BACHELOROPPGAVE:

Novel interactions for adventure game Mythophobia

Forfattere: **Lars Inge Reinsnos**
Sander Struijk
Arild Hembre
Heine Martin Brekken

Dato: **20. mai 2010**

Sammendrag av Bacheloroppgaven

Tittel:	Novel interactions for adventure game	
		Dato: 20.05.2010
Deltakere:	Lars Inge Reinsnos	
	Sander Struijk	
	Arild Hembre	
	Heine Martin Brekken	
Veiledere:	Simon McCallum	
Oppdragsgiver:	Høgskolen i Gjøvik	
Kontaktperson:	Øyvind Nordstrand, oyvind.nordstrand@k-h.no, 48178222	
Stikkord:	Spillmotor, 3D verden, XNA, interaksjoner	
Antall sider: 129	Antall vedlegg: 8	Tilgjengelighet: Åpen

Kort beskrivelse av oppgaven:
Produksjon av et PC spill med muligens utvidelse til XBOX 360, i XNA med fokus på utvikling av et innovativt spill interagerings system. Det inkluderer utviklingen av en spill motor basert på Microsofts XNA rammeverk, med bruk av spill motorens funksjoner i spillet. Disse funksjonene inkluderer tegning og oppdatering av 3D omgivelser, bruk av høyde kart (height maps), kontrollering av en spillbar karakter, administrering av inventaret til karakteren og interagering med 3D omgivelsene / verden.

Summary of Graduate Project

Title:	Novel interactions for adventure game	
		Date: 20.05.2010
Participants:	Lars Inge Reinsnos	
	Sander Struijk	
	Arild Hembre	
	Heine Martin Brekken	
Supervisor:	Simon McCallum	
Employer:	Gjøvik University College	
Contact person:	Øyvind Nordstrand, oyvind.nordstrand@k-h.no, 48178222	
Keywords:	Game engine, 3D world, XNA, Interaction,	
Pages: 129	Appendixes: 8	Availability: Open

Short summary of the main project:
Production of a game for PC and possibly Xbox 360 developed in XNA with focus on development of novel game interfaces. Includes development of a game engine based on the Microsoft XNA framework, and the use of the game engine features in the game. These features include drawing and updating a 3D environment, use of height maps, and control of a playable character, managing an inventory and interaction with the game world.

Novel interactions for adventure game – Mythophobia

Lars Inge Reinsnos
Sander Struijk
Arild Hembre
Heine Martin Brekken

20.05.2010

Preface

The project was presented at Gjøvik University College, December 2009. There were ten projects to choose between and every group was to consist of two to four members. The only project which resembled around a game was the Novel Interaction project, and then became the individual group member's natural choice as we all are students of the game programming bachelor degree at Gjøvik University College.

Before the bachelor projects were presented, students were allowed to apply with their own ideas for bachelor projects, and Novel Interactions was Lars Inge Reinsnos' contribution, who acts as our group leader.

The group work began early January 2010, with two weeks of planning, followed by programming work. All the group members have worked excessively from start throughout the whole project working up to 12 hours a day. We have been very dedicated and determined for this project to participate and get far in the Hamar Game Challenge competition, hopefully winning resulting in us starting our very own game development company.

Working on this project has been a valuable and great experience for us all, where we have participated in the development process of a game from the beginning. Now we have a greater understanding for a big game projects development process and what it means to plan ahead. We have also gained valuable experience with XNA, C#, graphics programming and other general programming.

We want to thank Simon McCallum for being a great supervisor to us and at the same time our Chief executive officer. We would also like to thank Øyvind Nordstrand acting the role as producer on the behalf of Kunnskapsparken Hedmark and Jayson Mackie for providing additional feedback and coding support.

Sander Struijk Arild Hembre

Lars Inge Reinsnos Heine Martin Brekken

Table of Content

1 Introduction

This chapter describes what we want to create, how we will create it, what frames we will work within, what we want to accomplish by working on this project and which tools we choose to work with.

We also describe who we are creating this project for, what our reasons are for choosing this project and specify our objectives. The Introduction defines the style and structure for the rest of the report.

1.1 Report organization

Chapter 1 - Introduction

This chapter describes the project report and its content.

Chapter 2 - Project requirements

This chapter describes the projects requirements, structure

Chapter 3 - Design

This chapter contains the design of the game and describes the pre project preparations

Chapter 5 - Testing and quality assurance

This chapter describes the quality assurance processes and methods we have used

Chapter 6 - Installation

This chapter describes what is needed to be able to run the game and the project solution

Chapter 7 - Development process

This chapter describes what kind of development process we planned for this project and how we used it.

Chapter 8 - Discussion of results

This chapter contains discussions regarding project results and choices

Chapter 9 - Evaluation of the group work

This chapter contains the evaluation of the group work

Chapter 10 - Conclusion

This chapter contains the projects end conclusion

Appendix / Attachments

The reports attachments will be in this chapter.

1.1 Project Description

For this project we aimed to create a fully functional 3rd person action/adventure game, focused on action-sequences and logical adventure-game puzzles. We wanted to make the players work with objects and store the items in their inventory in a realistic way. The game will emphasize challenges when it comes to strategy and thinking, but also challenging when it comes to action, such as shooting and fighting.

With interaction in mind, the GUI would be designed to make the interaction between user and objects at hand as practical as possible and easy to learn.

The graphics has been made in 3D, but the level of detail for the visual aspects of the game are not of the highest quality, but still fairly decent.

During the development period we realized that much of our work time would be consumed on making a functional game engine. We had to make the game engine based on the foundations of the XNA framework, and doing so was a great deal of work. Interesting aspects would be how the player moves in the world with the use of height maps, switching between camera modes and how player is able to shoot in one camera mode and able to pick up and interact with objects in another.

The story of the game is still under development, the beginning is determined, but no story has been included in the development seeing that we used our development time on the functional game engine and interaction between the player and the game world.

1.2 Group structure and organization

Group Leader – Lars Inge Reinsnos (Substitute if incapacitated: Heine Martin Brekken)

Document manager – Arild Hembre (Substitute if incapacitated: Sander Struijk)

Webmaster – Heine Martin Brekken (Substitute if incapacitated: Lars Inge Reinsnos)

Aside from these specified responsibilities everyone in the group worked as designers and developers on the project.

The project leader's role has been to keep track of the main interests in the project, to guide, and lead the team through any uncertainties. The Webmasters responsibility has been to keep the website working and add improvements if needed. The document manager's responsibilities have been to keep track of, and update all documentation throughout the project. This includes uploading meeting reports and other information to the website. However everyone has had a shared responsibility to improve the documentation as needed.

If any of the team members became incapacitated by any means throughout the project, we had appointed members who would temporarily substitute their respective roles.

1.2.1 Employer

Our employer was Øyvind Nordstand from Kunnskapsparken Hedmark. He was to give feedback on the project through an idea presentation and an early alpha presentation. He would also gain insight in the development through the projects' website.

1.2.2 Supervisor

Our supervisor was Simon McCallum, who also is one of our course lecturers. He was to assist the group to solve issues and problems during the development phase, and give feedback on the group's work on a scheduled basis. He was also to help the project management and suggestions and help when we would need it.

1.3 Target group

The target group for this end project report should have basic programming knowledge. We have tried to write the report with this in mind and described terminology where needed.

1.3 Purpose

The purpose of this bachelor project has been to practice what we have learned the past three years as game programming students, by starting to develop a fully playable 3D action adventure game.

The development of a game from start to retail can last up to three years and even more. The bachelor project does not require us to finish the game, but rather to start the development and get as far as we can in our time constraints, in the purpose of learning. Learning how larger projects are directed in terms of responsibility, deadlines, meetings and communication, keeping up with the schedule, planning ahead and the programming itself.

We think Mythophobia is a good game idea, and by starting to develop this game with the help of our supervisors at Gjøvik University College, we could get a good start at continuing with the development of the game after the bachelor thesis, for a future retail release and the creation of a new game developing company.

1.4 Our proficiency and background

Everyone in our group is attending the same bachelor degree, Game programming, but two of us had a little experience with programming before starting the degree.

A great deal of the programming experience we have is basic programming, and less experience with making the different components of a full 3D game.

Three of us worked with XNA as a developing tool for a few months before this project, where we made a 2D game for a mandatory assignment in one of our courses.
Our last member will therefore have to take up on the basics that the rest already know.
Apart from this we do not have much experience and knowledge about programming a project of this size, which is much more advanced.

Some of what we must learn is 3D projection and camera space versus world space and local space. Instead of 2D images drawn on screen we will be using 3D models and primitives, which is new to us. We will have to find out how this is done in XNA, how we set it up and how we draw it

to the screen. We also have to learn shader programming if we want to add custom lightning, shadows, particle effects and such. The language used for this is HLSL[1].

Collision detection in 3D is not something we have done before and it is a lot more advanced than simple 2D collision, so we need to learn how this works and find out how to do it efficiently.

1.5 Frames

To accomplish the most out of our goals on this project we started the development process by setting up both a work schedule in form of a Gantt chart and a timetable. Before starting to do any actual programming, we used a week to prepare ourselves for what programming is required.

We set up a work schedule where each member of the group would work on the bachelor assignment 5-7 hours each weekday. In addition, we also sat up weekly meetings each Friday to present and discuss individual progress with the rest of the group.

The project was managed by both the group leader and our supervisor, who were to sit in on every second weekly meeting. We did not get to follow these scheduled supervised meetings, but had regular meeting to present and discuss the progress with him as well.

With these preparations in place, we aimed to make the most out of our bachelor project, and get as far and get the best results as we could in the time we had.

1.6 Development environment

We have been using the following software or programs in the development of the game, and documentation.

1.6.1 Documentation

At first we were encouraged to write the report with the help of a program called LaTeX[2], which is a way of organizing the layout of the document following a given template. Using LaTeX gives an easy and automated way to make sure everything in a large document follows the same set of rules.

LaTeX is a programming based text editing software. This means that to be able to use LaTeX, one would have to know the syntax of the programming language and learn it properly. Seeing that learning a whole new programming language at the end of our bachelor would consume a great deal of time, we chose to rather write this document in Microsoft Word.

[1] http://en.wikipedia.org/wiki/High_Level_Shader_Language
[2] http://www.latex-project.org

[26] Using **Microsoft Word** will require more text editing work, and give very few automated tasks, in relative to using LaTeX. But, this work amount will not be as comprehensive as learning how to use LaTeX.

We have also used [28] **Doxygen** for auto generation of code documentation with class hierarchy

[15] We used **Visual Paradigm** in the creation of the class diagrams.

1.6.2 Code development tools

We have used [21] **Microsoft Visual Studio** with the addition of the [5] **XNA Game Studio 3.1** development kit, in code development for this project. They can both be found in freeware versions online at Microsoft's homepage so they can easily be acquired for any developers who wish to use XNA as a game development platform.

1.6.3 3D Modeling

We have used [23] **3D studio max** in the development of 3D models for the game, and exporting them to the games usable FBX format, baking in animations and textures into the models where needed.

1.6.4 2D Art and sprite creation

We have used [24] **Photoshop** in the process of developing game art and sprites for use in the game. Photoshop is a well known and excellent picture editing and creation tool.

2 Project requirements

This chapter describes the guidelines we had, and what we requirements we sat for ourselves when starting the project.

We also describe an analysis of the project where we describe the game we have developed during the projects lifespan.

2.1 Project guidelines

In this project we focused on the development of novel game interfaces. These interfaces include:

- Camera control (switching between views without losing world orientation)
  - 3rd person and 1st person
  - Moving the camera between different views
  - Controlling the camera by scripting
- Combat controls
  - Ranged and melee weapons
  - Aiming with both crosshair and without
  - Different targeting views for different weapons
- Equipment
  - Equipment carrying limitations
  - Equipment placement on character
  - Multiple results from combination of items
- Adventure interaction
  - Interacting with items in world and inventory
- Health features
  - Getting infected, affecting the character in various ways.
  - Dealing with and removing infections.
  - Getting exhausted from running, jumping and using heavy weapons
  - Exhaustion limiting your actions

This project will implement a prototype for testing the above interaction techniques.

Mythophobia at retail state would be a fully functional 3rd person action/adventure game, focused on action-sequences and logical adventure-game puzzles, where we want to make the player able to store items in their inventory on a realistic level and being able to interact with the world through a familiar adventure game interface.

We aim to make the game challenging in terms of strategically actions, such as saving ammunition or figuring out a way to pass enemies, either by moving strategically or figuring out an adventure puzzle to find a way around enemies.

Action sequences will even so be a large part of the game. Fighting your way through some parts of the game will be required, though with limited resources in terms of ammunition and health.

2.1.1 Demarcation

The main objective for the bachelor project was to develop an innovating novel interaction game. Meaning we would not focus on graphics and story in the first place, but make the interaction part work at its best. Interaction is both how you move around in the world and how you interact with objects in the world through the Graphical User Interface (GUI). We only had a limited amount of time to finish this project and were four students working on it, so we did not aim to achieve a complete retail game.

2.1.2 Appraisal

One of the things which are important for this project is to make a playable game. We will accomplish this is by focusing on the technical issues, instead of the visual effects. The interaction between the player and the game must be effective and intuitive, and yet innovative. To help us with this issue, we wish to find a group of people, in the suitable demographic area, who can play

the game and share their opinions with us. This information will provide us with the problems we could not see ourselves and we can correct the flaws in the game to make it better and have the game tested again.

2.2 Project analysis

The game developed in this project is based on interesting interaction in 3D and we have spent most of our time emphasizing this. Our idea is to allow the player interact with objects on a detailed level as well as using firearms and melee weapons to fend off enemies in the game world.

The player can pick up items, combine them and use them for the player's benefit. When a player discovers an object in the game world, the player can switch the camera mode to first person view and the object will appear with a name over it. Then the player can open his interaction menu and choose which action he wishes to perform on the object.

Storing items the player finds in the game is done in a realistic way, where every pocket and logical storage location on the game character becomes an inventory slot. This means that the player's inventory is limited to the clothes on the played character or other extra containers he or she may be carrying, such as a backpack or a purse.

When the player faces an enemy, he can switch the camera mode to aim view and he can shoot or hit the enemy, depending on the type of weapon he uses. We want the theme of the game to be dark and scary, so that fighting an enemy would be the last resort. Prioritizing sneaking past enemies or figuring out other routes past them will be preferred.

3 Design

This chapter contains a summary of the design and describes the preparations we have made in advance before the group began working on the project.

3.1 Design Summary

Mythofobia is a 3rd person true action/adventure game, focused on both action-sequences and logical adventure-game puzzles. Set in a horror-theme with mythological creatures such as zombies, warewolfes and vampires

The game is played mainly through a 3rd person perspective, using a 1st person adventure perspective for world interaction, and handling inventory items as well as examination and combination of these in a “pause” menu.

The player will be able to walk, run, sprint, hit with melee weapons and shoot with guns, as well as execute a dodge move, where the character dodges danger.

When the player wants to perform an action, such as attack with a weapon, the camera will zoom in and change to a over-the-shoulder 3rd person view, and give a centered view in front of the player.

When the player wants to interact with the world, the camera will zoom even further and go into a 1st person view from the character’s point of view. In this view the center of the screen will function as a cursor to choose objects to interact with. When the player then chooses to interact with an item, an adventure menu appears, in which the player can choose to interact by *Hand*, *Foot*, *Eye* or one of the *Items* currently in hands.

To not complicate each interaction, simple interactions such as opening doors or climbing a ladder will be available through the 3rd person point of view. This action will choose the most common action for that specific object, such as open for a door, push for a button, look at for a picture, pick up for an item on the ground and jump over for a gap.

The inventory screen will feature a 3D model of the character in the top left corner, with the ability to rotate and zoom on it. This model has a drag and drop ability (by the use of a cursor) of items the character is currently carrying from any pocket or other itemslot to another. This will give a limitation of carry capacity to the amount of pockets or containers the character posesses.

Some items will also not be able to store in any chosen item slot. For example a rope or a shotgun may not be put in a pocket, but must be hanged around a shoulder or held in hand. As well as a gun may not the put in a chest pocket or around the neck, but in a pocket of the right size, the belt or in hand.

The game also includes features, such as character exhaustion, making the player exhausted when running, sprinting and attacking, giving the player melee damage reduction and unsteady aim. Also the character gets infected when getting bit or attacked by the enemies, and must use an antidote before the infection is complete.

In addition to including these features the game is a horror game, and naturally, interesting aspects will include shadows and different kind of light sources in the world.

A complete version of the design document can be found in “Appendix G: Design document”.

3.2 Coding preparations

Before starting development and coding of the game itself, we decided to use one week of our time to create an overview of the coding needed to complete the game.

The diagram in Figure 3.1 was created using Visual Paradigm[3]. This allowed us to generate class files directly from the GUI as well as import classes from files.

During this preparation week, the whole group contributed and discussed classes we needed. We used blackboard and wrote down classes needed, as well as filled the classes and variables needed into Visual Paradigm.

This gave us a slight advantage when starting to code, since we already knew and had generated many of the classes we needed in the game. Although, there were classes and variables which we realized that was needed and was implemented at a later time in the project time-span.

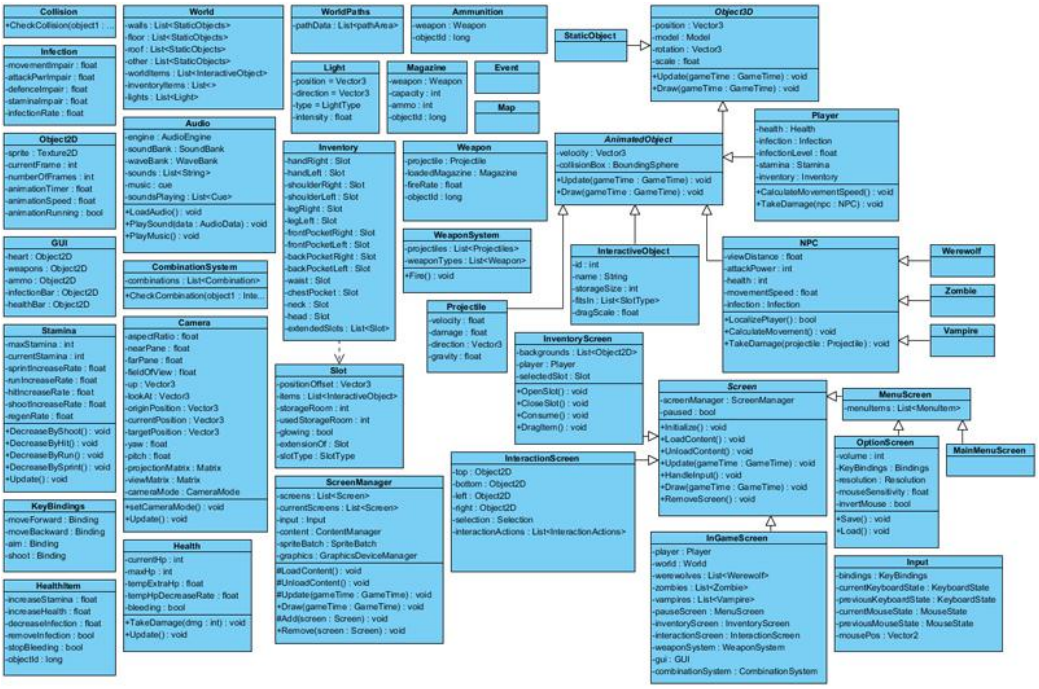


Fig 3.1 – The project's class diagram.

We separated the group into sub-groups of 2 when producing code for the project. Each sub-group was delegated separate programming tasks to do. As seen in "Appendix B: Original Gantt-chart", each sub-group got continuous tasks during the project's lifespan.

We chose to do it this way, so that we could group the weakest and the strongest programmers to work together and help each other and ensure that each tasks gets done appropriately.

[3] http://www.visual-paradigm.com

Although, during the projects span, we did deviate from the original plan and some tasks were separated into individual tasks, due to limited time near the end of the project. This can be seen in “Appendix C: Real Gantt-chart”.

4 Implementation, code and production

This chapter contains and describes all the technical aspects of the project itself.

Here we describe many of the most important coding and implementation aspects of different features that we have included in our game, game engine and the game engine's level editor.

4.1 Screen manager - Game State Management

When making a game, one of the first problems to tackle is how to manage the game state. In most games you will likely spend some time switching between different screens, like going from the main menu to the single player or multi player-screen.

More importantly going from the game-screen to the pause, option or main menu-screen. Every screen uses system memory and you do not want to remove the game-screen from the memory when switching to another screen.

To handle this we make a screen manager to keep track of and handle the different screens in use. We make base class “Screen” and a “ScreenManager” game component, where the “ScreenManager” keeps track of the active screens and in which order to draw the screens. The Screen class defines several methods like Initialize, LoadContent, Update and Draw that the different screens can use.

The Screen Manager has 2 lists with the different screens and the active screens as shown in figure 4.1.1.

```
private List<Screen> m_screens;
private List<Screen> m_currentScreens;
```

Fig 4.1.1 – Lists holding the screens.

When we want to open another screen, we make a new object of the screen we want and add it to the ScreenManager class. Depending on if we want the original screen to still be active or not, we can call RemoveScreen() to remove it from the ScreenManager. As shown in figure 4.1.2 we make the GameScreen in the LoadingScreen and then add it to the ScreenManager, before we remove the LoadingScreen.

The GameScreen will then become the active screen.

```
InGameScreen gameScreen = new InGameScreen(m_graphics, m_world,
m_playerOne.Clone(), m_inventory, m_weaponSystem);

screenManager.Add(gameScreen);
gameScreen.LoadContent();
gameScreen.Update(gameTime);
RemoveScreen();
```

Fig 4.1.2 – Adding a screen to the screen manager.

When drawing the screens the screen manager goes through the list of screens and checks if the current screen is active. If it is the screen is drawn, otherwise it goes on to the next screen until it has gone through them all. This is show in the code block in figure 4.1.3

```
...

foreach (Screen screen in m_currentScreens)
{
  if (screen.ScreenState == ScreenState.Inactive)
  {
    continue;
  }
  else
  {
    screen.Draw(gameTime);
  }
}

...
```

Fig 4.1.3 – Drawing the screens

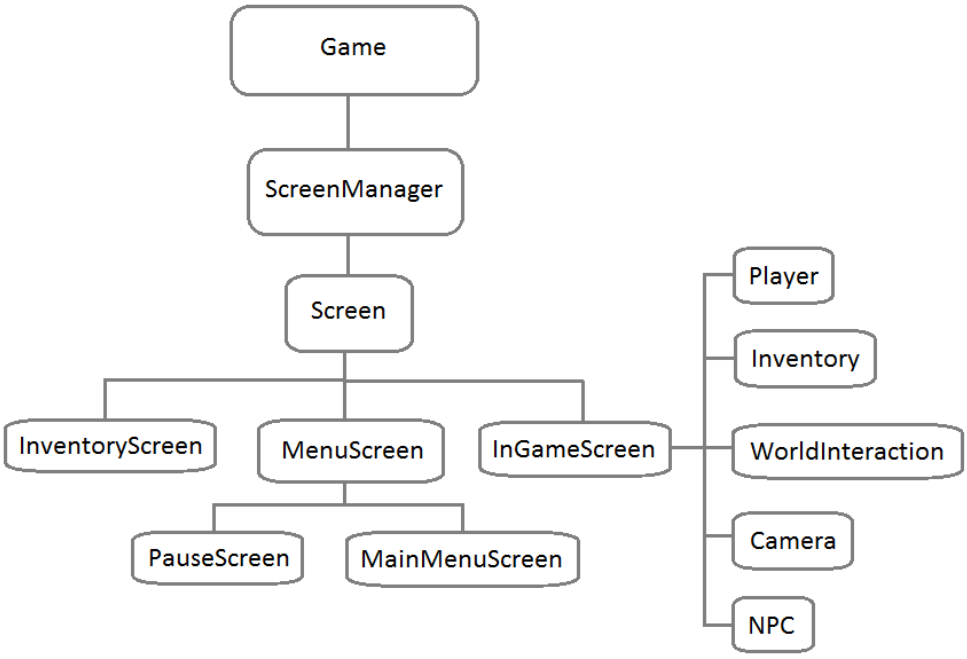


Fig 4.1.4 – Screen hierarchy

The different screens have overridden update functions as well as update functions derived from the abstract class Screen. The hierarchy of the screens are shown in figure 4.1.4.

The functions here are called from the ScreenManager which is added to the game components. This separates them so that they don't need to wait for each other if either of them is using more time than expected.

4.2 The game world representation

When it came to the representation of the game world, we first tried out having the whole world being made in 3D Studio as a whole model. This gave us the ability to create the walls, floors and roofs of the world correctly aligned to each other in a single step, but it had a few drawbacks as well.

If we were to use 3D Studio to model the whole level, we would still have issues regarding collision detection, both collisions against the player and enemies as well as the collisions against the camera. This would have to be done in 3D, model against model, and would produce a lot of collision checks.

Therefore we decided that we would build the level out of several models. Initially we had walls, which were represented by a 1x1 cube model, positioned and scaled so that it would look like a rectangle wall. Then we could place several of these scaled cubes around in the game world.

Regarding collision detection, we decided, after long discussions that it would be best and less demanding in terms of processing to perform the collision detection in 2D space. Even though we have a 3D world, we could take a top-down 2D collision detection approach.

This was done by cutting out the height position value of the player, while keeping the 2 other axis', and do the collision as if we saw the player in a top-down view in the world as illustrated in figure 4.2.1 and 4.2.2.

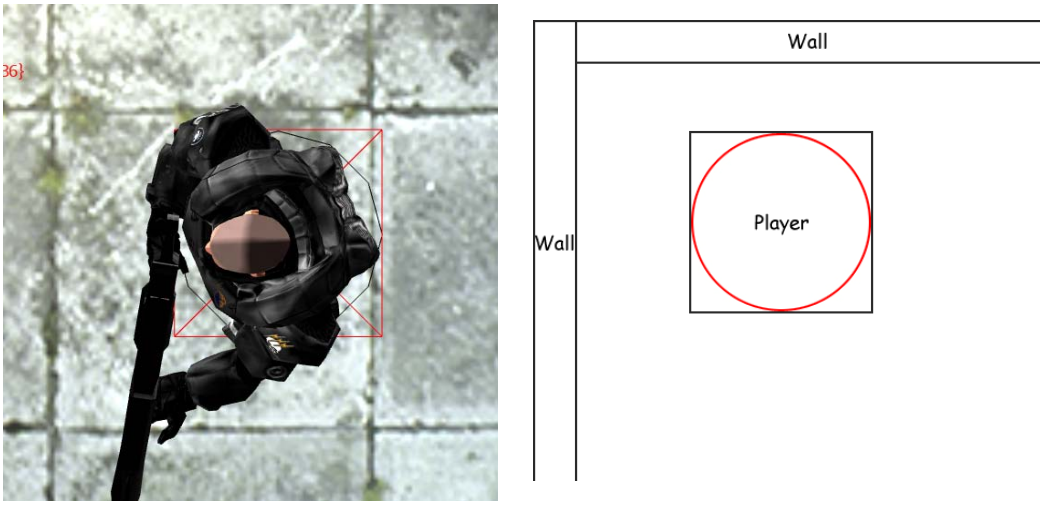


Fig 4.2.1 and 4.2.2 – Top-down world perspective.

Doing the collision in 2D space resulted in another issue, which were that the player would not be able to move up and down in the world. We did research this before we decided to use this approach, and realized that we could use height maps. A height map is a 2 dimensional array of height values for the world. When the player is standing at a specific location in the world, we read out of a height map table the height value for the player and set his height to the specified height. A graphical representation of a typical height map can be seen in figure 4.2.3.

With all this integrated, we had a system where we could do 2D collision detection in a top-down approach, while still having the player being able to go up and down in the world. A continuous check against the height map table sets the players height position to the appropriate height value and collisions against other objects in the world could be done in 2D.

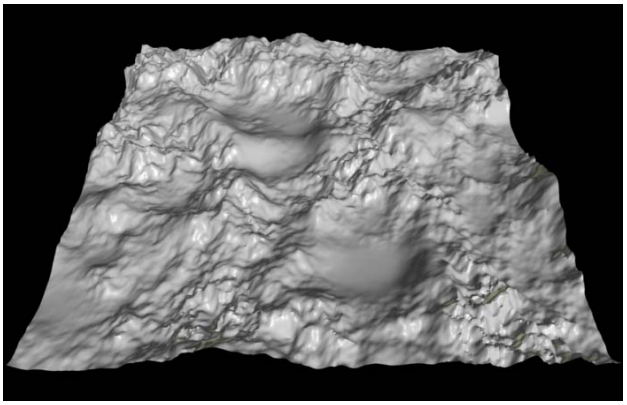


Fig 4.2.3 – Graphic representation of a height map.[4]

[4] Image from; http://en.wikipedia.org/wiki/Heightmap

The final issue we had when doing collision in 2D and using height maps was overlapping floors. If we had levels with overlapping floors, there would be 2 height values for the same position. There would also be an issue if there were walls above the player, which would result in a collision below the wall, as illustrated in fig 4.2.4 and 4.2.5. This was the main concern we had when choosing to do collision in 2D, but landed on a solution where we could still do everything this way, and still be able to have overlapping floors.

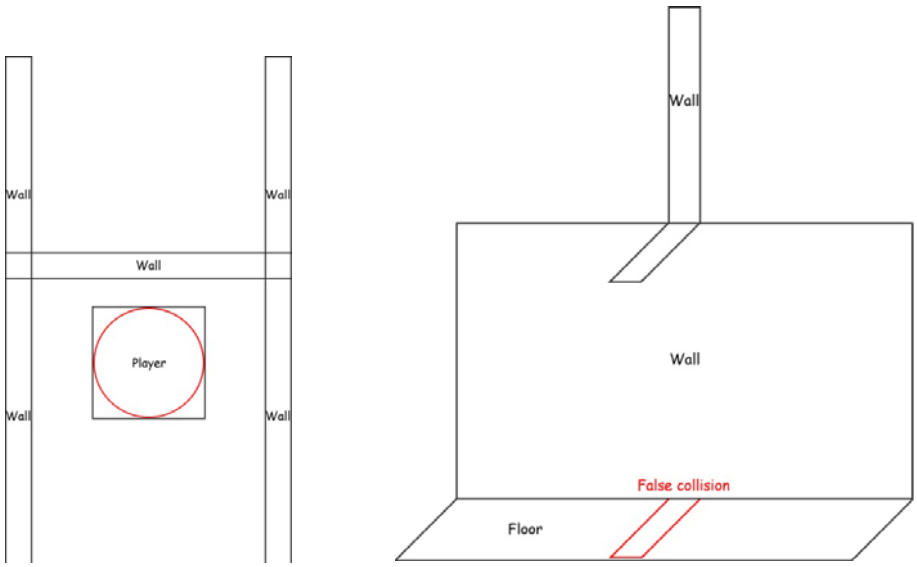


Fig 4.2.4 and 4.2.5 – Top down and side view of false wall collision.

The way we solved this was to have each surface that the player can be standing on, which represents the height map, store connections to other surfaces. The connection of surfaces is illustrated in fig 4.2.6. Also, we had the player store which surface that he was currently standing upon. This way we could limit the wall collision checks to only check against the walls that were on the surface the player was standing upon, and only use the height map value of that surface. And when the player moved onto a connected surface, the player's current surface would be updated to that surface.

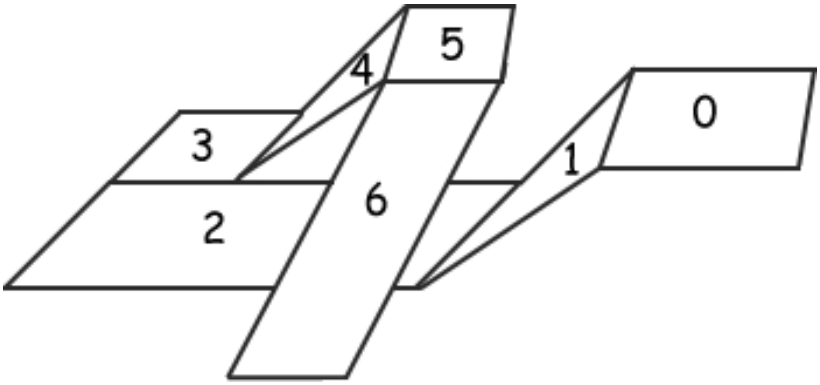


Fig 4.2.6 – Illustration of height map nodes with overlapping height maps.

It was a lot of work to make this world representation work the way we wanted, but we reduced the demand for processing power a great deal by doing the collision in 2D. The way the world is represented and the way the player moves in it is still approximately the same that it would be with 3D collision, but with fewer collision checks.

4.3 Height map

As mentioned earlier, we have chosen to use a height map to decide how high in the world the player is to be standing in our game. This is used instead of 3D collision detection with the floor, which would be much more demanding in terms of processing time.

The use of height map is simple. The height map has a position (X and Y values) in a 2D space, seen from a top-down perspective, as well as a value for the third axis which represent the height the player is to be standing on.

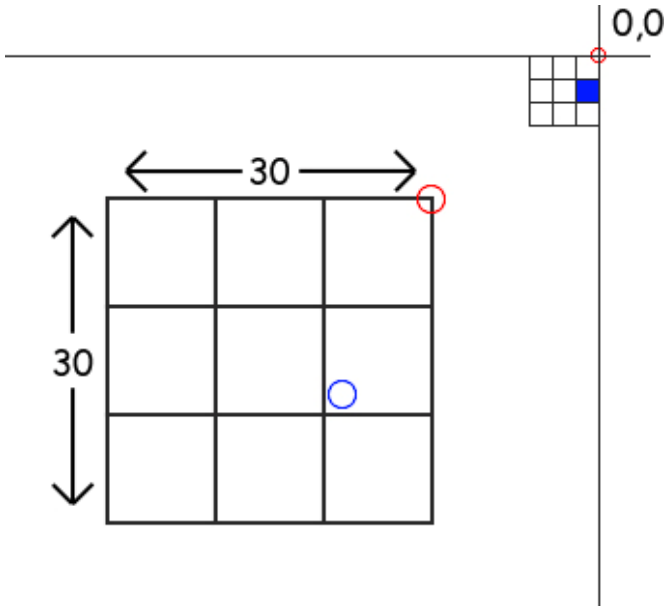


Fig 4.3.1 – Sample height map of size 30x30 units.

A floor surface is scaled down in a 1:10 scale, to reduce the size of the height map points and limit the memory usage for height maps loaded for the level.

A sample surface of size 30x30 will therefore have a height map of size 3x3. This sample surface is illustrated in figure 4.3.1.

The surface also holds a vector which describes the offset from world origin to where the surface is located.

When a height map value is to be set for the player, the player's position is translated to origin by using this vector and scaled down by 10 (scale of the height map). Then we get a position in the height map where the player is located and we can set the height of the player to the returned value from the height map table.

4.4 Collision

Collision points are calculated from finding out which walls are connected to a surface. The collision points are then stored in a 2D space, like the height map, but can be used in 3D since we will only have to check the points which are connected to the surface the player is standing upon.

This is done by creating a circle and a rectangle around the players (X, Y) position in 2D space as illustrated in figure 4.4.1.

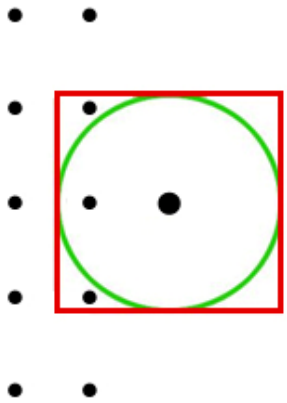


Fig. 4.4.1 – Player collision against 2D collision points.

When checking for collision, we check if any of the collision points is within the rectangle. Thereafter, for the points inside the rectangle, we check if the length vector from the player to the collision point is less than the radius of the circle. We check if the collision points are within the rectangle first to reduce the amount of collision points to be measured in length thereafter.

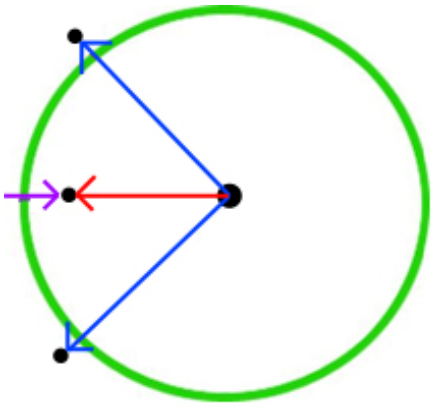


Fig. 4.4.2 – Pushback of player's position when a collision occurs.

If it is less, this means we have a collision. Then a pushback is done in the opposite direction of the collision vector. The amount of pushback is equal to the remaining length, which is the length from the collision point to circle radius. This is illustrated in figure 4.4.2, where the blue arrows represents no collision, the red arrow represent a collision and the purple arrow represents the pushback force applied.

4.5 The level editor

To both improve the time spent including content to the game, and to learn how to create a program known as a level editor, we decided it would be profitable to make one for our game.

A level editor is a simple program, shown in figure 4.5.1, where you can edit all the content of the game which is going to be shown in the actual game, without having to place objects by manually change values for positions, scale, etc.

In our level editor we are able to place walls and floors in a 3D space to create a level, which can thereafter be exported to formats that the actual game engine is able to read and create a world from.

The current version of the level editor is able to place walls and floors. We are yet to implement placing of other world objects and lights. A finalized version of the level editor would also include these.

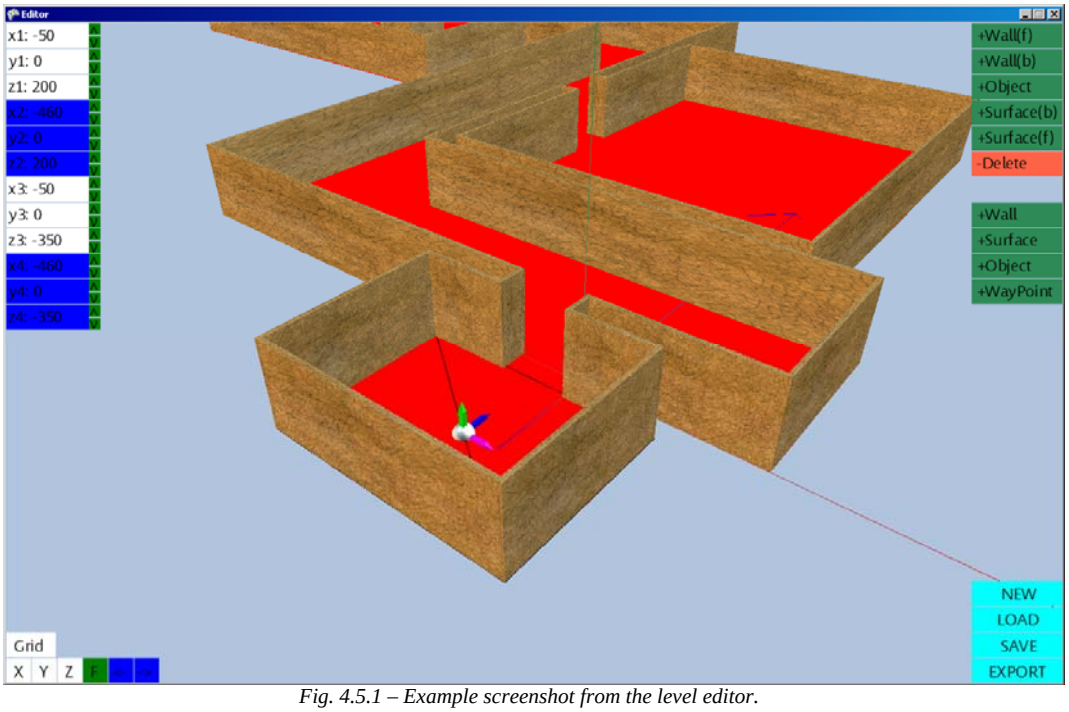


Fig. 4.5.1 – Example screenshot from the level editor.

4.6 Exportation from the level editor

Exporting a level from the level editor creates text files with values readable by the game project itself. We did not prioritize optimization of the exporting, but rather getting the right values exported, which makes the export currently very slow. For example, exporting the sample level in the project took about 30 minutes to export. This is due to calculating the height map, which is a very comprehensive task.

We also have at the moment a number of limitations to the exportation process, such as walls having to be connected to a surface for collision points for the wall to be generated. Also surfaces connecting each other cannot be overlapping each other. Overlapping surfaces will not get stored as connected, but must rather be placed on the direct edge of each other to become connected.

4.6.1 Exported files

The exported files are plain text files with values separated by the ";" and "|" characters.

For example, walls are defined by the following;
posX;posY;posZ | width;height;depth | textureName | boundingBoxMin(X;Y;Z) |boundingBoxMax(X;Y;Z)

So, a wall, with position in origin, with the length of 500, height of 200 and thickness of 10 would be defined as such;

0;0;0 | 500;200;10 | Walltexture | 0;0;0 | 500;200;10

4.6.2 Exporting collision points

Collision points will be added with a specific interval from the starting point of a wall to the end of the wall. This is done to reduce the number of collision checks to be done at a given time. Seeing that the collision check is done with a circle around the player, we only need enough points, so that the circle will be unable to pass through 2 following points. This is illustrated in figure 4.6.1.

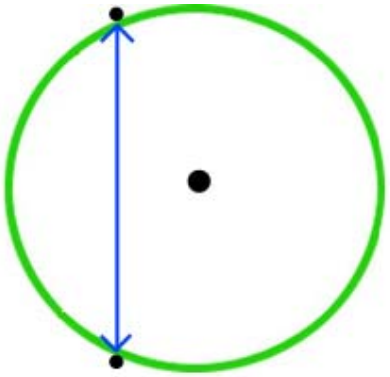


Fig. 4.6.1 – Collision points just close enough to prevent the player from passing through.

We have used an interval of 10 units, but this could probably be reduced even further, depending on the size of the player. Seeing that we used a placeholder model for the player, we chose not to perfect the values for the placeholder model, but rather keep the interval of 10 until we had a concrete model to work with.

When exporting the collision points, we first find the walls that are in connection with the surface in a 3D view. Thereafter we take these walls and generate the collision points in the form of 2D position coordinates along the wall.

These collision points are stored in connection with the surface it is in connection with, and will only be checked when the player is standing on that specific surface.

4.6.3 Exporting the height maps

The height maps in the game are represented by 2 connected triangle planes, making up a rectangle plane.

We have limited the plane to have 90 degree edges, because of calculation issues we ran into when trying to do the exportation with uneven edges. This does not otherwise affect the use of the height maps. An example of a surface being edited is shown in figure 4.6.2 and 4.6.3.

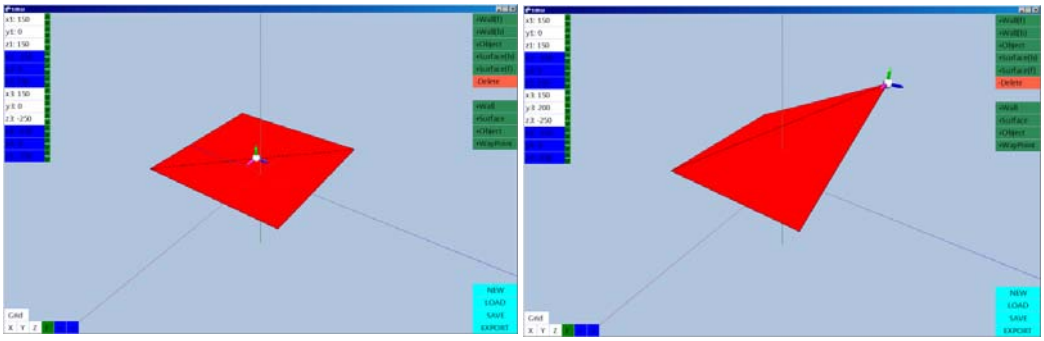


Fig 4.6.2 and 4.6.3 – A surface being edited in the level editor.

From this plane, we first calculate all the points which are inside the 2 triangles. This is the task in the exportation process that takes the longest, but we did not prioritize an optimization of the process, since it was giving the result we wanted even though it took a while to calculate.

When calculating the points inside the 2 triangles, we take a starting point in the edge of the triangle and create a unit vector to increase in each direction, and actually calculate the square that is made by expanding the triangle.

Thereafter we check if each point we receive is actually inside the triangle, which discards the points created from the whole square. We found this to be the most effective way to calculate the points inside the triangle.

Figure 4.6.4 shows the code process for calculating the points inside one of the triangles.

This is done with both triangles.

```
//Get triangle size
Vector3 t1widthVector = pos3point - pos4point;
Vector3 t1heightVector = pos2point - pos4point;

//Triangle size
float t1width = t1widthVector.Length();
float t1height = t1heightVector.Length();

//Unit vector
t1widthVector.Normalize();
t1heightVector.Normalize();

//Calculate and store the points inside the triangle
for (int j = 0; j <= t1height; j++)
{
  for (int k = 0; k <= t1width; k++)
  {
    Vector3 newVector = (t1heightVector * j) + (t1widthVector * k);
    newVector = new Vector3((int)(newVector.X),
                            (int)(newVector.Y),
                            (int)(newVector.Z));

    newVector = pos4point + newVector;

    Vector3 cpInside = Vector3.Cross((pos2point - pos3point),
                                     (pos4point - pos3point));
    Vector3 cpCheck = Vector3.Cross((pos2point - pos3point),
                                    (newVector - pos3point));

    if (Vector3.Dot(cpCheck, cpInside) >= 0)
    {
      trianglePointsList.Add(newVector);
    }
    else
    {
      break;
    }
  }
}
```

Fig 4.6.4 - Code for calculating the points inside a triangle in 3D space.[5]

After finding all the points, we generate a height map out of them. But, we did not want to use all of the points, since these results in a very large height map.

The way we chose to do it was to store every 10th value in the height map. Though, we realized later that the usual way to do this is to scale the height map, so that it fits into the computer processor's registry for fast access.

[5] Helping resource in figuring out the calculation method: http://www.blackpawn.com/texts/pointinpoly/default.html

4.7 Inventory screen

The inventory screen is where the player is managing all of the items he is carrying with him around in the game, from new items being picked up to already stored items.

The inventory system is built to be near realistic as to how the items are stored.
The player will have different set amount of slots as to what clothing or extra bags he is wearing at the time. For instance a shirt can have none, two or more pockets; each classified as a slot where items can be stored.

Slots have different size or storage capabilities, so large items like a gun cannot be carried in a chest-pocket but something smaller like pain-pills can. A gun would have to be stored in a holster around the leg or in the belt around the waist or simply be carried in the hand.

This way you would be looking at a near realistic way of storing items on a human body, in terms of how much storage room worn clothing could have.

In the inventory screen the player model wearing his clothing and bags are displayed with his viewable items on him. For instance if he is carrying a gun in his hand you would see him with the gun in his hand, if he has a shotgun strapped around his shoulder with the help or a rope it would be displayed like that. (This is a description of how it is meant to be, but is currently not implemented, few items are displayed on the player model as of now)

Managing the inventory is also done in a very realistic way, if a slot is selected the items in the slot is displayed in a circle around the player model, were the item in front of the player is the selected one, scrolling with the mouse wheel will cause you to scroll through the items and select a different item.
The selected item can then be dragged to a different slot, but only those slots which have storage room left or are big enough to carry the item.

The slots will be marked differently by a lighting effect depending on if it is the currently selected slot, if it is empty or full, if it has room for the item you are trying to replace or if the slot is valid for that item at all. (This is currently not implemented, but will be in retail release)

Selected items have multiple ways of interacting with them, for instance be examined in the inventory screen, were a description and item statistics will be shown in an info window, or combined with other items if possible. The game lets the player combine different items in the game as he likes, for instance if the player has a gun, a flashlight and some tape, he could tape the flashlight to the gun for more clever usage of both items.

The inventory screen also have four hotkey slots, were items put in these slot can be accessed fast in game by clicking the assigned hotkeys, instead of entering the inventory screen and manually put the item in the players hands for usage. This way pain-pills carried in a chest-pocket could dampen immediate pain by the push of a button. (This is currently not implemented, but will be in retail release)

Some of the inventory screen features are not working or yet implemented, but the game is still in the stage of development and the inventory system is a big part of the game. We currently have placeholders for models, so we lack various items like clothing, bags, guns, melee weapons and player models, and without some of these features cannot exist.

Development of the inventory screen

The inventory screen is viewed in Orthographic projection representing a three-dimensional object in two dimensions. Meaning all 3D objects and 2D objects are drawn in 2D space / screen coordinates, instead of 3D world coordinates. Objects size will this way not be affected by the Z-axis, only scale.

Sprites (which are 2D) are drawn first on screen to divide the inventory screen into different parts as content containers. One model background, item stats and info background, hotkey background and item combination background. The sprites were easy to place at screen coordinates, but the 3D models are a different story.

On top of these backgrounds, models, different slots and text are placed inside. The 3D models are placed in 3D space but drawn in 2D space, so to place the models accordingly were more difficult than with the sprites. The models had to be translated, rotated and scaled correct to each other to appear the right way.

For instance to make a gun appear in the player model hand, the scaling, rotation and transformation that is shown in figure 4.7.1, has to be done.

```
Matrix.CreateScale(itemScale) *
animationController.GetBoneAbsoluteTransform(boneName) *
Matrix.CreateScale(playerScale) *
Matrix.CreateFromQuaternion(playerRotation) *
Matrix.CreateTranslation(playerPosition);
```

Fig 4.7.1 – Scale, rotation and translation code.

Setting the world matrix to thease matrixes multiplied together will make the gun appear at the player models “boneName” cordinates.

Making boundingSpheres which is used for collision detection appear at the right places on the player model were the slots are supposed to be requires even more advanced world matrixes, with more translation, rotation and scaling. So this was the most difficult part to work out.

Fig 4.7.2 – The inventory screen.

Figure 4.7.2 shows a screenshot of the inventory screen, showing the placeholder player model to the left with a gun in his hand. The spheres on his body is boundingShperes placed at possible slot positions.

4.8 XNAnimation library

"XNAnimation is a skeletal animation library for XNA. This library allows developers to easily manipulate, playback, interpolate and blend animations." [6]

Controlling animations can be a difficult and complicated process with allot of things to keep in mind to make it work perfectly, controlling animations speed, cross fading from one animation to another ect. So we decided pretty early on in the project to be using the XNAnimation library to handle the animation part. The library seemed very solid, complete and easy to use, especially when it came to playing and controlling animations, but unfortunately we struggled a bit with the baked in shader to perform satisfying.

" XNAnimation main features:

- ***General***
  - *Easy to use.*
    - *Provides classes to handle skinned models, skeletons, animations clips, keyframes and poses.*
    - *Provides controllers to handle animation playback.*
  - *High performance.*
    - *Animation interpolation and blending made on the CPU.*
    - *Model skinning made on the GPU.*
  - *Low memory footprint.*
    - *Stores keyframe translation, orientation, and scale decomposed.*
    - *Shares skeletons, animations and meshes resources.*
  - *Supports Windows and Xbox platforms.*
- ***Animation playback***
  - *Plays animations forward and backward.*
  - *Controls animation speed and looping.*
  - *Supports models with up to 80 bones*
  - *Supports linear, cubic and spherical keyframe interpolation.*
    - *Stores keyframe translation, orientation, and scale decomposed*
    - *Interpolates translation, orientation and scale separately for best interpolation.*
- ***Animation blending***
  - *Supports cross fade between animation clips.*
    - *Allows smooth transitions between animations.*
- ***Material system***
  - *Supports multiple point light sources (up to eight) with light attenuation.*
  - *Supports materials with emissive, diffuse and specular properties.*
  - *Supports Diffuse, Normal and Specular textures.*
  - *Reports vertex and pixel shader profiles used*
- ***XNAnimation Model Processor***
  - *Splits any animation into a new set of animations (based on time or keyframe)*
  - *Transforms the entire scene using rotation or scale"* [6]

[6] http://xnanimation.codeplex.com/

In order to use the XNAnimation library it has to be included into the project, this is done by including a simple *.dll file to the projects properties. To use any of the XNAnimation library classes in any of the projects own classes the includation lines in figure 4.8.1 has to be included.

```
using XNAnimation;
using XNAnimation.Controllers;
using XNAnimation.Effects;
```

Fig 4.8.1 – XNAnimation library includes

Then an animationController has to be initialized to be able to controll and play the animations as you like, as shown in figure 4.8.2.

```
private void InitializeXNAnimationSetup()
{
  //Initializes a list of matrixes to hold the boneTransforms
  m_absoluteBoneTransforms = new Matrix[skinnedModel.Model.Bones.Count];

  //Copy the absolute transformation of each node into the
  m_absolutBoneTransform list. A list of how every bone is positioned
  acording to eachother by matrixes.
  skinnedModel.Model.CopyBoneTransformsTo(m_absoluteBoneTransforms);
  //Creates an animation controller. A class which controlles how
  animations are playd
  m_animationController = new
  AnimationController(skinnedModel.SkeletonBones);
  ...
```

Fig 4.8.2 – XNAnimation initialization code

An interpolation is a method for constructing new data points within a set range of known data points.

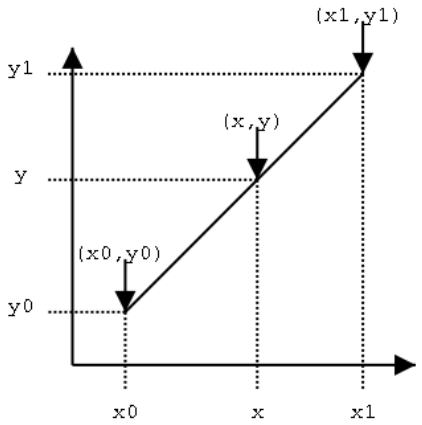


Fig 4.8.3 – Linear interpolation graph

In figure 4.8.3 a linear graph of interpolation between two points (x0,y0) and (x1,y1) is illustrated, and figure 4.8.4 shows the code required to enable this interpolation.

```
  ...

  //How translations are interpolated between animation clips.
  m_animationController.TranslationInterpolation =
  InterpolationMode.Linear;

  ...
```

Fig 4.8.4 – XNAnimation initialization code

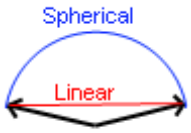


Fig 4.8.5 – Spherical interpolation graph

Figure 4.8.5 illustrates a spherical graph of interpolation between two points (x0,y0) and (x1,y1) and figure 4.8.6 shows the code required to enable this interpolation.

```
  ...

  //How orientations are interpolated between animation clips
  m_animationController.OrientationInterpolation =
  InterpolationMode.Spherical;

  //How scale are interpolated between animation clips
  m_animationController.ScaleInterpolation =
  InterpolationMode.Linear;

  //Start the first animation stored in the AnimationClips dictionary
  animationController.StartClip(clipNumber); //Idle animation
}
```

Fig 4.8.6 – XNAnimation initialization code

Now as the animationController is initialized we have to make sure it updates the animations so we get a good result. This is done by just putting the line of code shown in figure 4.8.7 into the XNA update loop.

```
...

m_animationController.Update(gameTime.ElapsedGameTime, Matrix.Identity);

...
```

Fig 4.8.7 – XNAnimation animationControllers update function

In a game you often want the animated models to use several animations not just one, for different moves and actions. So some kind of function which controlles what animation you want to play when what action is executed is very needed. For instance if you are just standing still you want the idle animation to play but as soon as you start running you want to play the running animation. And by using the crossFade technice in this library you can interpolate between two animations very easily so it looks clean and natural.

Figure 2.8.8 shows the code for the cross fading technique.

```
public void UpdateAnimationState(PlayerAnimationState animationState, int
speed, bool loop)
{
  m_animationController.LoopEnabled = loop;
  if (animationState == PlayerAnimationState.Idle)
  { //Controlls how to go from one animation to another, interpolating from
    the last bone stance to the first in the new one.
    m_animationController.CrossFade(skinnedModel.AnimationClips.Values[
    (int)PlayerAnimationState.Idle], TimeSpan.FromMilliseconds(speed));
  }
  else if (animationState == PlayerAnimationState.WalkingForward)
  { ...
```

Fig 4.8.8 – XNAnimation update animation state function

Figure 4.8.9 shows some code to explain how the library is used to draw animated 3D models.

```
//The function takes a model of type SkinnedModel and draws it
public void DrawAnimated3DModel(SkinnedModel model, Camera camera)
{
  //A for loop that loops through all the meshes the model has
  foreach (ModelMesh modelMesh in model.Model.Meshes)
  {
    //A for loop that sets effects on every modelMesh
    foreach (SkinnedModelBasicEffect effect in modelMesh.Effects)
    {
      ...
```

Fig 4.8.9 – XNAnimation draw model code

The SkinnedModelBasicEffect class contains the shader which came with the XNAnimation library and is used to directly communicate with the shader to set important graphic parameters to obtain the correct and best result.

First the cameras projection and view matrixes are sent to the shader, then different parameters are set like the models material, for instance metal often has a high specular reflection so naturally if we are to draw metal we would like the material to be specular by giving it specular values. But a specular material would be without meaning if the shader didn't know about lighting, so it has to be given information about what sort of light it is, where it is, does it hit the mesh that is about to be drawn and things like that.

This code is shown in figure 4.8.10.

```
...

      //Sets the shaders projection and view matrix with the correct
      matrixes from the camera
      effect.Projection = camera.m_projectionMatrix;
      effect.View = camera.m_viewMatrix;

      //Sets the animated bones to the model
      effect.Bones = animationController.SkinnedBoneTransforms;

      //OPTIONAL - Configure material
      effect.Material.DiffuseColor = new Vector3(1.0f);
      effect.Material.SpecularColor = new Vector3(0.1f);
      effect.Material.SpecularPower = 0.1f;

      //OPTIONAL - Configure lights
      effect.AmbientLightColor = new Vector3(0.4f);
      effect.LightEnabled = true;
      effect.SpecularMapEnabled = true;

      effect.EnabledLights = EnabledLights.Two;

      effect.PointLights[0].Color = new Vector3(0.6f);
      effect.PointLights[0].Position = new Vector3(100, 0, 0);
      effect.PointLights[1].Color = new Vector3(0.6f);
      effect.PointLights[1].Position = new Vector3(0, 0, 100);

...
```

Fig 4.8.10 – XNAnimation draw model code

And at the end the modelMesh is scaled, rotated and transformed to its correct position by setting the shaders world matrix, before drawing the mesh at the end of the code block as shown in figure 4.8.11.

```
        ...
        // Setup world transform
        effect.World = m_absoluteBoneTransforms[modelMesh.ParentBone.Index] *
        Matrix.CreateScale(scale) *
        Matrix.CreateFromQuaternion(rotation) *
        Matrix.CreateTranslation(m_position);
        }
    //Draws a model mesh with the set effects on it and the correct world
    transform matrix
    modelMesh.Draw();
    }
}
```

Fig 4.8.11 – XNAnimation draw model code

4.9 Adventure interaction

We wanted to include an adventure popup menu so the player could easily interact with objects in the game world. The interface we made is shown in figure 4.9.1.

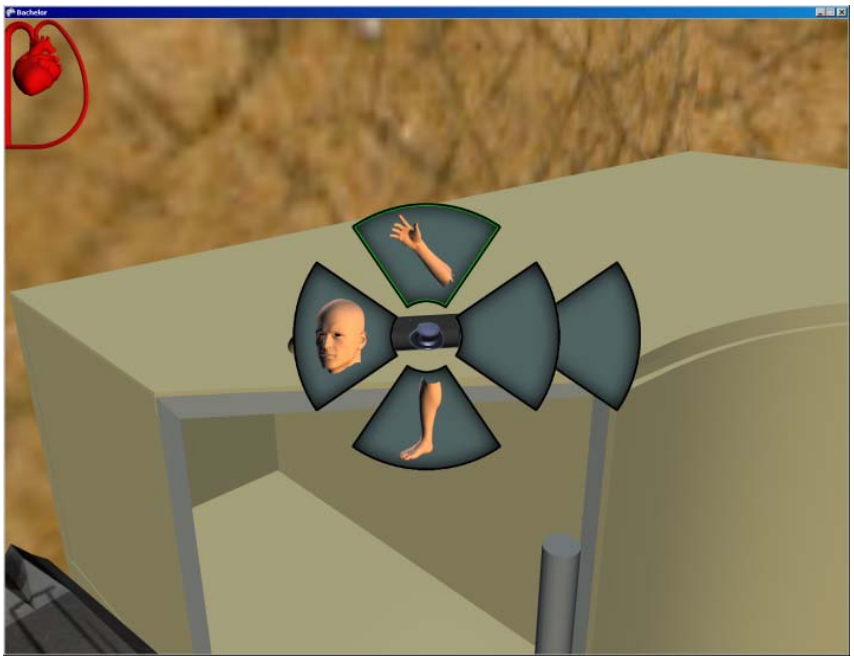


Fig 4.9-1 – The adventure menu.

This is a well known interface for interacting with world objects, and is also used in other popular games.[7]

The interaction menu includes the following;

- A hand - For picking up or using the object.
- A head / eye - For examining / looking at the object.
- A foot - For kicking or otherwise use your legs to interact with the object.
- Slots for the items the player is currently holding in his/her hands - For combining the item with the object.

This interaction menu was made in a separate class with a separate update loop. We chose to let the game world updating and drawing continue, so that interacting with an item in the game world would be done while the game still is progressing. The player must therefore be aware of elements and enemies surrounding him while interacting with objects.

[7] Other games using this interface; Condemned 2 (Monolith Productions, 2008), Turok (Propaganda Games, 2008).

The menu background itself is drawn in 2D space on the screen, and the 3D objects in the menu are drawn onto the screen in 2D space. To do this, we had to project a 2D position on the screen into 3D space, to be able to position the object correctly. The projection process is shown in figure 4.9.2.

```
public new Vector2 position
  {
    get { return m_position2d; }
    set
    {
      m_position2d = value;

      Vector3 nearsource = new Vector3(m_position2d.X, m_position2d.Y, 0f);
      Vector3 farsource = new Vector3(m_position2d.X, m_position2d.Y, 1f);

      Vector3 nearPoint = m_graphics.Viewport.Unproject(nearsource,
      m_projectionMatrix, m_viewMatrix, m_world);
      Vector3 farPoint = m_graphics.Viewport.Unproject(farsource,
      m_projectionMatrix, m_viewMatrix, m_world);

      Vector3 direction = farPoint - nearPoint;
      direction.Normalize();

      m_position = direction;
      }
}
```

Figure 4.9.2 – Projection of a 2D screen space into 3D space.

The projection calculates a 3D position 1 unit in front of the camera. We tried figuring out a way to calculate a position further from the camera, but had some issues doing this, so we chose to rather use the function that worked.

The drawback for using this was that a 3D object 1 unit in front of the screen had to be scaled down into a very small size, since it is placed so close to the camera. We solved this by adding a interface scale value for each object in the loading.

4.10 Scripting

We wanted to add scripted events for the game, both for events happening when a player passes a specific point in the game and for events starting when the player interacts with objects in the world.

This could be used for example for flickering lights, or lights that goes out when the player goes down a hallway, or for enemies appearing when the player turns opens a door or turns on the lights in a room.

We looked into a deal of possibilities for how to do scripted events during the development. There were some scripting engines we could have used, but looking through forums and discussions, we realized that a lot of these were outdated or incompatible with Xbox.

None on the group had any experience with creating a scripting engine, and thus developing a scripting engine from scratch at the point where we were to implement scripting in the game would be too time consuming, and we would most probably not get it working in time.

Therefore, we chose to use an approach where we could use the C# language and the game engine itself to make scripted events. The drawback from doing this was that we would have to recompile the whole project each time we wanted to change a script.

The way we did this was to create an “Event” class shown in fig 4.10.1, to store the data of the different events that runs as well as an “Events” singleton class which holds and updates the events that is currently running.

The “Event” class holds a function pointer to a function to run when the event get initialized and a function to run for updating the event.

```
public class Event
{
  public bool running;

  Func<GameTime, InteractiveObject3D, int> m_updateFunction;
  Func<int> m_initiateFunction;

  public InteractiveObject3D m_object;

  ...
```

Fig 4.10.1 – Event class stores functions to run when the event is initialized.

Adding a scripted event and linking it to a trigger event is done when starting the game. As an example, shown in fig 4.10.2, an event is made for a machinegun object for the “Hand” adventure interaction. It is then stored in the object, and is run by the “Events” class as shown in fig 4.10.3 when the interaction is made.

```
if (m_world.worldObjects[i].objectID == 1)
{
  Event machinegunHandEvt = new Event(MachineGun_Hand_Initiate,
  MachineGun_Hand_Update, m_world.worldObjects[i]);
  m_world.worldObjects[i].handEvent = machinegunHandEvt;
}
```

Fig 4.10.2 – Linking an event to an object interaction.

```
if (m_selection == Selected.top) //Hand interaction
{
  if (Events.Instance.StartEvent(m_interactiveObject.handEvent)
      == true)
    m_fadingOut = true;
}
```

Fig 4.10.3 – Starting an event when choosing hand interaction.

When an event is started, the event is added to a list of running events in the “Events” class, as shown in figure 4.10.4, and thereafter it updates each of the running events, as shown in figure 4.10.5. When the update-function in the event returns true, it is finished and is removed from the list of running events.

```
public bool StartEvent(Event evt)
{
  if (evt != null)
  {
    evt.Initiate();
    m_runningEvents.Add(evt);
    return true;
  }
  else
  {
    return false;
  }
}
```

Fig 4.10.4 – Start event function.

```
public void Update(GameTime gameTime)
{
  for( int i=m_runningEvents.Count-1; i >= 0; i-- )
  {
    if( m_runningEvents[i].Update(gameTime,
        m_runningEvents[i].m_object) == 1)
      m_runningEvents.RemoveAt(i);
  }
}
```

Fig 4.10.5 – Update events function.

4.11 Camera

4.11.1 Camera transition

When we got to the point of switching between the camera modes, first person, over-the-shoulder and 3rd person, we needed a function for this operation. We first came up with a function, the LinearTransition. This function allows the player to switch between the camera modes, but during the transition from one mode to another it is a bit "jumpy".

The code for a linear transition is shown in figure 4.11.1.

```
private void LinearTransition(float delta)
{
     float moveRatio = m_cameraTransitionSpeed;

     //CALCULATE MOVE VECTOR
     Vector3 moveVector = Vector3.Zero;
     Vector3 moveLookAtVector = Vector3.Zero;

     moveVector = m_targetPosition;
     moveLookAtVector = m_targetLookAt;
     moveVector = (moveVector - m_currentPosition) * moveRatio;
     moveLookAtVector = (moveLookAtVector - m_currentLookAt) *
                         moveRatio;

     //CALCULATE CAMERA POSITION
     m_currentPosition += moveVector;

     //CALCULATE LOOK AT POSITION
     m_currentLookAt += moveLookAtVector;

     if (moveVector.Length() < 0.1f)
     {
          m_inTransition = false;
          return;
     }
}
```

Fig 4.11.1 – Linear camera transition

Therefore we searched for other options on how to make the transition smoother.
We found the function called SmoothStep and this provided us with the functionality we wanted.

"The second method is SmoothStep, you invoke it the same way as Lerp, by passing a low, high and percent (0.0-1.0) value. The difference is in the value it returns. As it's name implies, the value steps down smoothly from the first value to the second using a cubic function. It's example is the blue line in the graphic.

SmoothStep(LowValue,HighValue,Percentage); "[8]

[8] http://johnnygizmo.blogspot.com/2008/10/xna-sidebar-smoothstep-and-lerp.html

We called the function BezierTransition and is shown in figure 4.11.2.

```
private void BezierTransition(float delta)
{
      DEBUG.Instance.PrintMsg("BEZIER", "348");
      m_bezierTime += (m_cameraTransitionSpeed * delta);

      if (m_bezierTime > 1.0f)
      {
           m_inTransition = false;
           return;
      }

      //CALCULATE THE NEW POSITION OF THE CAMERA
      m_currentPosition.X = MathHelper.SmoothStep(m_originPosition.X,
      m_targetPosition.X, m_bezierTime);
      m_currentPosition.Y = MathHelper.SmoothStep(m_originPosition.Y,
      m_targetPosition.Y, m_bezierTime);
      m_currentPosition.Z = MathHelper.SmoothStep(m_originPosition.Z,
      m_targetPosition.Z, m_bezierTime);

      //CALCULATE THE NEW LOOK-AT FOR THE CAMERA
      m_currentLookAt.X = MathHelper.SmoothStep(m_originLookAt.X,
      m_targetLookAt.X, m_bezierTime);
      m_currentLookAt.Y = MathHelper.SmoothStep(m_originLookAt.Y,
      m_targetLookAt.Y, m_bezierTime);
      m_currentLookAt.Z = MathHelper.SmoothStep(m_originLookAt.Z,
      m_targetLookAt.Z, m_bezierTime);
}
```

Fig 4.11.2 – Bezier camera transition

The Bezier transition uses a XNA function called SmoothStep, which provides a much smoother transition between the different camera modes, and this is the function we chose to use in the final project. Still we have the opportunity to switch between LinarTransition and SmoothStep for demonstrating the difference in between the different ways of transition.

Figure 4.11.3 shows an illustration of the increment when using SmoothStep (in blue) and LinearTransision (in red).

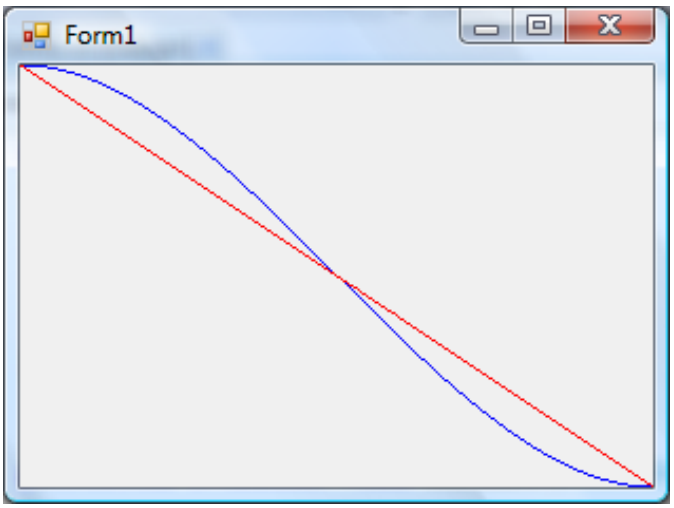


Fig 4.11.3 – A graph which describes smoothstep (blue) and linear (red) transition

4.11.2 Camera collision

A camera without collision detection going through walls would not look good and could be exploited by player's in-game, we do not want that so making the camera collide with walls and other objects make sense.

By shooting a ray from the player models head and towards the cameras original position you can detect if the ray intersects with any objects. If the ray intersects with anything you retrieve the ray's length and multiplied it with the ray's direction to get the new point where the ray is colliding. Then you set the cameras new position to be that new point on the wall, and there you have your new camera position.

This is a little simplified description, there is done some smoothing in and out from the original position to the new position to not make the camera seem "jumpy", but smooth in and out between these positions.

Figure 4.11.4 shows a graphic description of the concept.

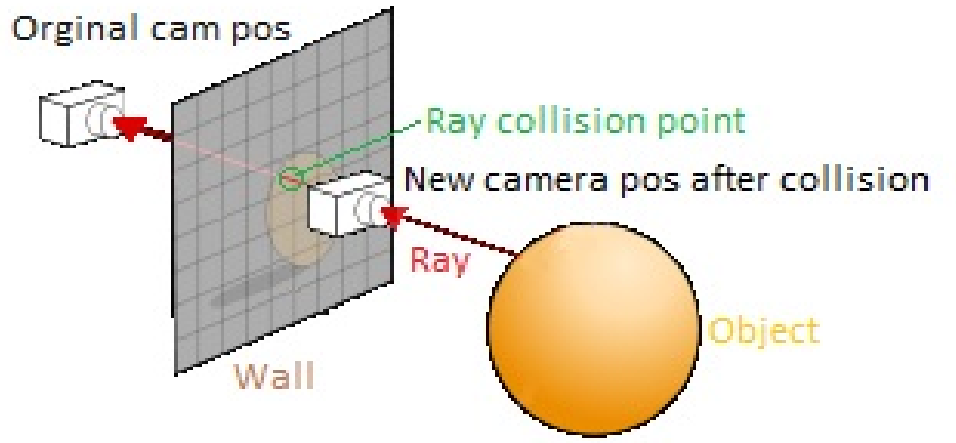


Fig 4.11.4 – Camera collision.

Currently the camera ray is checking versus every wall in a level, but that will be limited to nearby the player in retail to minimize checks for optimization. But what it is not doing is checking versus other in-game objects or the floor, but will of course be implemented when finalized.

4.12 Shaders

4.12.1 Shader implementations

"A shader is a set of software instructions, which is used primarily to calculate rendering effects on graphics hardware with a high degree of flexibility. Shaders are used to program the graphics processing unit (GPU) programmable rendering pipeline, which has mostly superseded the fixed-function pipeline that allowed only common geometry transformation and pixel shading functions; with shaders, customized effects can be used." [9]

The shader language used with XNA C# is High Level Shader Language (HLSL), developed by Microsoft for use with their Direct3D API. HLSL programs comes in three forms vertex, geometry and pixel shaders, for this project we have used vertex and pixel shaders.

"A vertex shader is executed for each vertex that is submitted by the application, and is primarily responsible for transforming the vertex from object space to view space, generating texture coordinates, and calculating lighting coefficients such as the vertex's tangent, binormal and normal vectors. When a group of vertices (normally 3, to form a triangle) come through the vertex shader, their output position is interpolated to form pixels within its area; this process is known as rasterization. Each of these pixels comes through the pixel shader, whereby the resultant screen color is calculated." [10]

For playing and controlling model animations in this project we have used an external XNAnimation library, with a customized shader. Unfortunately that shader does not perform to our standards.
So we either have to customize it to better suite our needs or replace it with a new one.

At first we tried to implement new features into that shader like light attenuation which determines a pointlights loss in intensity as the distance from the light increases, outside the light's range the models will be lit by the ambient light. This was more difficult than anticipated to implement, and we got bad results. Therefore we reverted to the un-customized XNAnimation shader before presenting the game at Hamar Game Challenge.

We also struggled a bit with using the shader from the XNAnimation library for the rest of the models.

In retail, the game will have new and improved shaders, giving the game perfect lighting to match the games dark theme. Also, it will include other new and improved effects, like improved particle effects and good texture representation.

[9] http://en.wikipedia.org/wiki/Shaders
[10] http://en.wikipedia.org/wiki/Hlsl

4.12.2 Wrapping of textures

When we first made a level for the game, we used one model for all the walls with different scaling. The problem with this is texturing and UV coordinates. With our current importer we cannot change the UV coordinates of a model after it is imported into the project.

As standard, the surfaces of a model have UV coordinates as shown in figure 4.12.1.

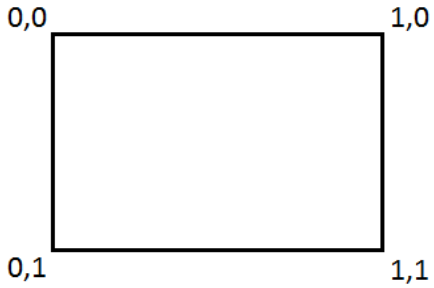


Fig 4.12.1 – UV coordinates

This will set the texture to cover the whole surface, and stretch or squeeze the texture to fit the surface. If the surface is wider than the texture we will want change the UV coordinates to repeat the texture across the surface. Example: If the surface is twice the width and half the height of the texture, the texture coordinates will be as shown in figure 4.12.2.

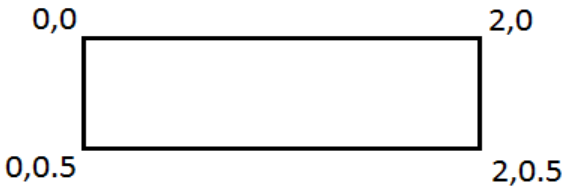


Fig 4.12.2 – UV coordinates for a scaled surface

With the texture sampler state set to wrap or mirror, the texture will be repeated twice horizontally and only halfway vertically.

So what we did was to make a “wall” class with a primitive to draw, as well as a bounding box for collisions. This primitive contains a list of all the vertices for a rectangle, where we could set the texture coordinates ourselves.

We have the scale of the model in the form of a Vector3.

To get the right UV coordinates, we used the values from the scale vector, and exchanged m_scale.X with m_scale.Z for the top and bottom surfaces of the box as shown in figure 4.12.3.

```
Vector2 frontTextureTopLeft =
new Vector2(m_scale.X / m_texture.Width, 0.0f);
Vector2 frontTextureTopRight = new Vector2(0.0f, 0.0f);
Vector2 frontTextureBottomLeft =
new Vector2(m_scale.X / m_texture.Width, m_scale.Y / m_texture.Height);
Vector2 frontTextureBottomRight =
new Vector2(0.0f, m_scale.Y / m_texture.Height);
```

Fig 4.12.3 – Code for setting the texture coordinates

Figure 4.12.4 shows the world with corrected UV coordinates, while figure 4.12.5 shows the world with incorrect coordinates.

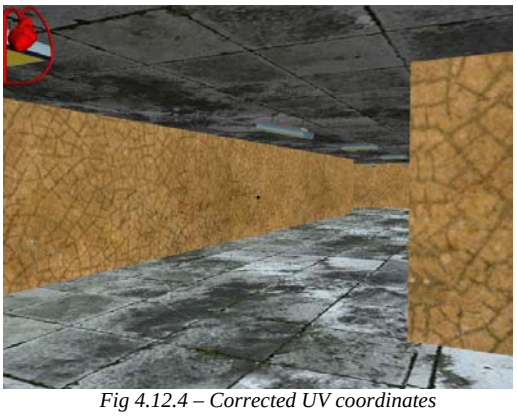


Fig 4.12.4 – Corrected UV coordinates

Fig 4.12.5 – Wrong UV coordinates

5 Testing and quality assurance

This chapter describes how the group planned on testing the product, how the testing and debugging actually was executed by the group.

We have also included a summary of how the group prepared for, and the performance and feedback from attending at Hamar Game Challenge.

5.1 Planned test-group

Originally we planned to give a high-school in Gjøvik the possibility to test and give feedback on the project game, but due to unforeseen delays in production of content, and the fact that we did not get to finish a fully functional game, we did not get the opportunity to carry this out.

The group did although test the alpha version we developed extensively while developing it, and we are aware of the issued and bugs we have in the product at the moment. Since we are aware of the issues and bugs, the use of a test-group to detect them, becomes obsolete.

The projects scope was admittedly larger than anticipated, so the game's state is behind schedule. Because the "game" is currently not a game, it has no story or elements of content that would classify it as a game. It is a game engine, and it is not finished.

The idea behind an engine is that it glues the game content together to act, react and look the way you want. So in other words as our game engine is not finished and we have no game content to put into the engine there has not been any point in having beta testers test the game and fill out surveys.

So the original plan, having beta testers test the game at least three times during the projects development, and filling out surveys had to drop out.

That aside most of the game engines features are working, but can be polished for optimal performance and better looks.

5.2 Testing and debugging

To improve the way we do debugging in the game, we decided to create an own class for this purpose.

The debug class includes a console, where we can modify settings used in the game in run-time. This makes debugging much easier than having to recompile each time we want to change a setting.

The console is shown in figure 5.2.1.

Fig 5.2.1 – The game console.

Also this class is implemented as a singleton class, which gives us the ability to print messages to the screen from everywhere in the code. This is useful for printing messages for collision, frame rate or different modes. An example debug message printout command is shown in figure 5.2.2.

```
DEBUG.Instance.PrintMsg("Position: " + m_position, "PlayerPosition");
```

Fig 5.2.2 – Example use of the debug class, which prints the player's position on the screen.

All debug messages added throughout the code is written to the screen as the last draw call, and then cleared for new debug messages to be added in the next run-through the code. How the messages get drawn to screen is shown in figure 5.2.3.

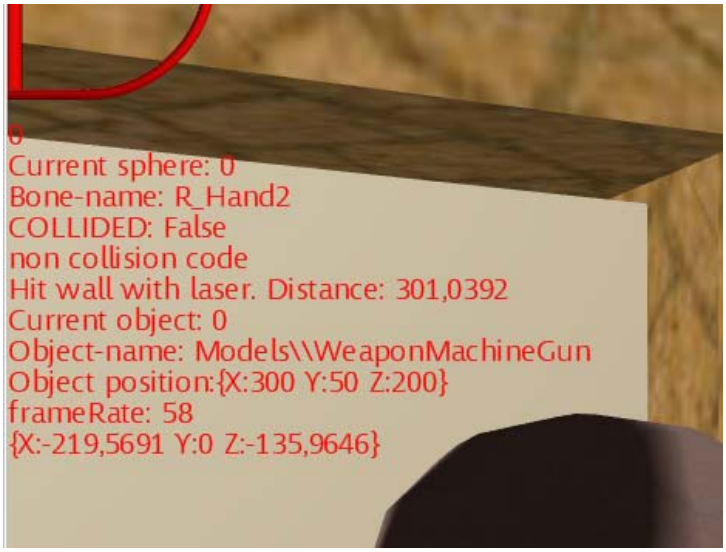


Fig 5.2.3 – Debug class example output values.

5.3 Hamar Game Challenge - Preparations & Execution

An important milestone for the project was the Hamar Game Challenge held on the 27th of April at Hamar College University. At this competition student projects was presented before a jury of business contacts, which in the end decided on one project to be supported and advised by "Kunnskapsparken Hedmark". They judged the project on the group's ability to sell the game as well as the realization chance of the idea, and the chance to actually gain an economic profit of it.

Before this presentation we prepared ourselves with a video of the alpha version of the game, as well as a presentation of our game idea. We had 10 minutes to pitch the game for the jury, followed by 5 minutes of questions.

Both video and presentation is included on the project DVD.

Only the top 2 projects were mentioned, and regretfully, we were not among them. We think this is because we are working on a large-scaled project with very limited resources. The judges also emphasized founding and economic profit of the project, and since our project requires external resources to be completed, our project were too demanding financial and time-consuming, counting the resources we currently have.

This presentation was focused on a realization realistic idea, as well as a good business idea. This project is on the other hand focused on learning programming, and thus, our project was not the best and most realistic business idea. We had a large-scaled and good idea, but were too demanding resource-wise.

Regardless, we are pleased with the presentation at Hamar Game Challenge and had great feedback from the audience during the intermissions.

6 Installation

This chapter describes how the program is installed and executed, as well as requirements for being able to run the game and use the developed project code.

6.1 Execution of the release version

The release version of the game can be found on the CD attached to the report, including everything else needed to run the game.

To execute the release version of the game it is required to have the XNA Framework Redistributable 3.1 installed on the computer beforehand. When the requirement is fulfilled the retail version can be started by executing the bachelor.exe file.

6.2 Browsing the project solution

To begin coding or debugging the project it is required to have Microsoft Visual Studio and XNA Game Studio 3.1 installed on the computer beforehand.

[27] Guidelines to creating a new project for new users can be found at this URL: http://creators.xna.com/en-US/create_detail

The projects solution can be found on the CD attached to the report, but not the required programs to run it. [22] Programs needed to run the solution can be found online at this URL: http://creators.xna.com/en-US/downloads.

To examine the projects code or start debugging, execute the projects *.sln file which is known as its solution file. Once Visual Studio has started and opened the project, it can be debugged with the F5 button on the keyboard or by clicking the play icon on the GUI. Inside the Visual Studio GUI a file list is shown either on the left or the right side depending on the GUI layout where all of the projects files can be browsed and edited.

7 Development process

This chapter describes what kind of development process we planned for this project and how we used it.

7.1 Our usage of the scrum development process

In the development process of this project we used the SCRUM framework, scrum is an iterative incremental framework for managing complex work (such as new product development) commonly used with agile software development.

Every second week period the group has had a SPRINT meeting, were the groups past two weeks and next two weeks have been discussed and planned for, operating almost as minor project milestones.

Our supervisor has been closely involved throughout the whole development process, as he has seen the work in progress almost every day and not just at the sprint meetings. The incremental prototypes meant to be shown at every sprint meeting blended into the normal work flow as we always had a working prototype. Our supervisor did not either see it necessary to display an incremental prototype every sprint meeting.

Scrum enables the creation of self-organizing teams by encouraging co-location of all team members, and verbal communication across all team members. Scrum relies heavily on verbal communication so daily scrum meetings have been held the first 15 minutes of every day. At these daily scrum meetings the previous day's work is discussed and the current days work is being planned.

A key principle of Scrum is its recognition that during a project, modules can easily be change for different purposes, and unpredicted challenges can as well easily be addressed. As such, Scrum adopts an empirical approach-accepting that problems cannot be fully understood or defined, focusing instead on maximizing the team's ability to deliver quickly and respond to emerging requirements.

There are several implementations of systems for managing the Scrum process, which range from yellow stickers and whiteboards, to software packages. One of Scrum's biggest advantages is that it is very easy to learn and requires little effort to start using.

7.2 Work distribution

Our original Gantt-chart described the work distribution amongst the group members very clean and nicely. We have tried following the Gantt-chart to the best of our abilities, but it came to our knowledge at a few of our sprint meetings, that some of our modules / tasks time constraints would have to be extended or decreased.

8 Discussion of results

This chapter concerns the results from the discussions the group had in advance of, and during the development of the project. The most significant discussions were about XNA, 2D collision detection and height maps. The chapter also mentions the criticism of the thesis which talks about what the group should have done different or better.

8.1 Discussions / debates

8.1.1 Results

XNA

Reasons for choosing XNA as tool for this project was it is simplicity for creating a game. XNA has the foundation for creating a game, such as collision, importing models, textures etc.
Since our project was of a much larger scale than previous projects we had done, we knew XNA would give us much more time focusing creating a good interaction game. As the group encountered many obstacles throughout the process, we would have to spend the double amount of time on the same problems without XNA.

With XNA, we also knew when, we were to attend Hamar Game Challenge that our project could be made into a game which runs on the Xbox 360, as well as a PC. And the Xbox live provides a simple way of distributing the game for profit, which was an advantage for our group, when competing at Hamar Game Challenge.

Disadvantages when using XNA is that we do not have a full low-level access to the machine, but working through a managed API. This would be a problem for developers who are creating a game of a very large scale and advanced graphics and technology. For our project this will not be a problem, since we do not need that freedom to create a good interaction game.

2D collision detection

Collision detection between the different objects in the game world we decided that a 2D collision detection would be proficient for our game. The 2D collision is simpler than 3D collision, it provides the same visual effect and our project do not require 3D collision. The 2D collision was used to detect collision between the player and walls or other objects placed in the world, as described in section 2.3.

Height maps

When we were designing the game world we encountered the issue on how make player aware of where he/she is vertically in the world.
We could choose if we wanted to create a 3D collision detection or create height maps. Since we already had decided to use 2D collision in our game and after discussing the usage of height maps with Simon, we got the idea of how to make height maps and how to use them in the level editor. Description of the height maps is found under section 2.3 in this document.

8.2 Criticism of the thesis

The projects scope was too large, for a group of four programmer students with no external resources available to finish in approximately three months.

A commercial game company with 100 employees would most probably use two years to finish a game like Mythophobia from scratch to retail. They would use a lot of work hours performing tasks such as designing game characters, modeling, creating environments, writing AI (Artificial intelligence), scripting game events, creating a new game engine and polish the story.

Since we were four programmers working on this project, which did not have much experience with large-scaled projects and less experience with 3D modeling, we surely could not finish the project in three months.

We also underestimated time consumption on various tasks, meaning they took longer time to finish than anticipated, and therefore pushing other important tasks aside, or to be finished later in even less time.

8.3 Further work, new bachelor task(s)

The group has discussed the further development of this projected after graduating from Gjøvik University College. The continuing development would involve making the game portable to XNA live and thereby playable on a Xbox as well as PC. But more importantly develop a game based on the game engine we now have. With more time and increasing the number of members in the group with the same background and goals as the original group, the project could develop much further. We would then go on to implement all the content we imagined before we started on the project.

The possibility to found a company after further development exists, but would require external funding, so finding investors is necessary before taking the development even further. Unfortunately we did not win Hamar Game Challenge and the chance of being accepted into an incubator where would receive assistance and funding for founding a company.

The potential of this project is present, but if we were to continue the development we depend on everyone agreeing on giving up more of their spare time and work even harder and that everyone willing to maximize their effort and improving their skills to a higher level.

9 Evaluation of group work

This chapter describes our group work with our goals and expectations for the project.
We will talk about us as a group, what routines and rules we put down, how we divided the work amongst the group members, our experiences on working with group projects and what we think of our experience with working on a bachelor project.

9.1 - Introduction

We are a group of four game programmers taking the game programming degree at the Gjøvik university collage. In December the end of 2009 we all had to choose a bachelor assignment for completing our bachelor degree. Between the ten bachelor assignments we could choose from, the Novel Interaction assignment Mythophobia - were the most appealing one to the four of us.

Myhophobia was also the game idea of our group leader Lars Inge, who had applied it for validation as a bachelor assignment, and got it approved. Three of the group members had also worked on a game together using XNA in a previous assignment. So the most of us already had a pretty good group feeling and work flow going, and decided to continue so with the addition of one group member. This made us feel more confident and secure that the bachelor assignment would turn out great.

9.2 Routines and rules

- We all agreed upon having a minimum demand of 35 hours of work each week on the project. This would equalize 5 hours each day, including weekends, or extended workdays beyond 5 hours in the weekdays.
- Everyone has committed logs of their work at the end of each day, on the project's web-page. Describing what they have been working on and the amount of hours spent working that given day.
- Since we decided upon using the SCRUM development method, we have had 15minute SCRUM meetings almost every day at room A112. Were we have gone through each group member's work, their progress and documents that would be done next.

This subchapter's text has been lent from the pre-project report with some editing, to give the reader an insight into the group's structure and how the group has organized itself.

9.3 Work distribution

Our group of four was distributed and divided into two small "teams" working two and two together on separate module/task in cooperation but with relevance to the project. Our work distribution is shown clearly in our planed gantt-chart, but unfortunately the real gantt-chart doesn't always reflect that.

We divided up to cover more modules/tasks at the same time, as working on the same thing seemed to us like over commitment and bad allocation of resources. The other thing is that SVN works best if not everybody works on the same thing at the same time, having 4 programmers editing the same code at the same time always lead to errors.

Each member was given explicit tasks aside from the general programming, as referred to in the previous structure chapter. Were each member also had to focus and fulfill those tasks at the same time.

9.4 Project as a work form

We have all worked on many group projects throughout our three years as game programming students and we are all in the same class, so we have pre experience with working with one another and group projects.

Working on a big project like this one has been a new and positive experience for all of us, choosing to go with the SCRUM framework made communication and workflow work very well for us. We have met almost every day in the meeting room A112 at Gjøvik university collage, had daily SCRUM meetings, discussed our work and helped each other really well this way by keeping the group gathered throughout the whole project.

Projects can be very big and feel very inconsistent at times, and we all know that this project is a small one compared to other major projects big game development companies can have. But still most of the time we felt that we had our project under control, and we all think it worked out just great.

9.5 Subjective experience of the bachelor thesis

Up until this project we have only worked on smaller and minor projects compared to this one, but having a project as a work form was not unfamiliar to us beforehand. Throughout this project we have all acquired better project management and planning skills, and are now better suited for bigger projects in the future. Our ability to work in a group and have teamwork has also improved.

We have had a very positive experience with this bachelor project, it has been a very learning and educating process for us all to be part of a group and work on a big project from the start to end.

All the way from planning ahead, choosing the correct work form, to the actual game programming and the end bachelor report. We have worked hard and tight in the meeting room A112 almost every day except weekends and eastern holiday for long work hours, some days up to 12 hours a day, and it still was very fun to us.

We struggled with some tasks we had but most of them we sorted out and finished with hard work by discussing the problems between us, studying online tutorials, accessing forums asking questions, and consulting Simon McCallum our teacher and in general being active.

In the beginning of the project we were very optimistic and eager to do a good job and get far, and possibly win the Hamar Game Challenge contest. But as the projects end drew near we acknowledged that the projects scope had been a little too optimistic for us to get a game up and running in our small time constraint, at most we could be looking at a game engine and as work in progress.

We all think this bachelor project has been a great asset to us all in preparing us for actual employment out there in the real world.

There has been neither lack of tips and enthusiasm from our teacher and supervisor Simon McCallum, in helping us with the project. He has also shown us great interest in the project and encouraged us throughout to always do a good and satisfying job.

10 Conclusion

This chapter discusses the end results of the project, the goals which the group succeeded, or failed to reach and the final product outcome.

This project was originally Lars Inge's vision and idea which he came up with three years ago when he started at Rena University College. He applied this idea as a bachelor project assignment at Gjøvik University College. We all chose this project to be our bachelor assignment in December 2009. The group was formed and we immediately started developing the project early January 2010.

When we started the development we had a vision about making an entertaining and functional horror/ survivor novel interaction game, with great effects and technical features found in games we all have come to like in the past.

The project has come to an end, yet the development of the game can still continue in the future, and the vision we had for this game was not realized. The outcome of this project is not a game, because the product does not contain elements which would classify it as an actual game, but rather a game engine. Although we had a vision of a game and ended up with a game engine, we can proudly state that we did a good job. The amount of time and resources we had limited our outcome for this project, with more time, resources and a larger team of game developers the vision could have been realized.

We learned a lot about our current limit as programmers and now we can know how much effort, time and resources it requires for creating a five star, A- list game. We as a group feel we got about as far as four game programming students with our proficiency and background could have gotten with this project in its time constraints.

We believe the producer and supervisor have more respect and worthy impression of us after displaying our skill as programmers and that they are pleased with our accomplishments and the product we have made.

If the time was not an issue we are certain the development of this game would continue and in the end become the vision we had in mind. Our opinion is that the vision behind this game has a great potential and we have discussed further development. We would love to implement the elements that would classify our product as a game, such as enemy AI, improve graphics, the story we have and evolve it further, create all the characters and npc's for the game, and have new models. With the time and a larger team of developers, this vision could become true and having our own game being sold next to other A-list games. could become true and having our own game being in sold next to other A-list games.

Resources

[1] Riemer's XNA tutorials, general XNA programming and shader tutorials
http://www.riemers.net/eng/Tutorials/XNA/Csharp/series1.php

[2] XNA Creators Club
http://creators.xna.com/en-US

[3] XNA Creators Club Forums
http://forums.xna.com/forums/

[4] XNA Creators Club Education section
http://creators.xna.com/en-US/education/

[5] XNAnimation library homepage
http://xnanimation.codeplex.com/

[6] Gamedev homepage, Articles and Resources
http://www.gamedev.net/reference/

[7] Stackoverflow article
http://stackoverflow.com/questions/328107/how-can-you-determine-a-point-is-between-two-other-points-on-a-line-segment

[8] Blackpawn, points in triangle test article
http://www.blackpawn.com/texts/pointinpoly/default.html

[9] Brage Bibsys, for looking up old bachelor reports for learning purposes
http://brage.bibsys.no/hig/

[10] Dark Codex Studios, shader tutorials
http://digierr.spaces.live.com/

[11] Wikipedia, HLSL – High Level Shader Language
http://en.wikipedia.org/wiki/High_Level_Shader_Language

[12] XNA Forum, thread we posted about: World matrix problem
http://forums.xna.com/forums/t/48514.aspx

[13] XNA Forum, thread we posted about: XNAnimation library question
http://forums.xna.com/forums/t/47963.aspx

[14] LaTeX Software Homepage
http://www.latex-project.org

[15] Visual Paradigm Software homepage
http://www.visual-paradigm.com

[16] Wikipedia, Heightmap
http://en.wikipedia.org/wiki/Heightmap

[17] XNA Sidebar – Smothstep and Lerp
http://johnnygizmo.blogspot.com/2008/10/xna-sidebar-smoothstep-and-lerp.html

[18] Wikipedia, Shaders
http://en.wikipedia.org/wiki/Shaders

[19] Gamespot homepage, for game rankings, sale numbers and market research
http://www.gamespot.com/

[20] Vgchartz homepage, for game rankings, sale numbers and market research
http://www.vgchartz.com/

[21] Microsoft Visual Studio
http://msdn.microsoft.com/en-us/vstudio/default.aspx

[22] XNA Game studio downloads
http://creators.xna.com/en-US/downloads

[23] 3D Studio Max 2009, for model and animation creation and exporting
http://usa.autodesk.com/

[24] Adobe Photoshop, for image and art creation
http://www.adobe.com/no/products/photoshop/photoshop/

[25] Adobe After effects, for video editing and creation
http://www.adobe.com/products/aftereffects/

[26] Microsoft Office, for documentation creation
http://office.microsoft.com/en-us/default.aspx

[27] XNA Creators Club, creating a new project
http://creators.xna.com/en-US/create_detail

[28] Doxygen, program for autogenerating code documentaion
http://www.stack.nl/~dimitri/doxygen/

[29] 3D Archives, free 3D models
http://archive3d.net/

[30] 3D xtras, free 3D models
http://www.3dxtras.com

[31] Quality 3D models, free 3D models
http://www.quality3dmodels.com

[32] Free 3D models
http://gfx-3d-model.blogspot.com

A Terminology

A **shader** is a set of software instructions, which is used primarily to calculate rendering effects on graphics hardware with a high degree of flexibility.

UV mapping is the 3D modeling process of making a 2D image representation of a 3D model.

UV coordinates are 2D coordinates that are mapped onto a 3D model.

Direct3D is a 3D API. Direct3D is used to render three dimensional graphics in applications where performance is important, such as games.

Application Programming Interface (**API**) is an interface implemented by a software program which enables it to interact with other software.

OpenGL (**Open G**raphics **L**ibrary) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics.

A **Bezier curve** is a parametric curve frequently used in computer graphics and related fields.

A **translation** is moving every point a constant distance in a specified direction.

FPS can have two meanings **F**irst **P**erson **S**hooter, and **F**rames **P**er **S**econd rendered from the video card onto the computer screen.

Polygonal modeling - Points in 3D space, called vertices, are connected by line segments to form a **polygonal mesh**

A **Model** consists of one or more meshes.

Scaling is a linear transformation that enlarges or increases or diminishes objects.

A **rotation matrix** is any matrix that acts as a rotation in 3D space.

I**nterpolation** is a method of constructing new data points within the range of a discrete set of known data points.

In geometry, a **vertex** is a special kind of point which describes the corners or intersections of geometric shapes. Vertices are commonly used in computer graphics to define the corners of surfaces (typically triangles) in 3D models, where each such point is given as a vector.

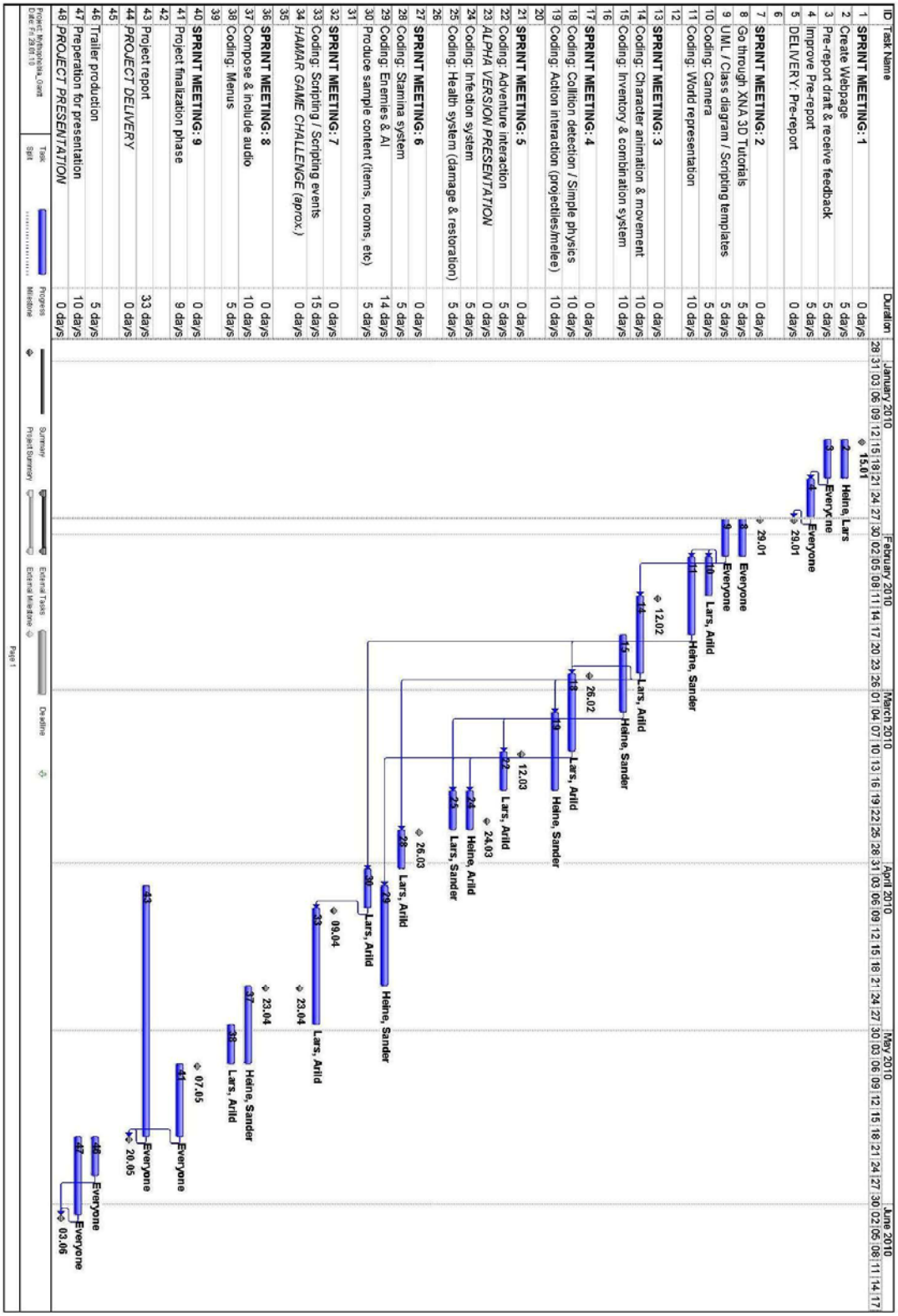
In computer graphics, a **heightmap** or heightfield is a raster image used to store values, such as surface elevation data, for display in 3D computer graphics.

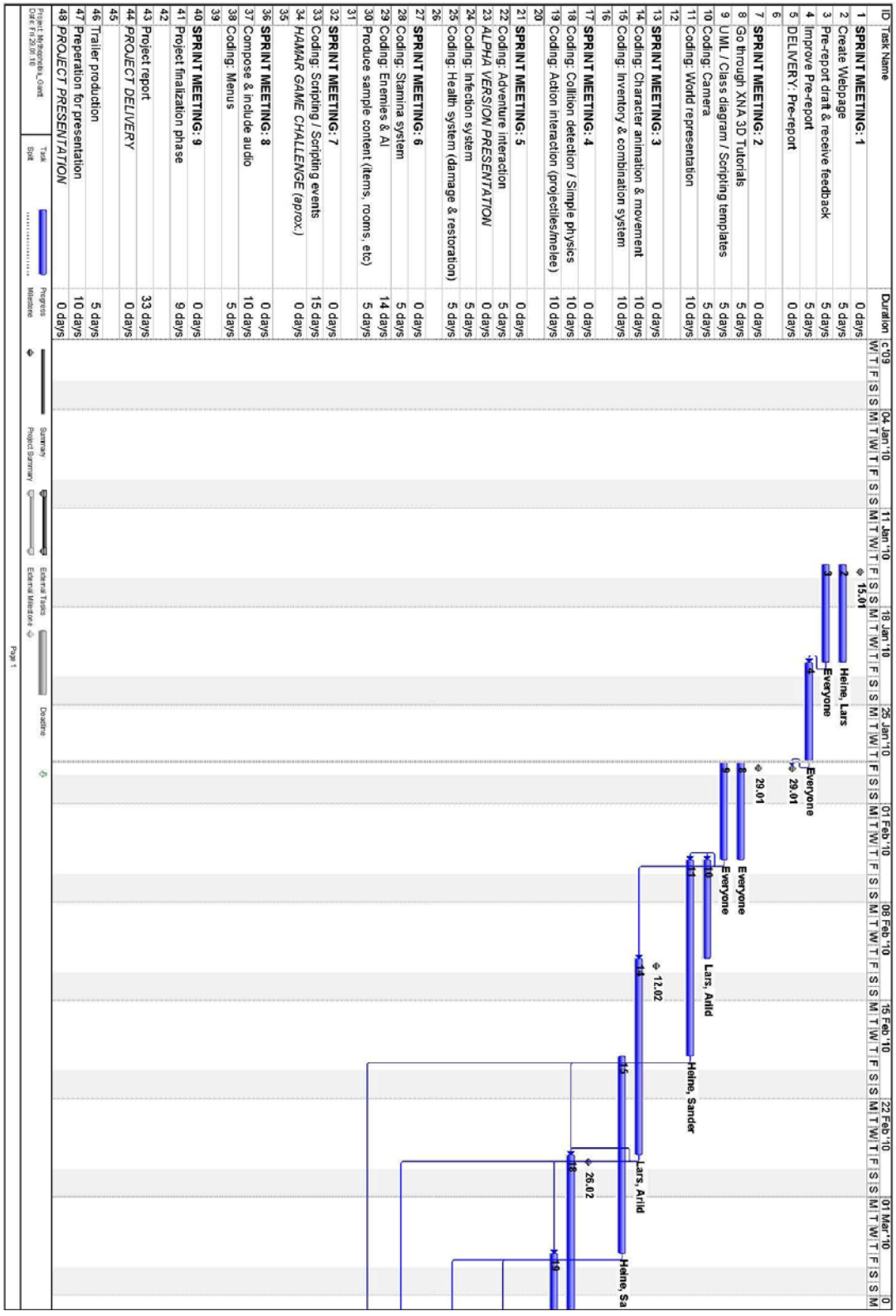
FBX 3D studio max model exporting format.

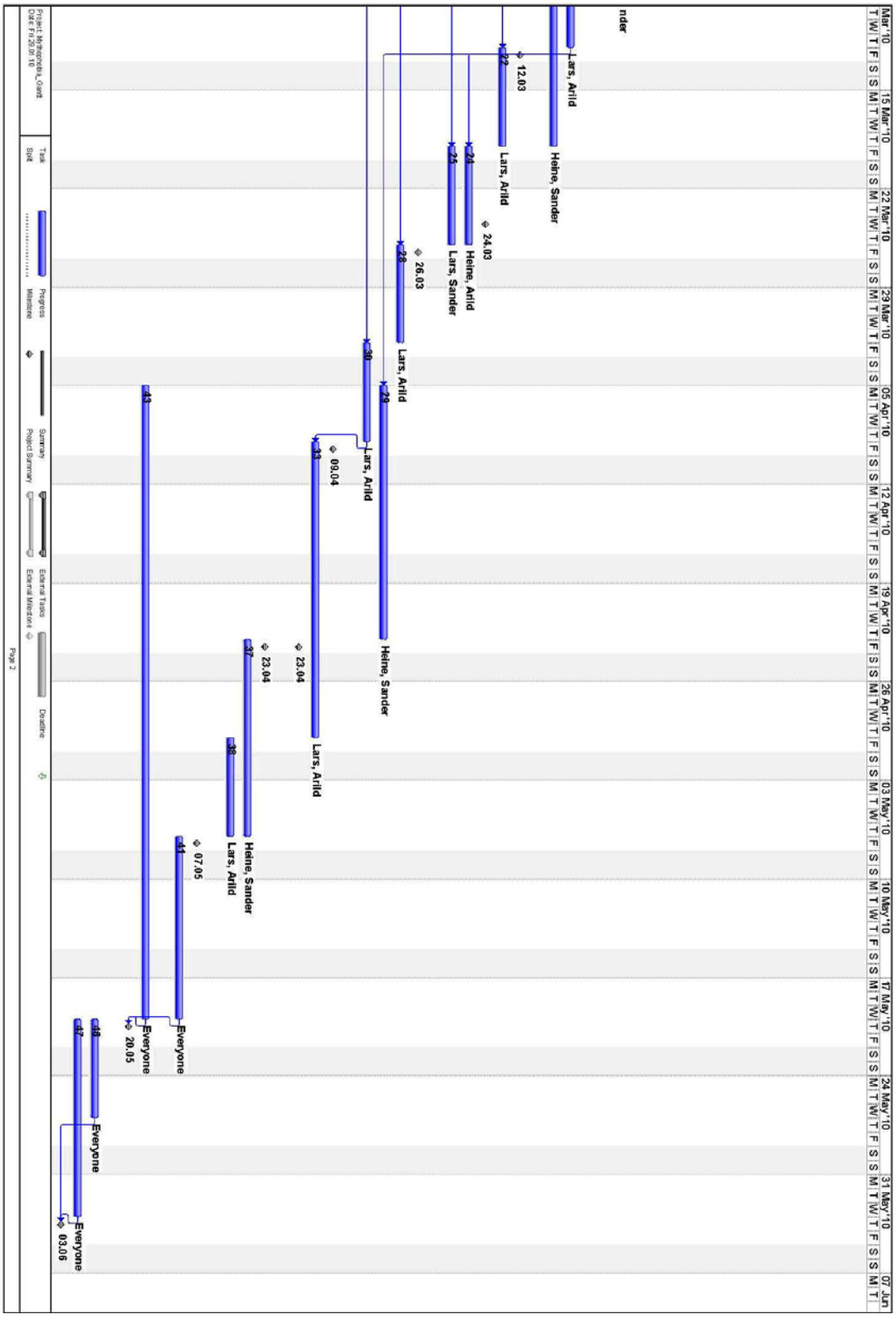
A **Graphical User Interface (GUI)** is a type of user interface item that allows people to interact with programs in more ways than typing such as computers.

A **Ray** is part of a line which is finite in one direction, but infinite in the other. It can be defined by two points, the initial point, A, and one other, B.

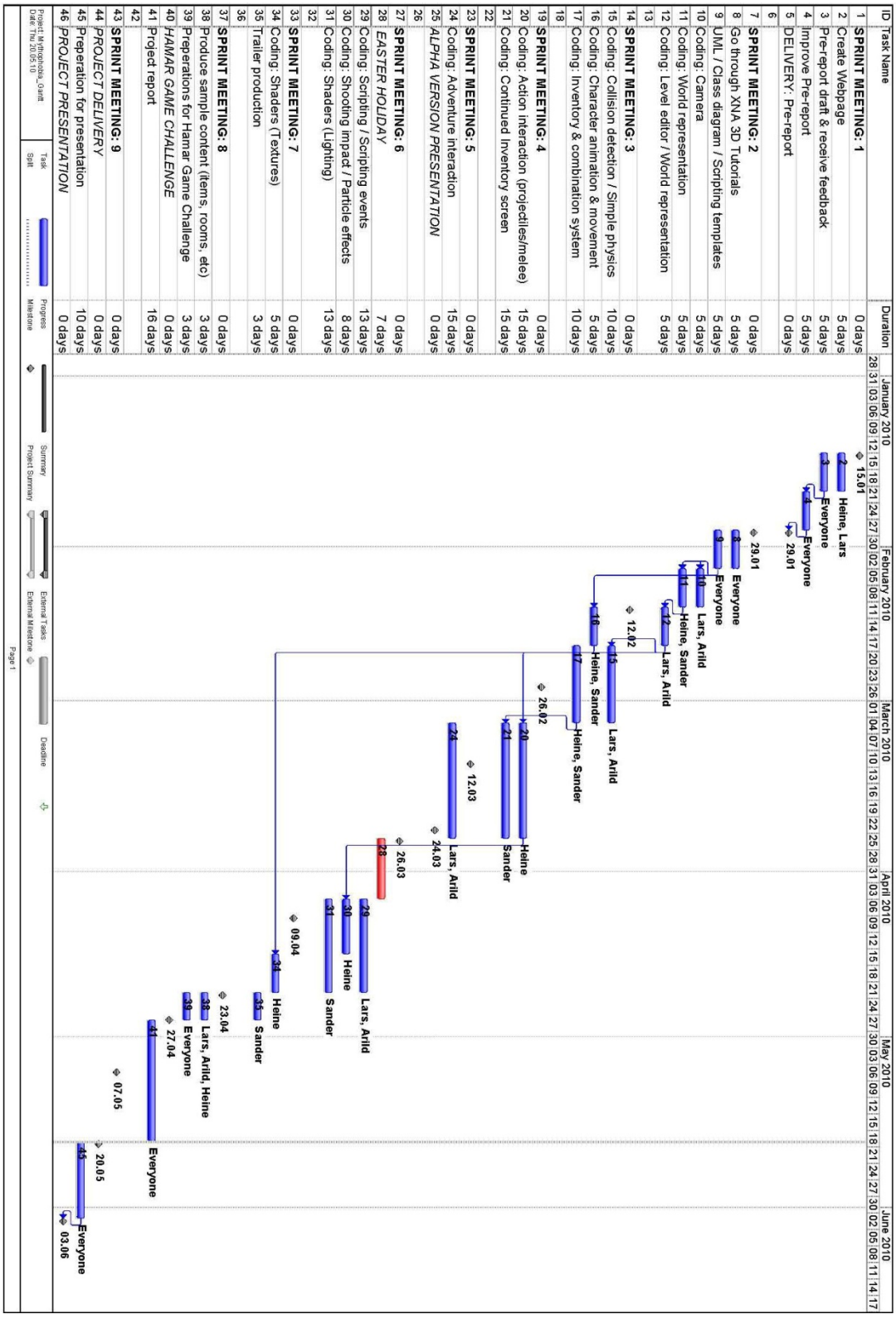
B Original Gantt chart

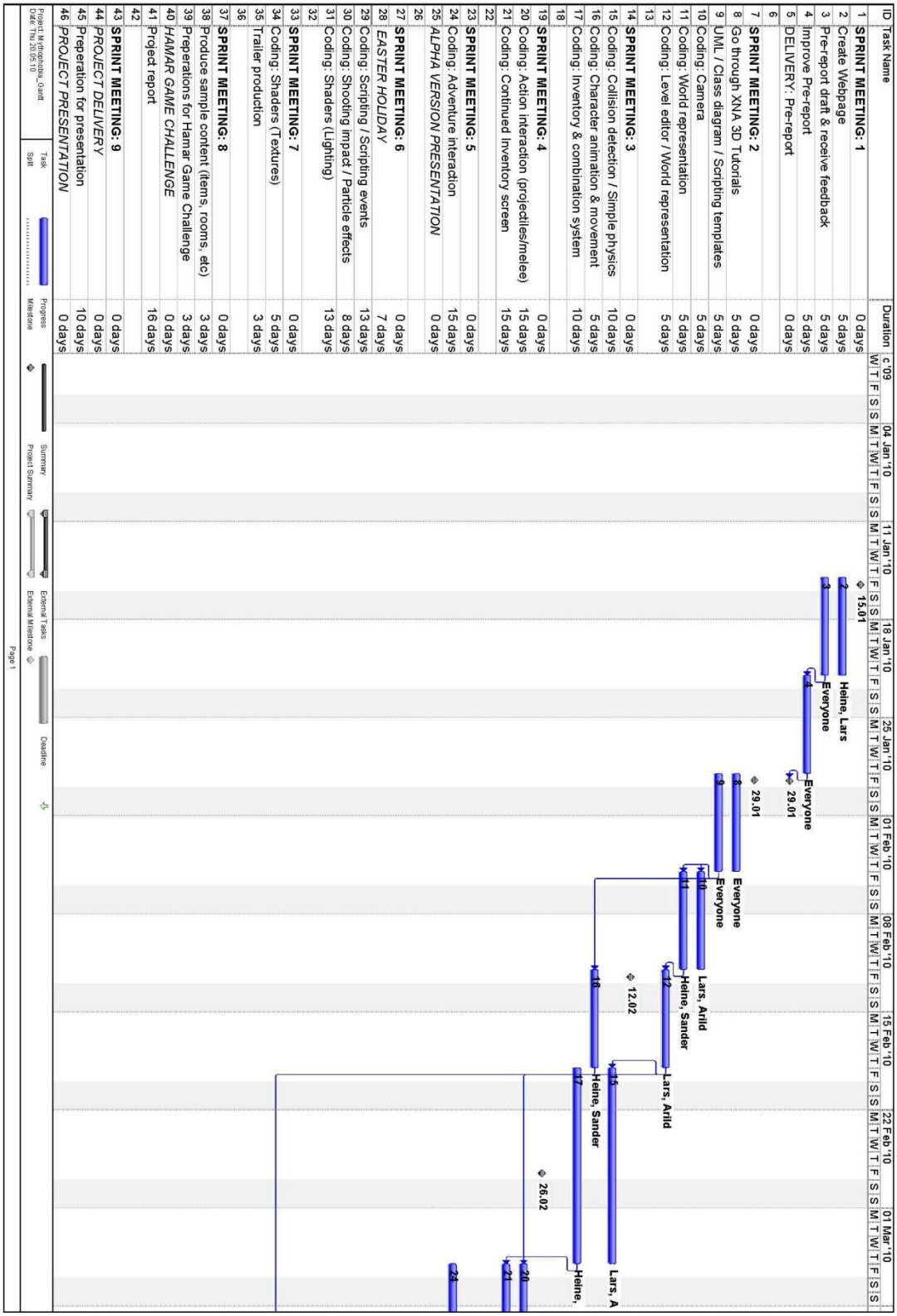


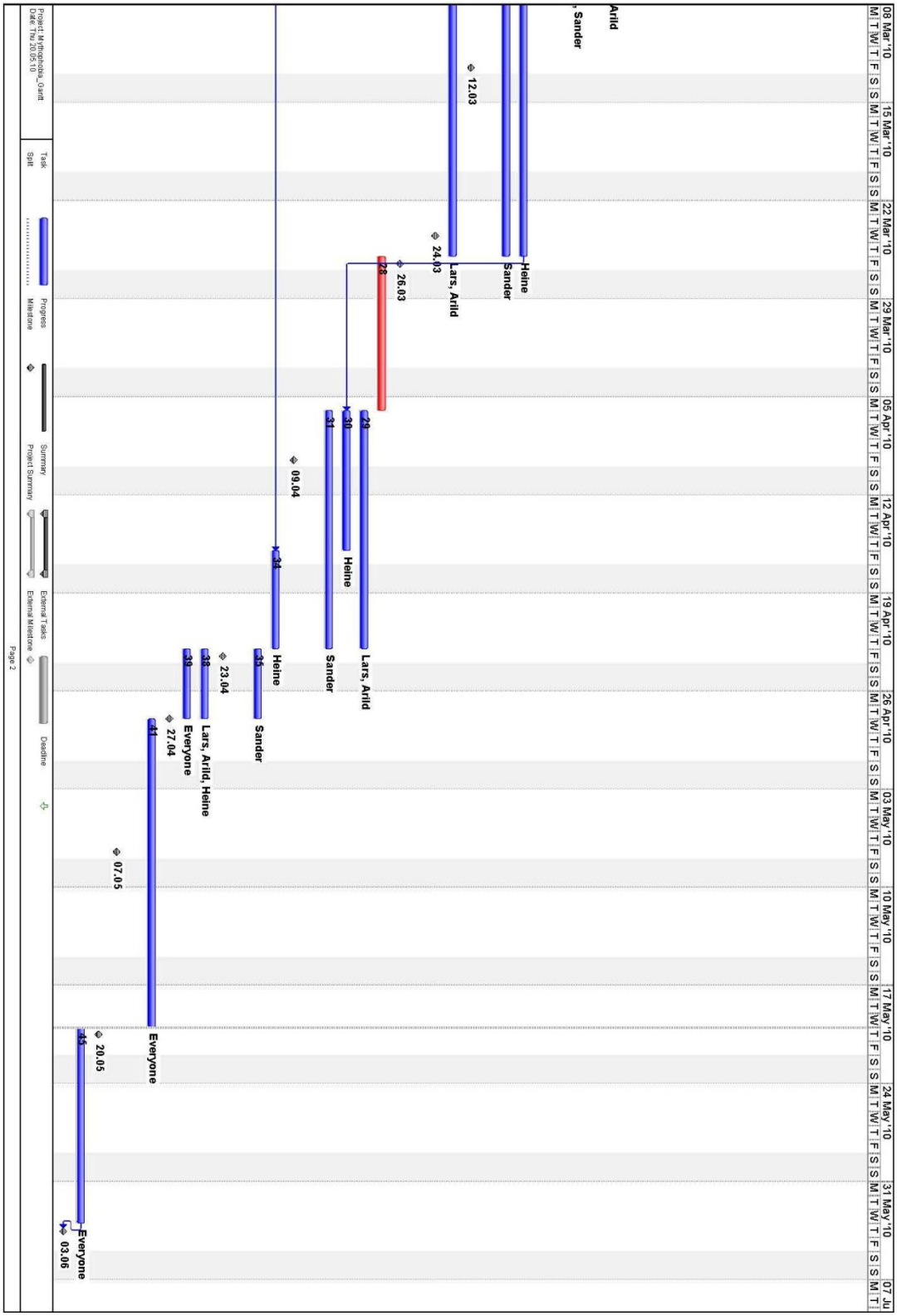

ID
Task Name
Duration
1 SPRINT MEETING: 1 0 days
2 Create Webpage 5 days
3 Pre-report draft & receive feedback 5 days
4 Improve Pre-report 5 days
5 DELIVERY: Pre-report 0 days
6
7 SPRINT MEETING: 2 0 days
8 Go through XNA 3D Tutorials 5 days
9 UML / Class diagram / Scripting templates 5 days
10 Coding: Camera 5 days
11 Coding: World representation 10 days
12
13 SPRINT MEETING: 3 0 days
14 Coding: Character animation & movement 10 days
15 Coding: Inventory & combination system 10 days
16
17 SPRINT MEETING: 4 0 days
18 Coding: Collition detection / Simple physics 10 days
19 Coding: Action interaction (projectiles/melee) 10 days
20
21 SPRINT MEETING: 5 0 days
22 Coding: Adventure interaction 5 days
23 ALPHA VERSION PRESENTATION 0 days
24 Coding: Infection system 5 days
25 Coding: Health system (damage & restoration) 5 days
26
27 SPRINT MEETING: 6 0 days
28 Coding: Stamina system 5 days
29 Coding: Enemies & AI 14 days
30 Produce sample content (items, rooms, etc) 5 days
31
32 SPRINT MEETING: 7 0 days
33 Coding: Scripting / Scripting events 15 days
34 HAMAR GAME CHALLENGE (aprox.) 0 days
35
36 SPRINT MEETING: 8 0 days
37 Compose & include audio 10 days
38 Coding: Menus 5 days
39
40 SPRINT MEETING: 9 0 days
41 Project finalization phase 9 days
42
43 Project report 33 days
44 PROJECT DELIVERY 0 days
45
46 Trailer production 5 days
47 Preperation for presentation 10 days
48 PROJECT PRESENTATION 0 days
c '09
04 Jan '10
11 Jan '10
18 Jan '10
25 Jan '10
01 Feb '10
08 Feb '10
15 Feb '10
22 Feb '10
01 Mar '10
15.01
Heine, Lars
Everyone
29.01
Everyone
Everyone
Lars, Arild
12.02
Heine, Sander
Lars, Arild
26.02
Heine, Sa
Project: MyProjects_Gantt
Date: Fri 26.01.10
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 1


Lars, Arild
Heine, Sander
12.03
Lars, Arild
24.03
Heine, Arild
Lars, Sander
26.03
Lars, Arild
Lars, Arild
09.04
23.04
Heine, Sander
Lars, Arild
23.04
Heine, Sander
Lars, Arild
07.05
Everyone
20.05
Everyone
Everyone
03.06

C Real Gantt chart




ID
Task Name
Duration
c '09
04 Jan '10
11 Jan '10
18 Jan '10
25 Jan '10
01 Feb '10
08 Feb '10
15 Feb '10
22 Feb '10
01 Mar '10
1 SPRINT MEETING: 1 0 days
2 Create Webpage 5 days
3 Pre-report draft & receive feedback 5 days
4 Improve Pre-report 5 days
5 DELIVERY: Pre-report 0 days
6
7 SPRINT MEETING: 2 0 days
8 Go through XNA 3D Tutorials 5 days
9 UML / Class diagram / Scripting templates 5 days
10 Coding: Camera 5 days
11 Coding: World representation 5 days
12 Coding: Level editor / World representation 5 days
13
14 SPRINT MEETING: 3 0 days
15 Coding: Collision detection / Simple physics 10 days
16 Coding: Character animation & movement 5 days
17 Coding: Inventory & combination system 10 days
18
19 SPRINT MEETING: 4 0 days
20 Coding: Action interaction (projectiles/melee) 15 days
21 Coding: Continued Inventory screen 15 days
22
23 SPRINT MEETING: 5 0 days
24 Coding: Adventure interaction 15 days
25 ALPHA VERSION PRESENTATION 0 days
26
27 SPRINT MEETING: 6 0 days
28 EASTER HOLIDAY 7 days
29 Coding: Scripting / Scripting events 13 days
30 Coding: Shooting impact / Particle effects 8 days
31 Coding: Shaders (Lighting) 13 days
32
33 SPRINT MEETING: 7 0 days
34 Coding: Shaders (Textures) 5 days
35 Trailer production 3 days
36
37 SPRINT MEETING: 8 0 days
38 Produce sample content (items, rooms, etc) 3 days
39 Preperations for Hamar Game Challenge 3 days
40 HAMAR GAME CHALLENGE 0 days
41 Project report 16 days
42
43 SPRINT MEETING: 9 0 days
44 PROJECT DELIVERY 0 days
45 Preperation for presentation 10 days
46 PROJECT PRESENTATION 0 days
15.01
Heine, Lars
Everyone
29.01
Everyone
29.01
Everyone
Everyone
Lars, Arild
Heine, Sander
12.02
Lars, Arild
Heine, Sander
Heine,
Lars, A
26.02
Project: MyHabitsMob_Gantt
Date: Thu 20.05.10
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 1


08 Mar '10
15 Mar '10
22 Mar '10
29 Mar '10
05 Apr '10
12 Apr '10
19 Apr '10
26 Apr '10
03 May '10
10 May '10
17 May '10
24 May '10
31 May '10
07 Ju
Arild
Sander
12.03
Heine
Sander
24.03
Lars, Arild
26.03
28
29
30
Heine
31
Sander
09.04
Lars, Arild
34
Heine
35
Sander
23.04
38
Lars, Arild, Heine
39
Everyone
27.04
41
Everyone
07.05
45
20.05
Everyone
03.06
Project: Mythgohoka_Gantt
Date: Thu 20.05.10
Task
Split
Milestone
Progress
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 2

D Status reports

The status reports are summaries and describe the total development of the most important tasks of the last 30 days.

D.1 Status report 1 - 12.02.2010

Camera

We began creating the class for handling the camera in the game. It needs to have three different modes, 3rd person mode, over-the-shoulder mode and first person mode.

The XNA tutorials was of great help when we started creating the camera, but we also gained more knowledge about how the camera works in other games, such as zooming etc., by studying the tutorials. We overestimated the time consumption this class would require and gave us more time working on the next task on the Gantt-chart.

Currently this class works and we have a camera which follows the player and is able to change between the different view-modes. In the future, the camera will collide with walls, because at this stage we do not have collision checking between the camera and the walls in the game world.

Game world

The game world is a more time consuming task than camera and is not scheduled to be finished at this point, but expected to be at 50% away from completion. Today it is on schedule and is still estimated to be finished in time. We have a controllable model moving around in a simplistic game world, this means we have a working model loader and are able to create levels where we can place models. So far we have not had any significant problems to deal with and the game world is developing as planned. We are the verge of creating a level editor, instead of creating the game world manually. The level editor will enable us to create a level in less time and in real time, so we do not have to for example place a wall, run the level, then see where it is in the world and if necessary move the wall again to the correct position.

D.2 Status report 2 – 12.03.2010

Inventory screen

Creating the inventory class is the highest priority of the month. The inventory class manages where the items the players pick are placed on his model. This task was quite the challenge when it came to positioning the items with the correct off-set and scale. We wanted the items to be shown in 3D in a circle around the model in the inventory screen. The player is able to rotate the items around the model and the item which is in front of the model is the one that is selected. The items are also dragged from its inventory slot, such as a handgun from a pocket over to the right hand. We underestimated the time we would need for this task because of unforeseen problems. The biggest problems were related to math and calculating matrixes. But today the inventory is working and we can drag items between slots, show items in the slots and rotate the items existing in the slots around the model. There is room for improvement if we gain some extra time, but until then we leave it as it is.

Camera collision

In any game there is camera and to this day a camera which is perfect does not exist, but a camera which follow the walls in the game world is something that our game need and the that we wanted to create. The camera collision checking we chose to implement was a ray-collision check. This was done by the least experienced programmer in our group, but was finished in the estimated time. XNA provides us with the Ray class and what it does is that it fires a ray, a vector, from the camera to the player. If that ray intercepts a wall, the camera will find the point of interception and move to that point. This forces the camera to always stay on the right side of the wall. The camera is now working properly and does not require more time and we can leave the camera as it is.

Level editor

As we mentioned we wanted to create a level editor and today we have a working level editor, which has allowed us to freely create levels as we wish, yet simple levels, but useful. In this level editor we can place surfaces and walls as we like, in addition it has the ability to scale these as small or big as we wish. We are certain this will save us a lot of time and was the right choice when it came to creating a game world. The level editor is not finished yet, but today it has the required functionality.

D.3 Status report 3 – 23.04.2010

Height maps

When creating a game world with multiple levels we had to figure out how to make the player aware of where he/ she is in the game world vertically. In order to do this we either had to create 3D collision checking, checking what the Y-value the player collides with, thereby place him above that value or we create height maps. The 3D collision checking is more advanced and time consuming than creating the height maps, though creating the height maps would not be easy either. We decided to create the height maps in the level editor on every surface. The surfaces then tell where the player is standing vertically and therefore we now have a player who can move up to a higher level by walking up ramps.

Shaders

We have started working on the shaders, but we learned that this was not as easy as we thought and are still not finished the shaders. For the shaders we used the XNAnimation library and tried to modify it. We encountered problems when doing this and are still working hard to find solutions and have this effect finished in time for Hamar Game Challenge.

Particle effects

The shooting is working and therefore we wanted to implement some effects connected to this action. We have implemented some particle effects to the bullet impacts and broken lights have sparks jumping out of them. The particle effects are complete and we like to have effects like this to show off at Hamar Game Challenge.

D.4 Status report 4 – 29.04.2010

Hamar Game Challenge

The group attended Hamar Game Challenge two days ago without claiming victory for us and Gjøvik University College. The competition was fierce and it was a close call when the winner was announced. We were disappointed, but we are satisfied with the product we have developed ourselves. Today we have working collision, camera, inventory, shooting, particle effects, interaction, level editor, shaders and loading of levels and models. The product is a working game engine at is unquestionably something to be proud of.

Now we have started working on the bachelor report and will spend the rest of our time on Gjøvik getting our work and everything we have learned down on paper and present the bachelor project to Gjøvik University College, and finally we deserve a spring break.

E Meeting reports

15. January 2010

Meeting Report
Friday 15. January(Full group)
- Started our first sprint.
- All read the examples of pre-project report.
- Assigned the following tasks to each one of us:
  Each write a dedicated part of the pre-project report.
- We agreed to have a draft of the pre-project report finnished by Friday 22. January.
- The group had a look on the website and it's coming along fine and will be finnished by the end of Monday 18. January.

Tasks for the weekend:
- Everyone have start working on their part of the pre-project report.
- Heine will continue working on the website.
- Lars is going to do some work the design document, making it ready for the presentation, and keep
filling in the Gantt-chart.

29. January 2010

Sprint Meeting Report
Friday 29. January(Full group)
- Website status: 80% finnished
- Pre-project report status: finnished
- Goals for the next sprint:
  * Finnish XNA tutorials
  * Planning the UML ,class-diagramsa and scripting.
  * Create a draft of the diagrams we need.
  * Begin coding the 2nd half the sprint.

12. February 2010

Sprint Meeting report
Friday 12. February(Full group)
- Camera-class status: 90%
- Gameworld status: 50% and on schedule
- Goals for the next sprint:
  * World representation is going to be finnished by next week.
  * Making animation for the game.

26. February 2010

Meeting report
Friday 26. February(Full group)
- Heine and Sander have worked on the inventory, they have managed to get the items to be positioned and rotated correctly on the player.
- Lars and Arild have finnished the heightmaps for the game world, so we now have a heightmap for the player to follow, which is calculated, exported and loaded.

Tasks for next week:
- Finnish the camera collision, which will be done by Arild and Lars.
- Heine and Sander will finnish the inventory.

12. March 2010

Meeting report
Friday 12. March(Full group)
- We had our test in Operating Systems.
- The group met with Simon for an update on the progress.
- We are on track with the project, but the next week will be intensive hard work to reach the finnishing line for inventory screen.
- We are now very eager to get some content in the game, such as models and textures, and Simon suggested we could take some pictures of some rooms etc. and try to build them in the game. Simon also said that we should write a mail to Nils and have him send some mails to his former students who has experience in 3D modeling.
- Sander sent a mail to Nils and we are now looking forward to having someone who can make some models and textures for the game.

Tasks for next week.
- Finnish the action- and adventure interaction, if are not finnished with this by then we have start cutting down on the things we want in the game.

25. March 2010

Meeting report
Thursday 25. March(Full group)
- Preparations for Hamar Game Challenge are being made.
- Sander and Heine worked on lights and shaders to make the game more visually appealing.
- Arild and Lars started researching for more info about the marked around games in our genre.

Tasks for tomorrow:
- The group will continue with the tasks above.

12. April 2010

Meeting report
Monday 12. April(Full group)
- Everyone back from Easter Holiday.
- Started picking up where we left off.
- Sander and Heine worked on the shaders and lights.
- Lars and Arild continued on the presentation, adding the market content they had found.

Tasks for tomorrow:
- Lars and Arild need to work out how to present the game in the best way, as a product which can produce income.
- Heine and Sander will try and make the shaders work with lights.

23. April 2010

Meeting report
Friday 23. April(Full group)
- Sander and Heine have been working on the shaders, but encountered some problems making them work.
- Lars and Arild have added more about the game's effects to the power point presentation.

Tasks for next week:
- Heine and Sander will try to finnish the shaders just in time for Hamar Game Challenge.
- Arild and Lars will polish the power point presentation and Lars will practice his speech.

07. May 2010

Meeting report

Friday 7. May(Full group)

- Last sprint meeting!
- We are well on our way to finnish the report.
- Since this is the last sprint meeting there will be no more meeting reports posted.

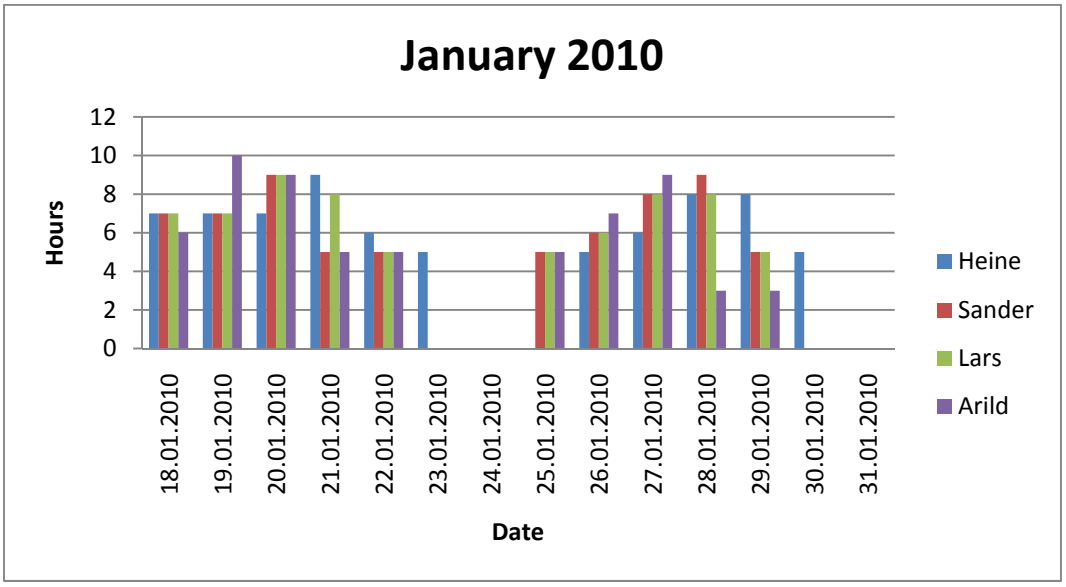


Fig. – January 2010

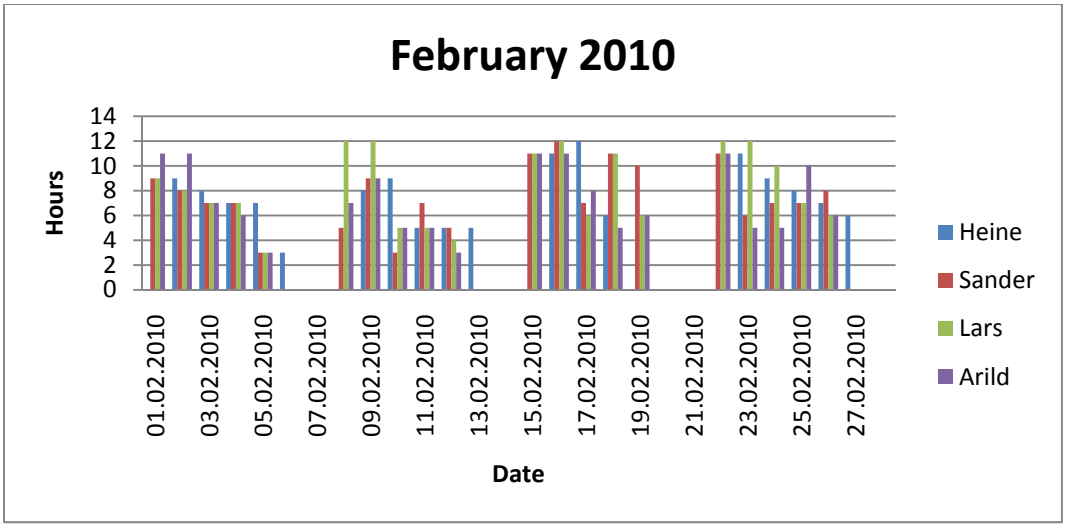


Fig. X.xx – January 2010

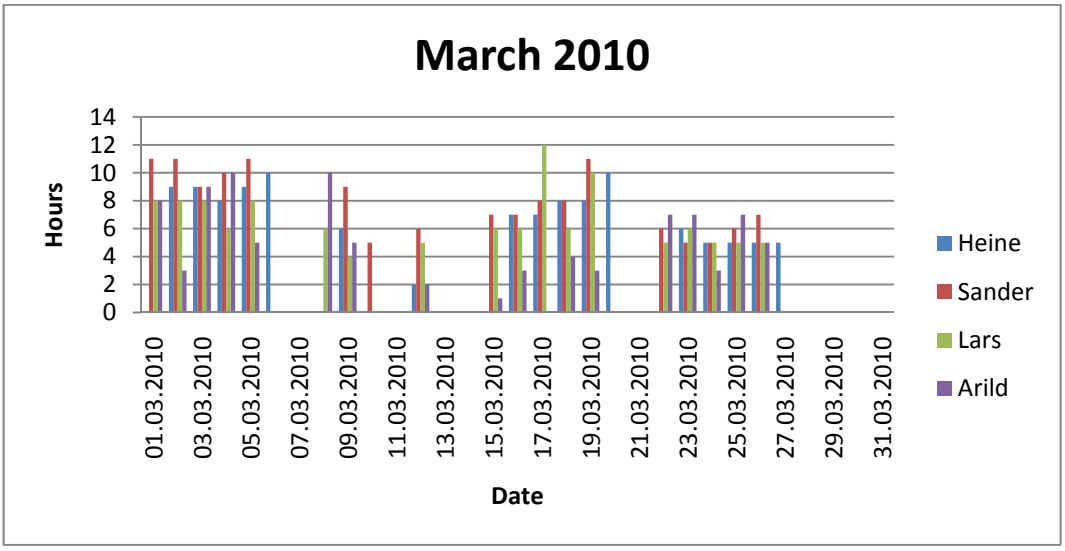


Fig. E.3 – January 2010

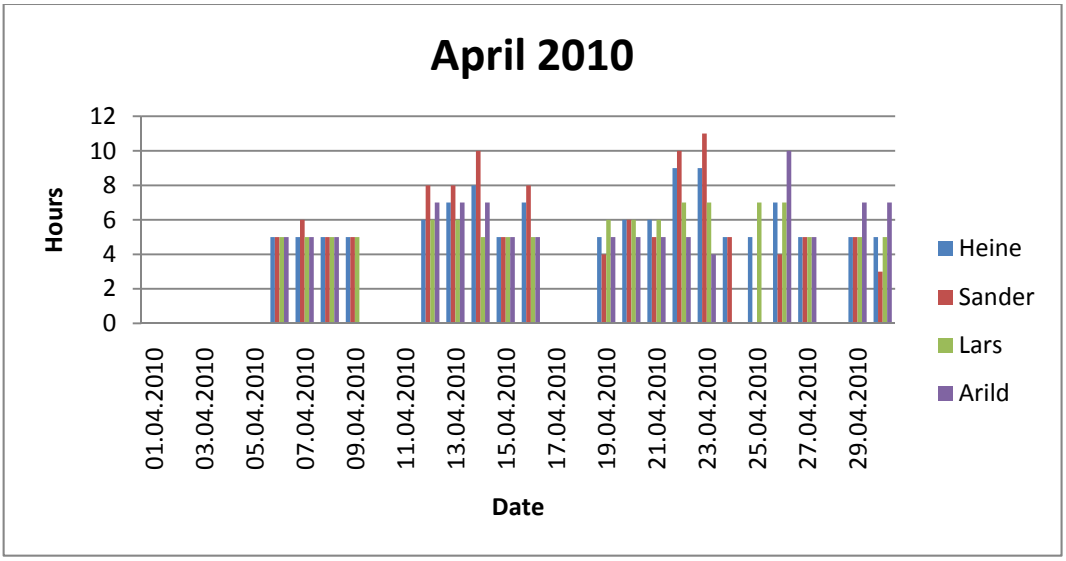


Fig. E.4 – January 2010

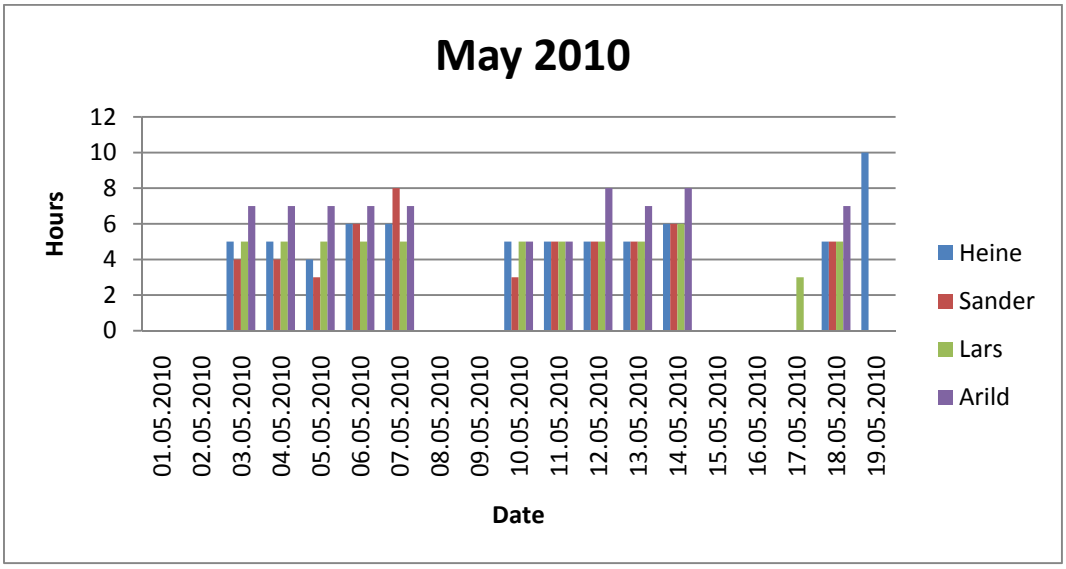


Fig. E.5 – January 2010

F Logs

This is a compressed version of the logs, the complete log can be found on the enclosed DVD.

Week: 3 Date: 2010-01-18 - 2010-01-24

Mon	**Hours**	**What was done**
Heine Martin	7	Website
Sander	7	Pre-project report.
Lars Inge	7	Gantt-chart.
Arild	6	Gantt-chart.
Tue	**Hours**	**What was done**
Heine Martin	7	Website.
Sander	7	Pre-project report.
Lars Inge	7	Pre-project report.
Arild	10	Pre-project report.
Wed	**Hours**	**What was done**
Heine Martin	9	Website.
Sander	9	Pre-project report.
Lars Inge	9	Pre-project report and website.
Arild	9	Pre-project report.
Thu	**Hours**	**What was done**
Heine Martin	6	Website.
Sander	5	Worked on the pre-project report
Lars Inge	8	Meeting with Simon and pre-project report.
Arild	5	Pre-project report.
Fri	**Hours**	**What was done**
Heine Martin	5	Website.
Sander	5	Pre-project report.
Lars Inge	5	Meeting with Simon and design document.
Arild	5	Design document.

Week: 4 Date: 2010-01-25 - 2010-01-30

Mon	**Hours**	**What was done**
Heine Martin	5	Website.
Sander	5	Pre-project report and design document.
Lars Inge	5	Power point presentation.
Arild	5	Pre-project report.
Tue	**Hours**	**What was done**
Heine Martin	6	Power point presentation.
Sander	6	Pre-project report and power point presentation.
Lars Inge	6	Power point presentation.
Arild	7	Power point-presentation.
Wed	**Hours**	**What was done**
Heine Martin	8	Presentation.
Sander	8	Presentation.
Lars Inge	8	Presentation.
Arild	9	Presentation.
Thu	**Hours**	**What was done**
Heine Martin	8	Worked on the pre-project report.
Sander	9	Pre-project report.
Lars Inge	8	Gantt-chart.
Arild	3	Pre-project report.
Fri	**Hours**	**What was done**
Heine Martin	5	Website and pre-project report.
Sander	5	Collision and pre-project report.
Lars Inge	5	Pre-project report.
Arild	3	Pre-project report

Week: 5 Date: 2010-01-31 - 2010-02-05

Mon	**Hours**	**What was done**
Heine Martin	9	Worked on the class diagram with the rest of the group.
Sander	9	Class-diagram.
Lars Inge	9	Class-diagram.
Arild	11	Class-diagram.
Tue	**Hours**	**What was done**
Heine Martin	8	Class-diagram.
Sander	8	Class-diagram.
Lars Inge	8	Class-diagram.
Arild	11	Class-diagram.
Wed	**Hours**	**What was done**
Heine Martin	7	Class-diagram.
Sander	7	Continuing with the development of the UML / class diagram.
Lars Inge	7	Class-diagram.
Arild	7	- Worked on the class-diagram with the group the entire day.
Thu	**Hours**	**What was done**
Heine Martin	7	Class-diagram.
Sander	7	AI research.
Lars Inge	7	Class-diagram.
Arild	6	Class-diagram.
Fri	**Hours**	**What was done**
Heine Martin	3	Worked on the class diagram with the rest of the group.
Sander	3	Simon and class-diagram.
Lars Inge	3	Class-diagram.
Arild	3	- I have worked on the class-diagram, like the rest of the group.

Week: 6 Date: 2010-02-06 - 2010-02-11

Mon	**Hours**	**What was done**
Heine Martin	8	Worked on the world representation, loading and drawing world objects.
Sander	5	Worked on the world representation, loading and drawing world objects.
Lars Inge	12	Code and class-diagram.
Arild	7	Tutorial.
Tue	**Hours**	**What was done**
Heine Martin	9	Worked on the world representation, loading and drawing world, npc and interactive objects.
Sander	9	World representation and SVN.
Lars Inge	12	Camera.
Arild	9	Camera.
Wed	**Hours**	**What was done**
Heine Martin	5	Worked on the world representation and did some modeling in 3ds max.
Sander	3	Modeling
Lars Inge	5	Camera.
Arild	5	- Worked on the camera together with Lars. The camera rotates properly now.
Thu	**Hours**	**What was done**
Heine Martin	5	Worked on the world representation and modeling.
Sander	7	Tutorials.
Lars Inge	5	Camera.
Arild	5	Worked on the camera and next step is to make the transition smoother.

Week: 7 Date: 2010-02-12 - 2010-02-17

Fri	**Hours**	**What was done**
Heine Martin	5	Spring meeting.
Sander	5	Sprint meeting.
Lars Inge	4	Sprint meeting.
Arild	3	Sprint meeting.
Mon	**Hours**	**What was done**
Heine Martin	11	Started working on the animation
Sander	11	Worked in 3dsmax with animation and modeling.
Lars Inge	11	World representation.
Arild	11	Formated laptop.
Tue	**Hours**	**What was done**
Heine Martin	12	Continued working on the animation.
Sander	12	XNA libraries.
Lars Inge	12	Level editor.
Arild	11	Level editor.
Wed	**Hours**	**What was done**
Heine Martin	6	Animation.
Sander	7	Animation.
Lars Inge	6	Level editor.
Arild	8	- Made a video of the level editor.

Week: 8 Date: 2010-02-18 - 2010-02-23

Thu	**Hours**	**What was done**
Heine Martin	0	After installing a new keyboard on my laptop, it suddenly became super slow.
Sander	11	Animation.
Lars Inge	11	Camera.
Arild	5	Camera.
Fri	**Hours**	**What was done**
Heine Martin	0	Got my computer working again.
Sander	10	Animation.
Lars Inge	6	Debug class.
Arild	6	Collision.
Mon	**Hours**	**What was done**
Heine Martin	11	Worked on the inventory screen.
Sander	11	Textures.
Lars Inge	12	Collision detection.
Arild	11	Collision detection.
Tue	**Hours**	**What was done**
Heine Martin	9	Worked with Sander on the inventoryscreen and started on making the loadingscreen.
Sander	6	Inventory-screen and loading-screen.
Lars Inge	12	Height maps.
Arild	5	Heightmaps.

Week: 9 Date: 2010-02-24 - 2010-03-01

Wed	**Hours**	**What was done**
Heine Martin	8	Website.
Sander	7	Working to improve the inventory screen
Lars Inge	10	Height maps.
Arild	5	Height maps.
Thu	**Hours**	**What was done**
Heine Martin	7	Worked on the loading-screen, loading the players inventory.
Sander	7	Inventory-screen.
Lars Inge	7	Worked on calculating the collision points for each surface.
Arild	10	- I worked with Lars on the height maps, working out the loading of it into the game world.
Fri	**Hours**	**What was done**
Heine Martin	6	Continued on the loading-screen and inventory-screen.
Sander	8	Working with the inventory screen.
Lars Inge	6	Height maps.
Arild	6	Camera collision.
Mon	**Hours**	**What was done**
Heine Martin	9	Inventory-screen.
Sander	11	In-game-screen.
Lars Inge	8	Level editor.
Arild	8	Camera collision.

Week: 10 Date: 2010-03-02 - 2010-03-07

Tue	**Hours**	**What was done**
Heine Martin	9	Implemented rayt-casting for checking collisions on the character slots in the inventory-screen.
Sander	11	Inventory-screen.
Lars Inge	8	Level editor.
Arild	3	- Looked for any examples or tutorials I could find on the internet, but not much luck.
Wed	**Hours**	**What was done**
Heine Martin	8	Sound.
Sander	9	Inventory-screen.
Lars Inge	8	Level editor.
Arild	9	Camera collision.
Thu	**Hours**	**What was done**
Heine Martin	9	Remade the loading of objects.
Sander	10	Enhanced the inventory-screen.
Lars Inge	6	Level editor.
Arild	10	Camera collision.
Fri	**Hours**	**What was done**
Heine Martin	10	Worked on the combination of items in the inventory-screen.
Sander	11	Inventory-screen.
Lars Inge	8	Height maps.
Arild	5	Collision detection.

Week: 11 Date: 2010-03-08 - 2010-03-13

Mon	**Hours**	**What was done**
Heine Martin	6	Worked on the combination system.
Sander	0	I am sick today and have been so the whole weekend.
Lars Inge	6	Had a presentation of our bachelor assignment.
Arild	10	Camera collision.
Tue	**Hours**	**What was done**
Heine Martin	0	PC trouble.
Sander	9	Camera collision.
Lars Inge	4	Camera.
Arild	5	- Finished the camera collision. Next step will be to make an more advanced camera collision.
Wed	**Hours**	**What was done**
Heine Martin	0	PC trouble.
Sander	5	Camera collision.
Lars Inge	0	Test.
Arild	0	Test.
Thu	**Hours**	**What was done**
Heine Martin	2	Waited for Dell to get back to me about the hard drive. Took a small peek on shaders.
Sander	0	Test.
Lars Inge	0	Test.
Arild	0	Test.
Fri	**Hours**	**What was done**
Heine Martin	0	Got an answer from Dell today, and they are sending me a new hard drive.
Sander	6	Sprint meeting.
Lars Inge	5	Sprint meeting.
Arild	2	Sprint meeting.

Week: 12 Date: 2010-03-14 - 2010-03-19

Mon	**Hours**	**What was done**
Heine Martin	7	Action interaction.
Sander	7	Wireframes.
Lars Inge	6	Reworked Object2D.
Arild	1	Sick.
Tue	**Hours**	**What was done**
Heine Martin	7	Collision.
Sander	7	Inventory-screen.
Lars Inge	6	GUI interface.
Arild	3	- I still felt sick, but met with the group anyway to try and help Lars as much as possible.
Wed	**Hours**	**What was done**
Heine Martin	8	Made a class for the hit zones for the model.
Sander	8	Cleaning up and improving the inventory screen, the code is unclean and hard to understand as it is now.
Lars Inge	12	Animation.
Arild	0	Stayed at home to recover.
Thu	**Hours**	**What was done**
Heine Martin	8	Collision.
Sander	8	Inventory class.
Lars Inge	6	Continued working on the adventure menu.
Arild	4	Feeling better and assisted Lars.
Fri	**Hours**	**What was done**
Heine Martin	10	Added collision checks with NPC-Spheres.
Sander	11	I have AGAIN re-arranged the inventory screen.
Lars Inge	10	Started taking a look at interactive object class.
Arild	3	Started working on the editor to add items in the world.

Week: 13 Date: 2010-03-20 - 2010-03-25

Mon	**Hours**	**What was done**
Heine Martin	6	Continued on the npc collisions.
Sander	6	Inventory screen
Lars Inge	5	Continued working on the adventure menu.
Arild	7	- Worked on the camera collision.
Tue	**Hours**	**What was done**
Heine Martin	5	Made a power point presentation.
Sander	5	Still inventory screen....
Lars Inge	6	Continued working on the adventure menu.
Arild	7	Prepared for the presentation with the rest of the group.
Wed	**Hours**	**What was done**
Heine Martin	5	Pitched the game again for the producer.
Sander	5	Presented the game for the producer.
Lars Inge	5	Had a project pitch for the producer, and got some useful feedback to be taken into consideration when pitching at Hamar Game Challenge.
Arild	3	Attended the presentation for our producer, Øyvind Nordstrand.
Thu	**Hours**	**What was done**
Heine Martin	5	Worked on the weaponsystem.
Sander	6	Shaders.
Lars Inge	5	Created a GG Studio logo movie.
Arild	7	Started the research in preperation for Hamar Game Challenge.

Week: 14 Date: 2010-03-26 - 2010-03-31

Fri	Hours	What was done
Heine Martin	5	We had a talk with Simon about lightning and shadows. Took a look at some lightning examples.
Sander	7	Simon "lectured" us about shadows and lights, so now we have a few ideas and are going to start looking at them.
Lars Inge	5	Had a talk with Simon about lightning and shadows. Added some more work on the adventure menu. Fixed some bugs and values to give a better look.
Arild	5	- Continued with the research of the market regarding our game.

Week: 15 Date: 2010-04-01 - 2010-04-06

Tue	Hours	What was done
Heine Martin	5	Back from vacation. Had to set myself into what I was working on before the vacation. Started on making collision checks for the NPC.
Sander	5	First day back to work after eastern holiday. Finding out where to start with the lighting, how to "attack" the situation.
Lars Inge	5	Back to work after easter holiday. Started looking at scripting. Researched our possibilities for creating scripted events on the internet. Found some possible solutions, but they will be time consuming to include at this point in the project. Will research some more tomorrow.
Arild	5	- Back in business after Easter holiday. Picked up where I left off with the market research.

Week: 16 Date: 2010-04-07 - 2010-04-12

Wed	**Hours**	**What was done**
Heine Martin	5	Continued on the weapon system.
Sander	6	Looking up the XNAnimation library scource code to find out how its shader is intergraded into the library.
Lars Inge	5	Continued reaserching the possibilites for scripting.
Arild	5	- Finding it hard to dig up statistics around the market. But have some numbers regarding copies sold of a few games.
Thu	**Hours**	**What was done**
Heine Martin	5	Wrote the function that is executed when the player fires a weapon.
Sander	5	Struggeling to understand the shader code in the XNAnimation library scource code
Lars Inge	5	Started creating scripted events.
Arild	5	Math work in the game.
Fri	**Hours**	**What was done**
Heine Martin	5	Put the collision tests together in the update function in the weaponsystem class. Also went over the collisions in the aim class.
Sander	5	Still working on how to change the baked in shader from the XNAnimation library
Lars Inge	5	Continued working on the scripted events.
Arild	0	Sick.
Mon	**Hours**	**What was done**
Heine Martin	6	Finished the aiming and shooting and bugs.
Sander	8	Looking up written shader tutorials and video tutorials about shaders
Lars Inge	6	Continued working on scripting.
Arild	7	- Started on the presentation for Hamar Game Challenge.

Week: 17 Date: 2010-04-13 - 2010-04-18

Tue	**Hours**	**What was done**
Heine Martin	7	Started looking at some examples of particle effects. Trying to find out how we want the effects.
Sander	8	Been looking at how to make the XNANIMATION library use attenuation.
Lars Inge	6	Continued working on the scripting..
Arild	7	- Added more content to the power point presentation we are making for Hamar Game Challenge.
Wed	**Hours**	**What was done**
Heine Martin	8	Used an particle-system example from Microsoft XNA Community Game Platform.
Sander	10	Been trying to work out the shader to work with light attenuation the whole day >_<
Lars Inge	5	Continued working on the scripting system.
Arild	7	- Needed more content for the presentation, did more research and added what I felt was useful information.
Thu	**Hours**	**What was done**
Heine Martin	5	Went over the settings for the particle effect, to make it fit for a bullet impact. Also made an image to use in the particle effect.
Sander	5	Made the lighting attenuation work with the skinned-models shader, the light shines brightest near the player while it gets dark as you get out of the range of the light gradually.
Lars Inge	5	Continued working on the scripting.
Arild	5	- Created the power point presentation and inserted the content I had found so far.
Fri	**Hours**	**What was done**
Heine Martin	7	Cleaned up some code in the loading-screen, weapon-system and aim class.
Sander	8	Trying to apply the XNAnimation shader to be working with the walls and floors
Lars Inge	5	Took a look at the inventory screen for adding the pick up object to the inventory at the end of the script. Had to set myself into a deal of Sander's code, and figure out how I am able to do this.
Arild	5	- Worked on the power point presentation.

Week: 18 Date: 2010-04-19 - 2010-04-24

Mon	**Hours**	**What was done**
Heine Martin	5	Looked at some models to put in the game.
Sander	4	Loadingscreen.
Lars Inge	6	Took up a bit of issues regarding the inventory screen.
Arild	5	- Worked on the power point presentation, added more about the group.
Tue	**Hours**	**What was done**
Heine Martin	6	Made a level for the presentation at Hamar Game Challenge.
Sander	6	Struggeling getting the XNAanimation shader to work
Lars Inge	6	Continued fixing on the inventory screen.
Arild	5	- Added more to the power point presentation
Wed	**Hours**	**What was done**
Heine Martin	6	Worked with Sander, trying to get some custom shaders for the walls and floor.
Sander	5	Me and heine discussed and exchanged shaders.
Lars Inge	6	Continued fixing on the inventory screen.
Arild	5	Worked on the power point presentaiton, added more about the innovasion.
Thu	**Hours**	**What was done**
Heine Martin	9	Continued trying to get the skinnedModelEffect shader to work on the walls, but came across alot of problems. Thought of other methods to get the textures wrapped on the wall models.
Sander	10	Started up alot of older projects who uses shaders to try and understand how they are used, but had no luck.
Lars Inge	7	Exported and fixed on a level that Heine created to be used in the Hamar Game Challenge.
Arild	5	- Worked on the power point presentation, designing the structure of the presentation.
Fri	**Hours**	**What was done**
Heine Martin	9	Started making a new class for walls, where I use a primitive for the wall instead of a model.
Sander	11	Finished the Riemers shader tutorial
Lars Inge	7	Populated the level I exported yesterday with objects and models.
Arild	4	- Added more content about the effects we have in our game.
Sat	**Hours**	**What was done**
Heine Martin	5	Continued on the wall class, and focused more on getting it to work.
Sander	5	Researching how good action adventure games have been rated over the last year and their sales numbers

Week: 19 Date: 2010-04-25 - 2010-04-30

Sun	**Hours**	**What was done**
Heine Martin	5	Fixed some small errors with the wrapping of textures in the wall class, and implemented wrapping on the floor as well. Started looking at some models to put in the game.
Lars Inge	7	Prepared myself for speaking at the Hamar Game Challenge and fixed up the powerpoint presentation.
Mon	**Hours**	**What was done**
Heine Martin	7	Polished the game for the showoff video we're making of the game. Added a couple of models to the world.
Sander	4	Made showof video for hamar game challange
Lars Inge	7	Prepared myself for speaking at the Hamar Game Challenge.
Arild	10	- Polished the presentation and creating a time table for Lars when he will present the project at Hamar Game Challenge.
Tue	**Hours**	**What was done**
Heine Martin	5	Presented our game at Hamar Game Challenge. We were happy about our presentation, but there were several good groups/games at the event, and unfortunately we didn't win. We hope we might be back next year with another game ;)
Sander	5	Attended Hamar Game Challenge.
Lars Inge	5	Attended and presented our game at Hamar Game Challenge.
Arild	5	- Attended the Hamar Game Challenge, it was a great experience and alot of fun.
Wed	**Hours**	**What was done**
Heine Martin	0	Everyone took a day of after Hamar Game Challenge, getting ready to start on the documentation part of the project.
Sander	0	One day off after Hamar Game Challange as a reward for our hard and long days of works the last few months.
Lars Inge	0	Took a day off after Hamar Game Challenge.
Thu	**Hours**	**What was done**
Heine Martin	5	Started on the documentation. Wrote a little about our proficiency and background, and what we need to learn.
Sander	5	Started on the documentation, about he testing group we were going to have.
Lars Inge	5	Started writing the final documentation on the Bachelor Report.
Arild	7	- Started on the bachelor report.
Fri	**Hours**	**What was done**
Heine Martin	5	Wrote about the project description
Sander	3	Started on the documentation
Lars Inge	5	Continued working on the Bachelor Report.
Arild	7	- Wrote about the project description in the bachelor report.

Week: 20 Date: 2010-05-01 - 2010-05-06

Mon	**Hours**	**What was done**
Heine Martin	5	Wrote some more on the project description part and tried finding a good solution for merging word documents.
Sander	4	We are just going to use word, since using MiKTeX requires you to invest very much time into getting to know how to use it, and we have to little time left to use time on MiKTeX instead of writing documentation.
Lars Inge	5	Continued working on the Bachelor Report. Wrote about implementations, some about level editor and collisions.
Arild	7	- Bachelor report, writing about discussions we have had during the development of the project.
Tue	**Hours**	**What was done**
Heine Martin	5	Wrote an introduction to the project report. Added a function on the website to list all logs for a user for printing.
Sander	4	Wrote about the testing group and quality assurance in the report, but some of it has been rewriten and lines has been added by Lars Inge.
Lars Inge	5	Continued working on the Bachelor Report. Wrote about quality assurance, the planned test group and debugging.
Arild	7	- Bachelor report, wrote about camera collision.
Wed	**Hours**	**What was done**
Heine Martin	4	Set up some of the layout for the report and wrote a summary.
Sander	3	Setup a document with all the headlines and parts that are meant to be in the project
Lars Inge	5	Continued working on the Bachelor Report. Wrote about hamar game challenge and fixed the whole document's formating.
Arild	7	- Bachelor report, wrote about the structure of the report.
Thu	**Hours**	**What was done**
Heine Martin	6	Started writing about shaders (wrapping of textures)
Sander	6	Wrote generally about shaders, and the introduction gudelines and demarcation.
Lars Inge	5	Wrote some on the Bachelor Report. Mainly used my time commenting code in the project and looked at compiling code documentation.
Arild	7	- Worked on the bachelor report, edited what I had written earlier and polished the report in general.

Week: 21 Date: 2010-05-07 - 2010-05-12

Fri	**Hours**	**What was done**
Heine Martin	6	Finished writing about wrapping of textures.
Sander	8	Wrote about the inventory screen and camera collision before delivering the document to simon for feedback
Lars Inge	5	Continued working on the Bachelor Report. Compiled a first draft with all the content we have up untill now and send it to Simon for feedback.
Arild	7	- Added more content to what I had written earlier.
Mon	**Hours**	**What was done**
Heine Martin	5	Working at home this day, catching up on writing logs and commenting code.
Sander	3	Fixing the loggs formating, and filled just a few logg holes Bouncing around in the documentation trying to read through stuff and fix sentences that doesnt sound right or make sence.
Lars Inge	5	Continued working on the final Bachelor report. Filled in some holes in the logs.
Arild	5	- Got feedback from Simon and started working on the flaws he could point out.
Tue	**Hours**	**What was done**
Heine Martin	5	Continued writing logs. Added some changes to the logging on the website. Hours worked are now logged individually for future statistics.
Sander	5	Working on general documentation. Target group, The bacchelor projects purpose, And About the reports sowftware tool used.
Lars Inge	5	Continued working on the Bachelor Report. Talked to Simon about his feedback on the first draft on the Report.
Arild	5	- Started writing the appraisal part of the report.
Wed	**Hours**	**What was done**
Heine Martin	5	Added and corrected some logs. Commented more code in the project, and installed doxygen for compiling the documentation. We used the same program for another project we did and XNA and have good experience with it.
Sander	5	Generally just working on the documentation, trying to fill inn holes and improve allready written text around in the document
Lars Inge	5	Continued working on the Bachelor Report. Started working on the final Gantt chart, with changes made during the project's lifespan. Looked at the members' logs, and created it out of them. Wrote about the game world representation and rewrote level editor and added description of the level editor exportation process.
Arild	8	- Removed flaws and did a spellcheck on the report.

Week: 22 Date: 2010-05-13 - 2010-05-18

Thu	**Hours**	**What was done**
Heine Martin	5	Continued commenting and cleaning up code.
Sander	5	Started writing about the XNAnimation library, how it works and how to use it.
Lars Inge	5	Continued working on the Bachelor Report. Integrated my documentation into the final document. Revised the report and fixed typos etc.
Arild	7	- Added some codeblocks in the report and commented some code.
Fri	**Hours**	**What was done**
Heine Martin	6	Worked on the documentation. Commented and cleaned up more code.
Sander	6	Finished writing about the XNAnimation library
Lars Inge	6	Continued working on the Bachelor Report. Fixed styling and general overview of the document. Changed headers to a professional standard. Added several illustrations and figures as well as a standard for description of the figures in the document. Added information in the introduction part and fixed some typos. Sendt the latest version of the report to Simon and Jayson for a second feedback.
Arild	8	- Checked the report for flaws and misspelling before delivering the report to Lars who sent it by email to Simon and Jason for feedback over the weekend.
Mon	**Hours**	**What was done**
Lars Inge	3	Continued working on the Bachelor Report. Fixed grammar and added content in the introduction.
Arild	0	- Norwegian national day, icecream and pizza.
Tue	**Hours**	**What was done**
Heine Martin	5	Fixed the front page and the norwegian and english summary according to the layout we're supposed to use. Commented more code and fixed some logs.
Sander	5	Filling in a few more logg holes, now the logg should be complete. Putting some work iv been doing home into the final bachelor report about the XNAnimation library. Rewriting some of the sentences in the XNAnimation section and some other places in the bachelor report
Lars Inge	5	Continued working on the Bachelor Report. Fixed page numbering of pages and sat up top-texts for the document. Added alot of figures and figure numbering.
Arild	7	- Wrote more about the report's structure, organzing etc.

Week: 23 Date: 2010-05-19 - 2010-05-24

Wed	**Hours**	**What was done**
Heine Martin	10	Worked on the documentation. Fixed some charts for the amount of hours we have worked.
Sander	12	Working on many parts of the documentation
Lars Inge	12	Continued working on the Bacherlor Report. Last day of writing on the report. Fixed up some typos, sentances and grammar based on the feedback from Simon. Continued on adding content in Adventure interaction and Scripting. Fixed page layouts to fit in the final printout for the document and restructured a bit to better match the template.
Arild	12	- Worked on the report all day. Added more content to the appendix.
Thu	**Hours**	**What was done**
Sander	0	Working on the documentation from 0000
Lars Inge	12	Continued working on the report from midnight. Contributed with content, merged content into the document and added appendixes. Overall worked on finalizing the document for hand in at noon.

G Design document

Mythofobia

Design Document
Version 0.1

Table of Contents

I. Overview

FOCUS:
Mythofobia is a 3rd person true action/adventure game, focused on both action-sequences and logical adventure-game puzzles. Set in a horror-themed setting, it tells a rich, fictional, but yet believeable non-linear story, with multiple solutions and storyline outcomes, dependant on the players actions and puzzle-solutions. The game features a realistic method to store inventory and to interact with objects and items in the game world.

Mythofobia is a 3rd person true action/adventure game, focused on both action-sequences and logical adventure-game puzzles. Set in a horror-theme, it tells a rich, fictional, but yet believeable non-linear story, with multiple solutions and storyline outcomes, dependant on the players actions and puzzle-solutions.

Mythofobia tells the story of three people waking up in test-chambers in a underground laboratory overrun with mythological creatures such as zombies, warewolfes and vampires, created by rogue scientists on the goverments payroll, as well as the story of a special operations police officer entering the laboratory complex along with his unit, to make an end to the illegal research taking place there.

The game is played through the eyes of these three characters, switching back and forth between them throughout the game, each with his/her own strength and weakness.

The game is a non-linear story driven game, where the players actions will have effect on both the story and outcome of the game as well as what the other playable characters will have to do and will be facing. Various scripted events can change the story of a character, dependant on what the player has done when playing a previous character.

The players main objective in the game is to attemt to escape the underground complex for the test subjects, while attemting to reach the core of the complex to destroy the entire laboratory for the special ops officer. All while fighting off hordes of fabricated mythological creatures with variuous melee weapons and firearms, and solving logical puzzles to reach their goal.

The setting of the game is in a unspecified english-speaking country, in the present. The whole game will be played out in a sealed off underground laboratory, though this is a large complex with everything from lab-areas, dining-areas, sleeping quarters and entertainment areas, etc.

The theme of the game is dark and frightening. The lab has been overrun with the creatures, so it has seen a lot of damage and is now not in top condition, making varous paths in the laboratories blocked and destruction to be seen everywhere.

The game starts with the first of the characters waking up in a test chamber sealed with glass and bars on the inside. The chamber suddently gets jumped at by a warewolf outside, though the glass makes it blured and unrecognizable. The warewolf hits through the glass and rips off the bars from the inside, but to the side so that some glass still remains. While this is going on a person appears in the door of the room and the warewolf gets more interested in him, jumping away from the chamber and towards the door, with the man fleeing for his life. The character then has to get out

of the chamber, only to find himself trapped in a large underground complex with the only goal to survive and see sunlight again.

Furthermore he finds and saves another test-subject, which is the second playable character, but they get seperated and will have different paths to survive.

The government has realized that something is wrong in the complex and sends a team of special ops to deal with it. Not realizing that it was as bad as this, almost the whole team got killed off, though a small few survives. The player plays one of these survivors which gets seperated with the rest during the attack where almost the whole team dies. Without any communications to the outside, he decides to execute the emergency operation, which is planting a bomb in the core of the laboratory to bring an end to anything left in the laboratory.

II. Game mechanics

Overview

The player will be playing one character at a time, playing the story from 3 characters, and thereby 3 different points of view. Which character the player will be playing will change at spesific markers in the story. The playtime will overlap each other, making it so that you can see the character you first played or is going to play later, from another one's point of view.

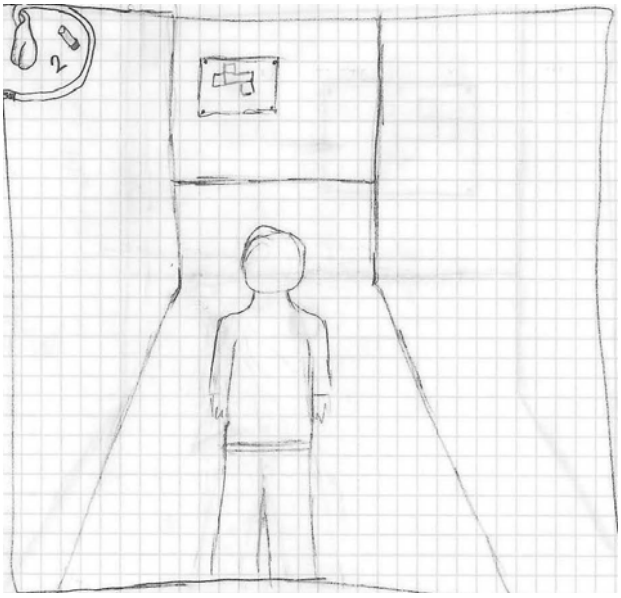
The actions you perform while playing one character will effect characters you play later, such as taking or leaving an item in a area, making it available or not for another character at a later time. Killing or running from an enemy will decide if the next character visiting the area is going to have to deal with it or not.

The game is played mainly through a 3rd person perspective, using a 1st person adventure perspective for world interaction, and handling inventory items as well as examination and combination of these in a "pause" menu. (Everything is descriped in details under later topics in this section.)

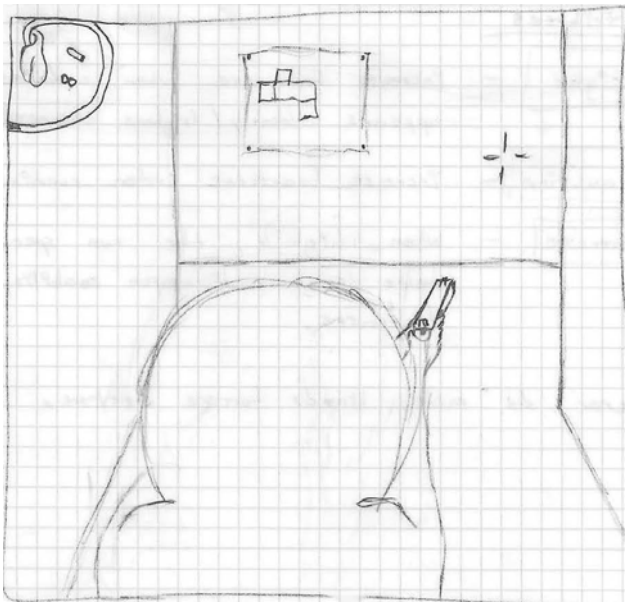
The player will be able to walk, run, sprint, hit with melee weapons and shoot with guns, as well as execute a dodge move, where the character dodges danger. Jumping will not be possible except for scripted events.

Camera

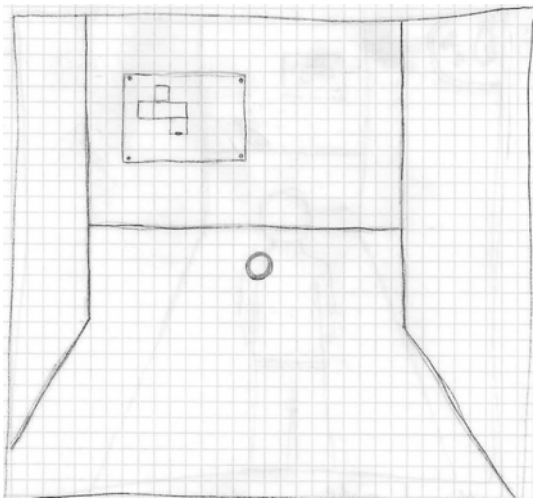
The game will feature 3 different modes of camera, depending on what action the player is going to perform.



The camera will be generally be placed in a distanced, semi top-down 3rd person view behind the character, and will follow the character as he/she moves. This will be the default camera position when exploring and moving in the game world. The camera will also be controllable up/down and left/right around the front of the character (limited to not go behind him/her or directly down/up).



When the player wants to perform an action, such as attack with a weapon, the camera will zoom in and change to a over-the-shoulder 3rd person view, and give a centered view in front of the player.



When the player wants to interact with the world, the camera will zoom even further and go into a 1st person view from the character's point of view. In this view the center of the screen will function as a cursor to choose objects to interact with.

Action Mode

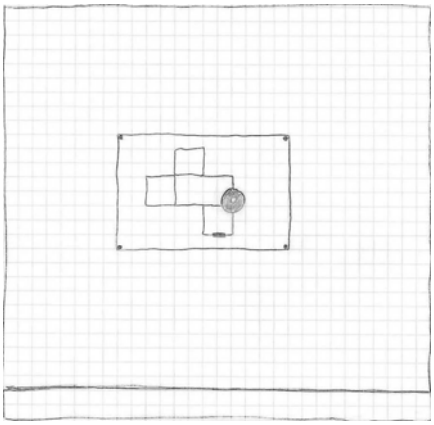
The player will enter the action mode (over-the-shoulder 3rd person camera view) when he is to use attacks with various weapons.

A crossair or laser aim will appear for firearms or other ranged weapons, but not for melee weapons. Melee weapons will still be moveable through aiming, but must be aimed without a crossair.

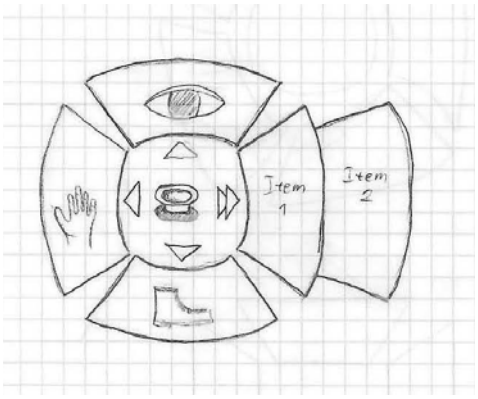
The player will still be able to move the character, as well as move the aim for the weapon in this mode. (See the Controls section for how)

Adventure Mode

The player will enter the adventure mode (1st person camera view) when he is to interact and use items or objects in or with the world.



The middle of the screen will function as a crossair, which will notify when it is over and in range of something to interact with.



When the player then chooses to interact with an item, an adventure menu appears, in which the player can choose to interact by *Hand*, *Foot*, *Eye* or one of the *Items* currently in hands. The interaction method is chosen by moving the analog stick in the direction of the method, and confirming it with a choose button.

The second item is chosen by flipping the stick to the right twice. If only one item is held in hand, or a two-handed item is used (like a shotgun), only one item will be shown here.

Example use:

- Hand: Pick up, Use, Open, Push, etc...
- Foot: Kick, Jump over, Step on, etc…
- Eye: Look at / Examine.
- Item: Combine / Use with world item.

The player will still, in the 1st person view be able to move the character and the 1st person view camera as well. (See the Controls section for how)

Player Movement and Player Actions

Movement:
The player will be able to walk around in the world through controls described under Control Summary. The player will be able to move forward, backwards and to the sides in all of the 3 different camera/view modes though the same controls. Though in 1st person and Action mode, walking to the sides will be replaced with strafing and turning by aiming to the sides or moving the 1st person camera around will be needed to turn.

Running:
The player can choose to hold the run button while moving to make the character move faster than when walking. This will make the character exhausted when used over longer periods.

Sprint:
When the player wants to get away from something or move somewhere really fast, he can hold the sprint button while moving. The sprint will be the fastest available movement, but will make the character exhausted really fast. (See Exhaustion section for further details)

Dodge:
By tapping the run or sprint button twice while holding a movement direction, the character will perform a dodge move, which will be a small jump or toss of body in the movement direction held. This can be used when the player gets too close or when an enemy is trying to hit or jump at the character, to dodge the enemy's attack.

Quick-turn:
By holding move backwards and clicking the run button, the character will perform a "quick-turn". This is a 180° turn to face the character in the opposite direction of what he/she was currently facing.

Adventure-action:
When the player chooses an adventure-action in the adventure menu, and the action will feature the character doing something, the camera will go back in 3rd person mode, showing an animation of the character performing this action. This can be everything from opening a door, picking up an item, pushing a button or moving an object.

Quick-action:
When the player does not have time to open the adventure menu, but is in need for a quick interaction with a world item, such as opening a door or climbing a ladder, he/she can click the quick-action button while close to the object in the 3rd person mode. This action will choose the most common action for that specific object, such as open for a door, push for a button, look at for a picture, pick up for an item on the ground and jump over for a gap.

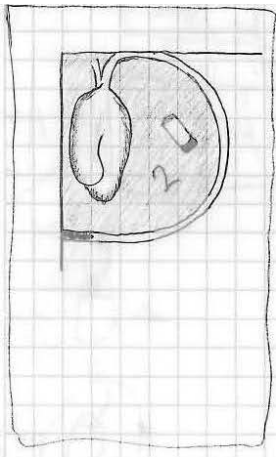
Aim and fire/hit:
The player can aim by pressing and holding the aim button, and then move the aim with the camera control, while moving the character with the movement control. In this "aim-mode", or "action-mode", the player can fire or hit with his weapon by clicking the fire/hit button. Some attacks will feature exhaustion if exhausting to use, such as melee weapon or a highly powerful firearm.

Reload:
When the player runs out of bullets in his weapon, he can press the reload button once to reload his

weapon with bullets, shells or other projectiles in a controlled, but average speeded motion. Reloading can be speeded up by constantly pressing the reload button fast (smashing it), which will “speed-reload”, but will feature a high percentage chance of dropping bullets or magazine to the ground.

Hotkeys:
Items can be placed on up to 4 hotkeys, which can be used for the player to change between items or weapon currently held in hand. Which item or weapon is going to be represented for which hotkey is customizable in an inventory menu. (Described in details under Inventory section)

In-Game GUI



The only in-game GUI that will be featured is a bloodvein-incapsled heart in the top left corner of the screen.

Inside this half-circled GUI-object there will be an image of the current ammo-type of the weapon the player is currently using, and a number indicator showing the number of bullets left in the weapon.

Also, the heart is the game’s health indicator, and will gradually get a blackish color when taking damage, and when fully consumed with this color, the player will die. The heart will be featuring a small beating animation, which will beat faster when the player gets exhausted.

The vein encapsling the heart and ammo-indicator is a infection-status indicator, which when infected will gradually fill itself with a “infection” color towards the heart. (More on the infection status under the Infection topic.)

Health

When the player get hit by enemies attacks, such as bit, hit or shot, he will suffer an amount of damage based on enemy and attack/weapon type.
(See Game Elements – Enemies section for detailed damage calculation)

Though there are ways for the player to replendish health, with various types of healing items available throughout the play world, which heals completely, partially or temporarily, depending on the item.
(See Game Elements – Items section for detailed healing items)

Infection

When the player gets bit by an enemy, there is a chance than he gets infected with that type of infection of the creature that bit him. Its important to difference this from every attack, as the infections will only happen though biting. The more he get bit, the faster the infection will spread.

There are 3 different infections; zombie, warewolf and vampire, each with a distinct difference in color featured in the vein. When the player gets bit while not infected, he will be infected by the type he is bit by. Though while infected by one type of infection, he is immune to both other infections.

There are healing items and antidote items available for the user to temporarly halt or completely stop the infection.
(See Game Elements – Items)

Exhaustion

When the character runs, sprints or uses a weapon by exhausting means, he/she will gradually get exhausted. The exhaustion is kept track of and measured in 0-100%. This is shown to the player through the heart rate of the animated heart in the in-game GUI. The faster it beats, the more exhausted the character is.

Running will constantly gradually add exhaustion to the character.
Sprinting will also gradually add exhaustion, but at a much higher rate.
Weapon use will add a static amount of exhaustion at the point of fire/attack only.

The different characters will also have different stamina amounts. For example a small and physically fit character will be more enduring than a muscular and tall character. Exhaustion will also be calculated out of the amount of items the character is carrying, making him/her more exhausted when carrying more.

The rates will have to be balanced in the end result, but the basic consequences will be as follows:

- Run speed gets lower by increased exhaustion.
- Sprint speed gets lower by increased exhaustion.
- Aiming gets more unsteady by increased exhaustion.
- Melee damage gets lower by increased exhaustion.
- 100% exhaustion will feature:
    - Only able to walk.
    - Almost no damage on melee.
    - Extremely unsteady aim.

When walking, and not attacking, the exhaustion will gradually decrease. When standing still, it will decrease even faster.

Ammunition and reloading

The ammunition indicator will show how many bullets/shells/projectiles remaining in the weapon the character is currently holding. Total amount of ammunition will only be visible from within the inventory menu. (see the next topic)

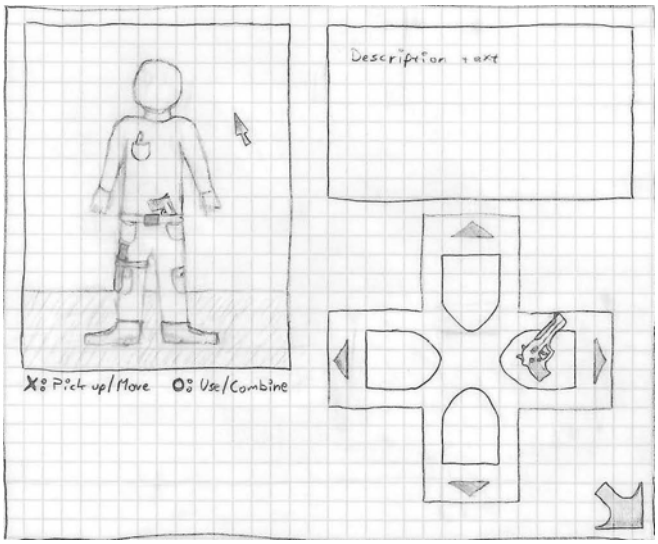
Reloading will be a bit slow, though not as slow as real-life, if the player is to reload with loose bullets or other projectiles.

This can be speeded up in clip-loaded weapons by having pre-loaded clips. Clips with ammunition will be prioritized before starting reloading manually.
(See Game Elements – Items section)

The reloading animation can also be speeded up the player constantly pressing the reload button fast (smashing), which will “speed-reload”, but will feature a high percentage chance of dropping bullets or magazines to the ground. Bullets will be lost, but magazines will be available to pick up again.

If the player drops his last magazine, he will start reloading with loose bullets if available. If not, he will have to pick back up the magazine (can be done by quick-action), and then start reloading again.

Inventory



The inventory screen will feature a 3D model of the character in the top left corner, with the ability to rotate and zoom on it. This model has a drag and drop ability (by the use of a cursor) of items the character is currently carrying from any pocket or other itemslot to another. This will give a limitation of carry capacity to the amount of pockets or containers the character posesses.

The model will feature a bit of animation, such as blinking, shrugging from time to time and a bit of occational swinging, to give a non-static inventory screen.

Pockets and other containers will get semi-transparent when hovered over with the cursor, so that the item can be seen more clearly.

Item slots:

- Pockets (1 for each pocket)
- Belt (1 front, 1 back)
- Shoulders (1 around each shoulder)
- Hands (1 in each hand)
- Neck (1)
- Head (1)
- Extra/addon slots such as; Knifestrap, Fannypack, etc…

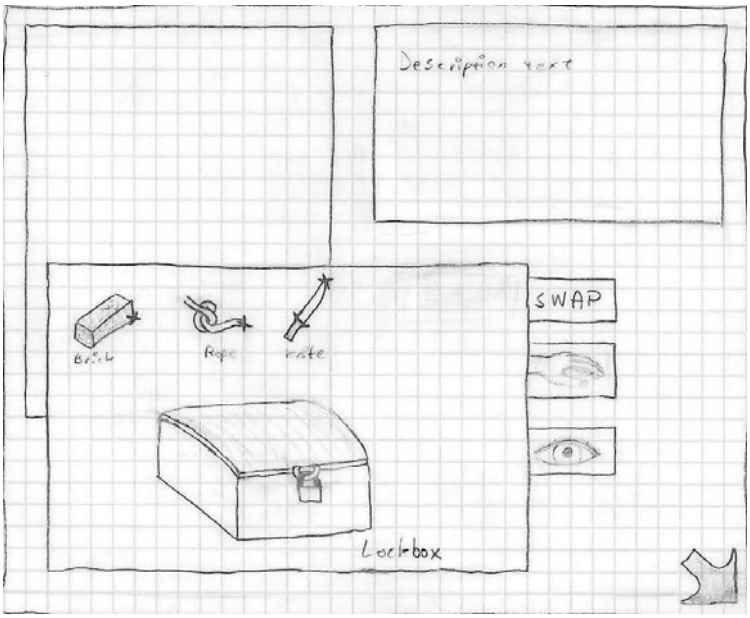
To see what items that will be available to each slot, see the Game Elements – Items section of this document.

Some items will also not be able to store in any chosen item slot. For example a rope or a shotgun may not be put in a pocket, but must be hanged around a shoulder or held in hand. As well as a gun may not the put in a chest pocket or around the neck, but in a pocket of the right size, the belt or in hand.

The availability of an item will depend on the item slot you decide to put it. For example it will take more time to pull out an item from your leg-pocket than from your front pocket or belt.

Up to 4 items can be hotkeyed to be placed in your hands in a menu on the inventory screen, as shown in the figure. This hotkey feature will function the same way as if the item would be dragged into the characters hand, but is for changing items/weapons in hand in-game.

Examining and combining items



By dragging one item over another in the drag and drop 3D model in the inventory menu, and pressing the combine button, a new window will appear as shown in the figure.

In this window, one of the combination items will be the subject of the screen, with the ability to rotate and zoom.

The subject will have different parts on it, reacting differently to operations. For example as in the picture, examining the padlock will not be the same as examining the lid of the box. Using the brick on the box will not provide any usefulness, but using it on the padlock will break the box open. Thereafter you can use the lid to open it.

The player can here choose between 3 operations on the subject item, as follows:

- Combination item (The other item)
- Use/Operate
- Look at/Examine

The subject of the screen can also be swapped with the combination item.

By using an item without dragging another item over it, the same window till appear, but without the combination item, making only use/operate and look at/examine the operations available on the subject item.

Using items with objects in the world

As explained under Adventure Menu topic, two of the adventure-actions available through the adventue menu will be either of the items the character is currently holding in his/her hands.

When this interaction is chosen, the player will combine the item he is holding with the item in the world. This means that to use a item, such as a key on a door, the key has to be put in the characters hands first, either through hotkey interaction or drag and drop in the inventory meny, and thereafter used in the adventure-menu.

Maps

The game will have maps available to obtain throughout the game in the form of blueprints, fire-escape plans, and such. These will be stored just as normal items and weapons are, and are examined to view.

In addition, if the player obtains a pen, combining the map and the pen will feature free drawing on the map. This can be used to mark locked doors, make notes or mark items of significance by the player him/herself.

Also, blank paper found could be used to freely design a map by the player himself, or just simply make notes in-game.

Flashlights

The game will feature various flashlight items (see Game Elements – Items section for details), that provides different amounts and ways of lighting up the world around the character.

As the game is set in a horror theme, the general environment lighting will be dark, and the need for flashlights or other light sources will be present.

The various kinds of flashlights consist of the following:

- Normal flashlights (yellow cone shaped light)
- Powerful flashlights (normal, but lights up further)
- Lantern light (360° light)
- Led light versions (blue, dimmer light)
- Sungun (uv-light for use as weapon against vampires)

Animations

When the character pulls out an item from a given item slot, there will be a different animation for each item slot, making it appear as he is reaching inside the right pocket and pulling the item out.

This animation can be as small as moving the hand to the chest pocket and pulling out the item, and as big as pulling a shotgun off the shoulder and into his/her hands.

The main point is a graphical reality of how the item/weapon goes from the itemslot and into the characters hands.

Control Summary

Playstation3:

Common controls:

D-pad – Hotkeys
Left thumstick – Movement
Cross – Run
Square – Sprint
Cross + Back on left thumbstick – Quick-turn (180°)
Square + Square + Move direction – Dodge
Triangle – Inventory
Right thumbstick – Camera rotation

L1 (hold) – Action mode
L2 (tap) – Adventure mode

Start – Pause menu

Action mode controls:

R1 – Fire / Hit
Right thumbstick – Aim
Sircle – Reload

Adventure mode controls:

R1 or R3 – Open adventure menu / Choose adventure option
R2 or Square – Open adventure menu / Cancel adventure menu
Right thumbstick – 1st person camera / Choose option adventure menu

Inventory controls:

Left thumbstick – Character model rotation
L1 , R1 – Model zoom in / out
Right thumbstick – Cursor
Cross – Pick up / Move item
Sircle – Use / Combine item

PC (Mouse + Keyboard):

Common controls:
1 2 3 4 – Hotkeys
W A S D – Movement
SHIFT – Run
CTRL – Sprint
S + SHIFT – Quick-turn (180°)
E – Quick-action
Q – Dodge
F – Inventory
Mouse XY – Camera rotation

Mousebutton 2 – Action mode
TAB – Adventure mode

ESC – Pause menu

Action mode controls:
Mousebutton 1 – Fire / Hit
Mouse XY – Aim
R – Reload

Adventure mode controls:
Mousebutton 1 – Open adventure menu / Choose adventure option
Mousebutton 2 – Open adventure menu / Cancel adventure menu
Mouse XY – 1st person camera / Choose option adventure menu

Inventory controls:
W A S D – Character model rotation
Mousewheel – Model zoom in / out
Mouse XY – Cursor
Mousebutton 1 – Pick up / Move item
Mousebutton 2 – Use / Combine item

Saving

The game will feature saving at specific locations only. When the player uses specific save-computers in the game, he/she will be able to save the game at these workstations.

There will be unlimited amounts of saves available, though.

Storytelling

The story of the game will mainly be played out through cut-scenes when reaching a spesific point in the world, or executing a certain action. These cut-scenes will be pauseable and skippable for the users own choice.

It is an important focus not to make these cutscenes longer than absolutely neccesarry, but still captivating in the terms of interesting or excitement, so that the end-user does not get bored while watching them.

Parts of the story will also be explorable throughout readable books, notes or other documents found in the game. These will only be text that the player will have to read if he/she wants a more in-depth explanation of parts of the story. This kind of storytelling will be addition, and will only be optionable reading. The player will not have to go around reading everything he/she comes over, unless they really is interested in depth history.

Difficulty levels

The player will be able to choose between the following difficulty levels at the start of a new game:

Entertaining

(Easy difficulty)

- Enemies: Normal amount, 50% hitpoints.
- Weapons: Normal
- Ammunition: More available
- Healing items: More available
- Maps: Marked automatically with everything discovered.

Challenging

(Normal difficulty)

- Enemies: Normal amount, 100% hitpoints.
- Weapons: Normal
- Ammunition: Normal
- Healing items: Normal
- Maps: Marked automatically with key elements.

Realistic

(Hard difficulty)

- Enemies: Lots, 150% hitpoints.
- Weapons: Rare
- Ammunition: Rare
- Healing items: Rare
- Maps: Not marked automatically at all.

III. Bibliography

Resources contributing to understanding the game mechanics, story or description.

Similar games or games with borrowed elements/features

Resident Evil 4

The controls and gameplay is much similar. Camera is similar in control and placement, but should be placed further away from the character than in this game.

Cold Fear

The camera is very alike in shooting mode. Camera is similar in placement, but should be placed a bit closer. The camera should be a middlething between RE4 and CF.

Condemned 2

The adventuremenu design is borrowed from the interactionmenu in this game. It is similar, but is redesigned to get another look.

Mythological reading

How to fight off a zombie invation

Dracula

Warewolf

Movies

Dracula

Silver bullet

Night of the living dead / Day of the dead / Dawn of the dead (Original movies)

H DVD content

The following is included on the DVD marked "All content" which is included in the report (Folder names in bold):

Code
Here all the project code we have developed is stored, both the for game itself and the level editor.

Code Documentation
Here the documentation generated with Doxygen for the project code is stored.

Documentation
Here this written report is stored, with all appendixes.

Presentations
Here some of the presentations we used for pitching the game for producer and at Hamar Game Challenge is stored.

Release
Here a compiled version of the project is stored, which can be run without having to compile the code.

Required files for running the release and instructions for this are located directly on the root of the DVD.