnieuw naar het vuur gekeken, in wireframe mode is duidelijk te zien dat de particles als kogels de andere voorwerpen weg schieten. Vuur gecorrigeerd. |
| 20-6-07 | 8 | Geprobeerd text aan de scene toe te voegen. Werkt wel in aparte file, maar niet in volledige scene. |
| 21-6-07 | 8 | Functionaliteit toegevoegd. Text lukt nog steeds niet. |
| 25-6-07 | 8 | Begonnen aan stageverslag. |
| 26-6-07 | 8 | Versie 1.0 van verslag af. |

Appendix B

Literatuur:

[1] www.blender.org

[2] www.delta3d.org

[3] www.kaneva.com

[4] www.wikipedia.org

[5] Carsten Wartmann, The Blender Book, free 3D graphics software for the web and video, Linux Journal Press, 2001