Blekinge tekniska högskola
Sektionen för teknokultur, humaniora och samhällsbyggnad
Digitala Spel
VT08
Kandidatarbete/slutreflektion

Reflection of the graduate exam project

Biosteel

Rickard Remelin
rire05@student.bth.se
remme@home.se
Examination date: 2008 04 29

Tutor:
Joel Hakalax

Foreword

A very big thank you goes to the very skilled computer programmer, 2D/3D artist and game developer Roy Martinsson for creating the fantastic music I used in the game. Computer game music is an area I've worked with before but simply did not have time to go through while making this game, so why not accept a fantastic tune from a master when given.

Another big thank you goes to Henrik Möllergren who is a big source of inspiration for being able to do everything yourself when it comes to creating computer games. A man whose ideas in designing computer games are the most original and thorough I've ever seen. A man who himself have done 100's of proof of concepts programming in many different languages and engines.

I also want to thank Stefan Noreman for giving me some of the bases for the sounds I used.

And the last thank you goes to:
A man who shines like the brightest star of hope and creation I have ever seen.
A man who sees possibilities in both ideas and people and like magic brings it all to life.
A man who always sees you how small or insignificant you might seem to the world and who always see to it that you feel alright and have everything you need.
The founder of my school and education.
Mr Peter Ekdahl.
You made it all possible.

Abstract

To create a computer game does not need to be a particularly complex process. I chose to make it a complex process. My reason was that I wanted to know that I could handle and understand most of the process of making a typical computer game such as creating the concept, design, gameplay, 3D modeling, texturing, programming, sound editing and all those small details that makes a typical computer game.

To really be able to understand and be able to do this, I chose to work alone. I was prepared. I knew that I could do this. I had made a foolproof plan by not setting a static goal. My goal was to go as far as possible and not to stop after one small goal was met. This was a very big task but I had prepared about a year ahead so that I knew that I could do it. My goal was not to make a perfect, bug free, shining retail game, but a proof of concept. A working game of a concept which would use most elements of a typical computer game in a smaller scale. And I did it.

Keywords

Game Design, Gameplay, Story, Programming, 3D modelling, Texturing

Abstrakt

Att skapa ett datorspel behöver inte vara en komplicerad process. Jag valde att göra det till en komplicerad process. Anledningen var att jag ville veta att jag kan hantera och förstå mestadels av processen för att skapa ett typiskt datorspel så som att skapa konceptet, design, gameplay, 3D modellering, texturering, programmering, ljudeditering och alla de små detaljer som skapar ett typiskt datorspel.

För att förstå och klara av att göra detta, valde jag att arbeta ensam. Jag var förberedd. Jag visste att jag kunde klara av det. Jag hade en idiotsäker plan genom att inte fastställa ett statiskt mål. Mitt mål var att gå så långt som möjligt och inte stoppa efter ett litet mål var mött. Detta var en väldigt stor uppgift och jag hade förberett mig ungefär ett år i förväg så jag visste att jag kunde klara av det. Mitt mål var inte att skapa ett perfekt, buggfritt, skinande, kommersiellt spel, utan ett ”koncept bevis”. Ett fungerande konceptspel vilket använder sig av alla de delar som ingår i ett typiskt datorspel fast i mindre skala. Och jag gjorde det.

Nyckelord

Speldesign, Gameplay, Story, Programmering, 3D modellering, Texturering

Table of contents

Chapter 1
6 - Work description

Chapter 2
8 - Project plan

Chapter 3
9 - Weekly reports

Chapter 4
11 - Reflection
11 - The Vision
13 - Exploration
13 - Character evolution
14 - Interactive Dialogs
14 - Minigames
15 - Psychological and philosophical heavy story
15 - Voice acting
15 - Writing the story
16 - Design document
16 - Proof of Concept
17 - Dynamic text
17 - Interface
17 - Combining scenes
17 - The main gameplay
18 - Targeting the player
18 - Weapons
18 - Ordinary shot
19 - Shields
19 - Missiles
19 - Mines
20 - Energy
20 - Balancing the gameplay
20 - Sound
21 - Background
21 - Explosions
22 - Animations
22 - Intro sequence
22 - The limits of the engine
23 - Bugs
23 - Crashing when running the executable
24 - Crashing when the playership explodes
24 - Collision and physics
25 - Workarounds
25 - Letting go
26 - A complete and working “Proof of Concept”

Chapter 5

28 - Final Words – English version
29 - Slutord - Svensk version (Final Words - Swedish version)
30 - References & Links to external information or applications

Apendix

30 - Wordlist
31 - Projektplan

Chapter 1 - Work description

This was my last year at the “Digital Games” education program at Blekinge Institute of Technology (BTH) in Karlshamn. What I am going to tell you is essentially about my final project but I also feel that I need to tell you how I got to the point where I was able to do it.

The third and last year of my education was open for anyone to focus their work in the direction of their interest. I was able to set up a whole year of study for myself with 14 weeks towards the end focusing on this project. So I prepared the whole year to be able to develop 3D content and creating a game on my own. I did this both to be able to work alone once the university studies were finished and to get a wide and extensive feel for the work in most areas during the creation of a typical computer game.

In the end of the second year at the university, I was in a group of 18 people, completing a small sidescrolling(*1) shooter(*2) making the 3D content(*3). The game was all about running around shooting Zombies. This was the first experience for me creating 3D models and I totally loved it. I not only made models but also unwrapped(*4) and textured(*5) everything I did. This project went on for 10 weeks. I had worked in groups together with other people before but after that experience I felt that I had enough knowledge of how to structure work in groups and how a gaming company deploy work. So now when I understood the process I wanted to become the process. I wanted to do what we did in that group by myself and get a more in depth feel.

But to do that needs motivation, planning, and knowledge. So I had to prepare. I could use the first part of the third year to study whatever I wanted. As I in the last project had started to learn 3D modeling I wanted to continue to model but I could not use 3D Studio Max at this point as I wanted to be able to work at home. So I had to find a freeware alternative. I had heard about blender before so I took a more in depth look at it and decided to use it.

I figured for quite some time about different game engines such as Ogre 3D or Crystalspace but decided to learn Blenders game engine due to the time constraint as I had to do everything myself and their engine were really fast to work with even though it did not have the power of the others. So I had 4 blocks of 5 weeks each to learn something.

The first block I focused on learning everything that Blender could do and get an overview of every part of it's functionality.

The second block I focused on modeling lots of low polygon models in all different shapes and forms which can be hard to handle and become very efficient in this area. I also went for learning bone animation.

The third block I went for learning the basics of Gameblender.

The forth block was mainly focused on creating different kinds of gameplay with the engine so that I would know what I was able to do.

All information I gathered from the web searching Google and finding sites with the content needed. Finding information about Blender was no problem at all as it is well documented. Finding specific information for Gameblender was not always possible as no one had done some of the things I wanted to do.

But doing this I not only trained for being able to get knowledge and skill, I also tested myself if I was able to have full motivation while working for such a long time by myself. Many people said that it is very hard to do and they wanted me to work in a group, but I knew that I could do it as I always want to have full control over my psyche and my work. So I got 2 different kinds of stop watches. 1 that counted down and another that counted up. I set the downcounter for 8 hours each day, and the other I measured how many hours I actually worked. Because, I did not want to stop just because I had worked 8 hours. That was just a minimum of what I had to do before I could do anything else. And the perfect thing with having stop watches, is that nothing can distract you, and

trick you into believing that you have worked more than you actually have. Because if someone called or I went to check something on a homepage or whatever that did not have anything to do with the work, then I stopped the watches so that I would get the exact time I had worked. And I succeeded greatly. No week went beyond 40 hours of real work and most weeks actually went towards 80 hours or more. After these 4 blocks of trials I felt ready to be able to do what I wanted to do.

Soon the day came to start with the project. The school wanted me to set specific goal and not to low. Why they want that is probably most due to not let the student get away with being lazy because the student feels it has to start working right away to get anywhere. But I am not lazy. I have a mind of motivation, structure and effectiveness. I do not want a static goal and here are some reasons.

- We did not get any time to structure our project. The more static a plan is, the more you need to structure it and we had no time. The further you go towards a goal the easier it is to know where you are going to end up.
- I did not want to set a goal which I did not know if I could manage. Failing looks very bad and lowers self esteem.
- I did not want to set a goal that was to easy as I wanted to be able to continue working as long as possible, to learn as much as possible and do as much as possible.
- You get more knowledge by making as many different things as possible instead of doing more of the same thing and I wanted to try to make as much different things in a game as possible. The more things you set out to do, the harder it becomes to have a static goal.
- Setting a to strict path restricts possibilities. If you are open for anything, then you can do wonders. It's easier to do a fun game, if you are able to change what you notice is bad or could be better.

Not doing as the school recommends may be a dangerous move and can be bothersome for the school, but I trusted myself in this decision and so did my creative and talented tutor Joel Hakalax. As it went very well it seems like the choice was a perfect one.

Chapter 2 - Project plan

The graduate project consists of 20 weeks which are separated into two parts. The first part is the actual project with the length of 15 weeks and then a 5 weeks reflection part.

The first 15 weeks starts with a project plan in which the student is describing what should be done and when. To be allowed to start with the project, the student needs an approval of the project plan from the appointed tutor.

Each week weekly reflections are turned over and when it feels legitimate one has appointment with the tutor to discuss the work and sees to it that the project plan is followed. If the project plan is changed during the project, it ought to be rewritten with an argument for how and why. The work can be continued after a new approval. Part examinations should be also be done with the tutor after agreement.

The student shall share success, setbacks and tell how the problems got solved. The student shall tell how the cooperation has worked in the group and also the cooperation with external sources.

These requirements shall be upheld to get an acceptance of the graduation examination.

- Delivered and approved project plan before start of the course
- Delivered weekly reflections
- Continually tutor discussions
- Part examinations, productions and reflection examination

Chapter 3 – Weekly reports

Week 1:
Started to write on story and design document.

Week 2:
Created text functionality, dialogsystem, groundwork for the interface, main menu, changed the program code structure that that it became easier to use for a larger project.

Week 3:
Removed a bug that made the .exe file crash.
Removed a bug that made it impossible to remove the ship.
Developed the interface from showing only numbers to objects.

Week 4:
Implemented the enemy HP bar to the interface.
Created a more natural sight for the enemy ship.
Developed the collision system. Figured out how to set the settings for a face and copy the settings to all the marked faces.

Week 5:
Created a heatseaking missile.
Created a missile that blows when you push a button.
Fixed a bug that made the tracking system crash the game if an object dissapeared.
Developed the system for how the energy to the shields and shots works.

Week 6:
Tried to fix primary problems like:
- The ships goes to far from each other.
- The ships cant collide with each other.

Created mines.
Started creating lasershots.

Week 7:
Continued to develop the laser.
Put an energy cost for some of the weapons.
Created sounds for the weapons.
Created a temporary model for the missiles.
Started to put weapons onto the enemyship.

Week 8:
Balanced the gameplay for the enemyship.
Created a temporary solution so that the ship wont fly to far away.
Made it possible to shoot down the enemy missile and get energy.
Created sound for when the missiles fly. Created a system for looping the sound as the sound does not start with a loop.

Week 9:
Installed the shotpower icons for the enemy ship.
Made the enemy ship use energy.
Installed the energy counter for the enemy ship.
Created a mesh for bomb01.
Created a fire animation for missile 01.

Week 10:
Made an intro sequence and end sequence for the fight.
Created an animation for the enemy ship.

Week 11:
Tried to create an animation that make the ship rotate when it's turning. Failed.
Changed the introduction for the game and removed the scene which I made for this.
Tried to fix the distance problem between the ships.
Tried to fix the problem with the background texture not being repeate able.
Studied Gimp.

Week 12:
Created a background picture.
Textured the interface.

Week 13:
Textured bomb01.
Textured an explosion.
Succeeded to create transparent backgrounds for the textures.
Problem with the textures changes depending on what other textures are shown.

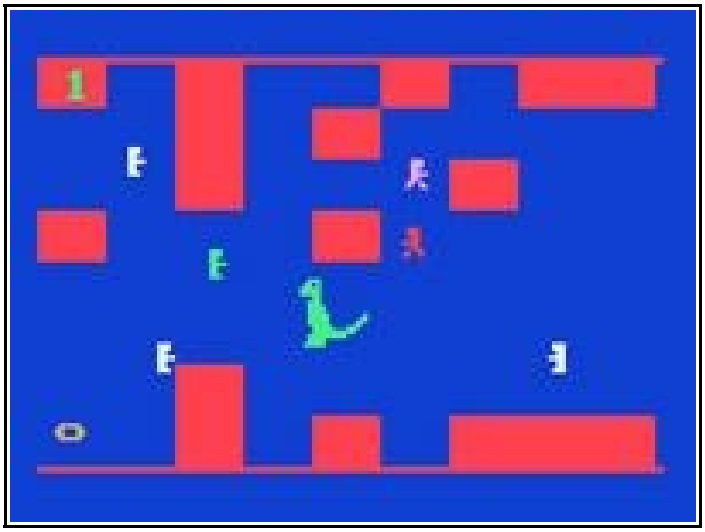
Chapter 4 - Reflection

The Vision

I had a vision. It was to build a groundwork or springboard for my future creations. And that was what I did.

I love games. Games are my passion and have always been. I was quite young when the first computer games started to come out on the market. Generation 2 as they were called, was the first wave of game consoles that could actually play something else than other versions of "Pong". These games were something that fascinated my mind incredibly. It felt like magic to me that you could steer something on a television screen. Every single game I felt like exploring and understanding. At that time, the games had incredibly bad graphics. You could play an old human magician that wandered through a mountain range full of fire and brimstone, looking for a magic ring, casting magic bolts on horrific creatures and with a gigantic dragon guarding the precious ring.

With imagination you can go very far, but not far enough. There was something else behind the games that made them fun to play. As I have always been a fan of every game genre that have ever been, I found patterns in each and everyone of them of what made them fun to play. Explaining these patterns is like explaining a painting. I could give a lot of information on how it looks like, but the picture would never be the same for me as for the person I am explaining it for. It also would take a whole book to explain and would never be as good as playing the games oneself. The important thing here is that I learned the core of what make games fun to play without the need of heavy graphics or advanced programming.



Quest for the Rings on Philips G7000 (aka Magnavox Odyssey)

So when I set out to do my graduate exam, I wanted to make the groundwork for a big and fascinating game that I was still able to do myself. That did not exclude the possibility to work with partners or even doing a high budget production game. Actually I always want to keep all possibilities open. But if I can make a game that is very large and complex and at the same time possible to do alone, then it is very easy to transform that project into a high end title as the groundwork is already there. All you need to do is making it look nicer. The production cost will most likely be lower than usual as there is nothing to hard needed to be in the game for it to be finished or fun and thereby the risk of a halting production is also gone. Doing the revers, that is to create a one man version of a high end budget game can be extremely hard if not impossible in some cases as the only aspect of fun might be to experience a beautiful world and interact with it which will not be possible to create with only 1 person.

A reason for why old games had great gameplay even though the graphics was horrific, was because the game designers were the programmers and could play around with the code and playtest their ideas long before they set any idea in stone. It feels like todays game designers are lacking the possibility to play around with the code themselves and therefor cant know if it their idea is fun or not. If you create a story and a setting then you get stuck in there trying to create a gameplay into that environment. This constrains the possibilities of how the twitching(6*), tactics,

strategy, and problem-solving may develop which actually is the core gameplay. If you create the core gameplay first and that is fun, you actually have a great game even without graphics, story and setting. It also is quite easy to implement all those elements to the gameplay and can even give you unique ideas of how the graphics, story and setting could look like.

I started with what I like within a core gameplay system.

- Lots of different ways to attack
- Every attack needs to have a unique quality
- Possibility to block any incoming attack.
- A way to counter every move from your opponent.
- Strategic moves by combining different attacks and counter attacks.

So what form can this system take? You have two positions. You and your enemy. In a realistic fighting game without weapons, it is very hard to see the difference between blows because you have the same arms and legs used as weapons for the different attacks.

Tekken 3 from Namco on Playstation 2

The only way you can recognize the different attacks is because of their movement. This is probably the reason why you usually can block only high or low.

Something that gets overlooked in lots of games which makes it very hard for any combat system is the range of the attack. If the range is to short for example in melee(*7) combat, it gets very hard to see what move is being done. Not only should you be able to see an attack coming, you need to recognize each and every attack that is being done and for that we need to be able to see every

attack very clearly.
So I visualized attacking and blocking between two positions and started figuring what it could look like. It would be a nice idea to do a fighting game but with sorcerers casting spells at each other. Very simple to evolve the gameplay and create an interesting story around that theme. But two human looking characters, running around shooting at each other, would mean advanced models and animations and a nice looking complex environment. No point really as it would not extend the gameplay. I also had to think of making it as easy as possible to program at the same time.
So what can fit best into the picture here if you and your enemy are objects without the need for animation and without a complex environment? Only two objects in empty space shooting at each other. And there it was. two spaceships dueling each other in space. Extremely simple to do and yet something different than ordinary space shooters which usually is to blow millions of things up by button mashing.

As of now, a game could be done by the pure idea I already had, but I also had the possibility to expand it much further and give it greater value and longer lifespan. So what more do I like in games after creating the core?

- Exploration
- Character evolution
- Interactive Dialogs
- Minigames
- Psychological and philosophical heavy story
- Voice acting

Exploration
The exploration can be easily implemented here as we have spaceships and thereby a whole universe to explore. Exploring planets can be as easy as moving a dot to another dot which is a planet and when getting near, showing text of what was found on that planet. This can be much more interesting than it sounds. In many RPG's(8*) you kill monster after monster and loot them to see if you found something. What usually is interesting here is not if you find a weapon but what stats the weapon have on it. Same thing could be implemented here. When you find a planet you might want to see what kind of minerals you can find. Also always give the player something for his effort of actually exploring the planets. Such as always give the player some minerals but if he is lucky or have followed some clues, give him more expensive or rare minerals. The minerals are of course only numbers and there is no need for any graphics at all. A very easy way to make the exploration feel big and complicated when it actually is not.

Character evolution
Character evolution does not need to actually be about a character. In our case it can be the ship itself. Evolving the ship can also be done in different ways. By upgrading it's weapons, functionality and appearance. You can also give it a story that explains it's background and thereby give the player a glimpse of what may happen later on. You can even give it a personality by giving it artificial intelligence(*9). All this can be done without much effort of either graphics or programming. It is often the change that intrigues more than the look itself and what the player have been told about what is happening.

Interactive Dialogs

Interactive Dialogs can really build the immersion for the player. As long as the player has a choice between some simple lines, the player feels like he's changing the world. Always give the player a feeling of great change by giving feedback about what he just said. Then you can go back to the linear story without needing to write a new branch in the storyline for every choice made. Giving the choice of changing the story in a big way does not need to be that hard either. You can simply go the same way, do the same missions, talk to the same characters, but when you do, some of them may say something different. So you only need to add an extra line of text for a few key elements throughout the story.

Minigames

Space Quest 3 by Sierra

Minigames are something that often feels misunderstood for some reason. They are usually added as a game in the game but not as an element of the actual game which I feel is a bit sad. Putting a small game in a larger one was a big hit when it had never been done before. I remember playing Space Quest III from Sierra which was a classic adventure game. When the character walked into a restaurant, there was an arcade game which you could put money into and play a minigame called "Astro Chicken". To be able to do this felt really mindbending at the time and gave great immersion for the game that anything could happen. But nowadays when the games are so very advanced, finding a separate game in another game is not a big deal anymore. It is always fun to find more gameplay but if it is separated from the actual game, it does not add to the overall feeling of achievement the player has of mastering the actual game.

One of the first games I played which integrated their minigames into the game was Final Fantasy VII where in one case you needed to ride the motorcycle to get from one place to another and the ordinary game was a heavily story driven RPG. If you give the player some minigames integrated into the real game, the player can still get the illusion that the game can withhold many more secrets and seem much bigger than it actually is just because all of the sudden the game showed a new dimension of itself.

The thing I feel minigames should be focused on is giving a gameplay element to parts of the game which exists but have none. World of Warcraft have the perfect example of a need for a minigame without implementing it. Crafting (that is creating weapons, armor, portions, etc.) in World of Warcraft entails getting all the ingredients and then push the "Create" button to make the item of your choice. The only problem is getting the ingredients. Most often that entails buying them or be given them. If you have crafted a lot and have reached the very end of the line exclusive crafting possibilities, you might want to do something really hard. The only thing that is hard with this is that you cant buy or be given the item and maybe have to kill a monster to get it. So crafting actually have no gameplay other than select the item you want to create and push the "Create" button. Making crafting a minigame in World of Warcraft would really make you feel that you actually are making an item yourself. For example making a sword you could have the player need to go trough a real process of actually creating a sword by heating up the metal and try to form it by hitting it with a hammer. You would feel that you are the one creating. You would not need to do this more advanced than necessary. Right now when you push the "Create" button you see a

progress bar that actually is just a timer for how long you have to wait until your item is created and nothing more. You could put in another bar with a moving pin inside it and have the player push when the pin is close to the middle to make the progress bar go up. All of the sudden you made the player do something instead of waiting and at the same time you gave more playtime to the game. Such simple methods as this is possible without either great graphics or advanced programming and yet you can give great gameplay to the player.

Psychological and philosophical heavy story

I myself love a psychologically and philosophically heavy story. A story can be really heavy even if the graphics is not. A story does not even need any form of graphics or gameplay. But there are things to consider when making a story for a game.

1. Don not push the story onto the player. Make the player choose what to read and what to be a part of.
2. Make it consistent with the gameworld and/or vice versa.

Voice acting

Voice acting can really make a difference if it is good. Not all gamers like to read and it can make someone listen through a needed but boring part of the story and even make it enjoyable to do so. It can also give a deeper feeling for what the character actually means by the sound of the voice. Fun voice acting can seem hard to put within a philosophically and psychologically heavy story but can also build a lot of atmosphere and act as a true comic relief. To tell a joke does not make everything else a joke as long as you separate the joke from the rest of the story. This is true even in horror movies where they sometimes actually use a joke to ease up the tension to be able to scare the viewer once again later on.

Writing the story

So everything I wanted to do seemed doable for my game. I started writing a story in the form of a book. This had many reasons.

1. It's easy to show the story for an outside reader.
2. I could easily continue making it an actual book.

I started with writing down all special kinds of moments that I wanted to see happening without thinking if it would fit with something else in the story or not. I wrote all things that fascinated my mind and gave me strong emotions. I continued with this even when I did other things. Because if you do something like a 3D model for a ship, then the mind starts to fantasize what that ship might have gone through. If I designed a weapon for the ship, then I started thinking of who might have built it, what they used it for and why. So most good parts of the story just came through to me without any kind of effort or outside inspiration. I just opened my mind for writing down the ideas that would have been there anyway. Then when I had a good base for the story with lots of interesting events I started connecting the dots. I did not even focus on this. It just came naturally. So when I had the main story of the middle and end finished I started to write the beginning.

Even in writing I see the same patterns of creation as in painting. I write down lots of things until I notice they form a picture and then I write what seems to be missing in the picture. That way it's much easier to make a story with interesting things happening as they actually were created before the main story. They will fit perfectly because the main story evolves from it.

After I wrote 1½ chapters of the beginning I felt that I had enough material to continue with the

game as a whole. If I had continued here then I would need to create the game after the story which is not a very good idea so the timing was perfect.

Design document

At this point I had already at the same time worked out a design document for what I needed to put into the game. The making of this document was not writing a big massive bible of things. I wrote down only small leads to things I liked and the main structure of which direction I wanted things to develop in. The main goal was to make it as easy to get information from as possible and flexible to rearrange. For example, I wrote sections for the ships history, looks, maneuvering, abilities and combat abilities but all of them was very tiny in text. Just enough to get the overview and to get the mind started in creating more ideas. I'm always in the feeling of not writing to much of things I do not really know anything about. For example. I might have an idea for an ability to the ship. It might only say “Tractorbeam - Pulls or drags an object”. This is enough to develop it more later on. As pulling an object can impact many areas, there are no point in locking the possibilities at this point. How the tractorbeam works might be constricted of what is possible to program and how it looks like can be constricted of how it works. So I only write the mainframe of an idea to make it easy for the development later on and at the same time enough to start working.

I was very enthusiastic to learn more about everything that would end up in my path while entering the second goal which was to make a proof of concept for a game.

Proof of Concept

Already before the project went online, was I ready to do a game on my own. As I do not have much money and worked with this project at my home on my own and wanted to be legal according to piracy laws, I had only the choice of using freeware software. I had already heard and studied that Blender was a 3D modeling tool that had the power of its high end counterparts. Blender also had a game engine which was said to be very easy to use so I learned the basics of it. I knew that the blender game engine was restricted as it uses it's special system of “programming” but it also had the possibility to combine it's interfaced programming techniques with real programming in the form of python which seemed good if you wanted to do more complex stuff. I felt pretty used to programming C++ so python should only be simpler to work with even if I did not know the syntax.

So I was well prepared as I learned Blender and the blender game engine before the project. I did not only have knowledge in my backpack after this learning period. I was able to recycle some of the things in the project, at least the groundwork of it. These were:

- Basic models for the ships.
- Simple interactive movement usable for a ship.
- Very simple AI(*9) for the enemy ship.
- The possibility to add an object from another object and have it disappear when hitting another object such as a shot from a ship.
- A programmed camera movement sequence which makes the camera follow and distance itself to properly cover 2 objects.

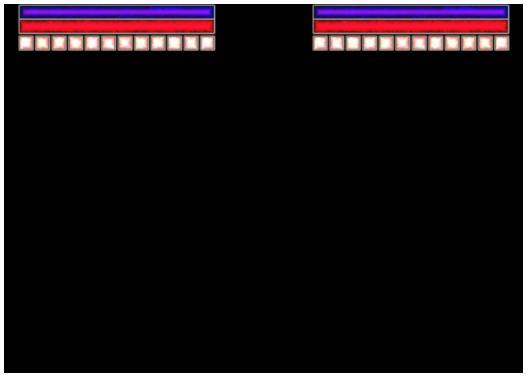
So I actually already had the groundwork for a spaceshooter to the point that I knew it was possible to do. These things were all modified under the development of the real project.

Dynamic text

So I started to develop things that I knew I had to implement. The first thing was to get text into the game. I've already practiced doing this before the project but the technique I used back then did not have the full power I wanted it to have so I had to rebuild it from scratch with a different technique. I wanted to be able to put dynamic text on the screen. That is text which changes according to situation. Such as numbers shifting in the form of score or statistics. I wanted to be able to put every information up in the form as text before using an interface, due to the fact that you can put the text where you want the interface to be to see if the positioning is practical. So I made a mockup of the interface by positioning the dynamic text where I found it suitable.

Interface

First of I made the interface with the possibility to show values in the form of numbers where I wanted to see them just to get it working. As the game developed it got more and more values that I needed an interface for. The first value I had was the HP(Health Point) for the ships. This was as the others only numbers in the beginning. Then I made a rectangle which changed in shape to show a moving bar. Later on the energy system also had a moving bar, but not only that. After the bar went full a square lit up to indicate that you had 1 point you could use to shoot simple shot with. I made the interface show 12 such shotpower points.



The final version of the interface

I waited to texture the interface until I knew that it had all the functionality I wanted from it, then I textured everything at the same time. This also simplifies making the textures seem seamless.

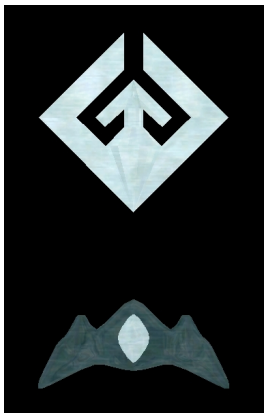
Combining scenes.

When programming this game I made 3 scenes which showed graphics at the same time. Those were the game, the interface and the dialogs. Separating these had many qualities. Foremost it was much simpler to program everything separately. The code did not get cluttered and also much easier to check for bugs. Another thing was that I could much more easily shut down those parts I did not want to see at the moment. I also made a mainloop scene to control the other scenes.

The main gameplay.

Two ships shooting at each other and making this fun. 1 of them being a player, the other a computer. I could easily make it a 2 player game, but then I would miss the knowledge of making an artificial Intelligence, the possibility to develop the game further into an RPG, and I could always very easily make it 2 player later on if I wanted to. This would pretty much be copying the already made player ship.

I already had some things done. Ships that actually could shoot and hit. I mentioned earlier that I
did this during my time learning Blender. What I needed now was to develop this a lot more. First I wanted the ship to glide in a special manner through space. This was just some variables that needed changing as the game engine already has a built in physics engine.



The enemy and player ships

Targeting the player.

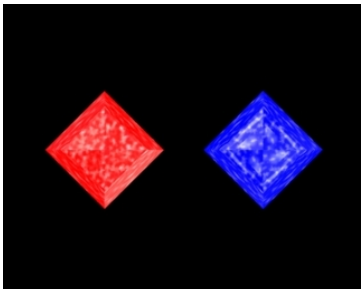
Then I needed a system for the enemy ship to hit the player with it's shots. There was a function in the game engine called “Track To”. This function made an object direct itself towards a target object. So by using “Track To” on enemy ship with the player ship, the enemy always shoots straight towards the player. This sort of function is very common in many games. All enemies in “World of Warcraft” is locked towards a player when entering a battle in exactly this manner. But I felt that this was not enough for me. I thought it looked much to stiff and unnatural so I wanted to make it smoother. If an enemy always would have you targeted, then it also would make it to powerful with shots that were to fast or to predictable to avoid if it used slow shots. But I figured out a nice and simple solution. I made a new object which acted like some kind of sight for the enemy ship. I made the enemyship target this new object which I called “enemySight”, and the sight target the player ship while always moving towards the player. This made the targeting look very smooth and naturally flawed.

Weapons

This is where the main gameplay and balance actually goes. To be able to choose the right weapon at the right time while in combat. My main objective here was to make very different weapons according to gameplay and not looks. Of course a different look on a weapon is a good thing, but if it still acts as every other weapon then it becomes pointless.

Ordinary shot

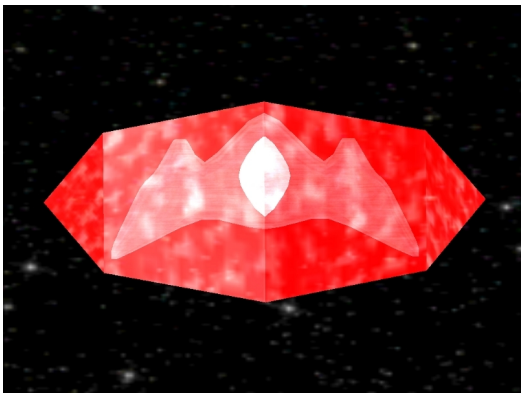
So the first weapon was the absolute basic shot. A ball which moves in the direction it was aimed. It's as simple as it can get it seems, but then you have to consider other circumstances. In this game, it is not only the enemy ship that is moving. You yourself are also moving. This is very common indeed but we have another factor in this game which is more uncommon. This game does not involve shooting multiple targets and the target is as powerful as you. I want the player and the enemy to be able to use the same kind of weapons. This also makes it much more easy to remake the game to a 2 player game.



The ordinary red and blue shots

In a game where you shoot multiple targets like an ordinary side scrolling shooter, the player ship has very fast shots to almost instantly hit it's enemies, and the enemy shoots lots of slow moving shots for the player to be able to move away from. But here the enemy needs to be able to move away as well. So the players shots cant be too fast. But slow moving shots combined with a moving player makes a problem. What if the player flies into it's own shots? Sometime this can be a good thing, but not if it happens all the time. So you want the shots to go a little bit faster than the ship goes. Also to consider is that in reality if you throw something while moving, the speed of the movement combines with the speed of the throw. To achieve this in a game you have to program this physics into the game as the blender physics engine does not consider this. Also, is this really what you want? If you fly backwards in full speed with your ship and shoot, that would make the shot barely go forward. So if you change direction to full speed forward, you definitively will crash into your own shoot. So I chose to make it simple and let the shot have it's own speed without adding the ships to it.

Shields

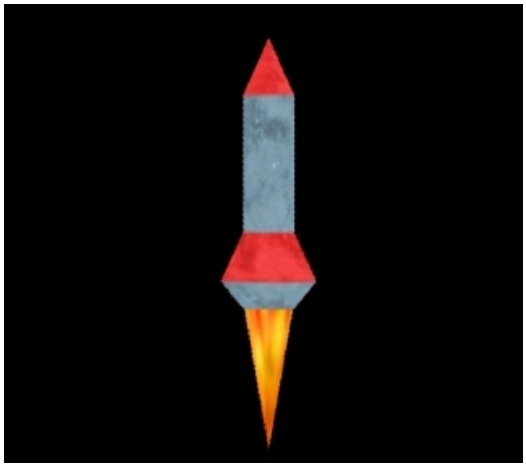


The red shield on the player ship

Basic shots in all honor, but as I want as much skill as possible within every single thing happening, I wanted to develop even this a little bit further. As the ordinary shots are uncomplicated in their use, they should also be simple to avoid. But flying away from shots is a bit boring as you do that almost always with every shot fired against you. So by adding a shield which instantly could block a shot, you suddenly have a new thing to do.

My philosophy is that every single thing you can do in a twitching game should be very easy to do. The hard part shall be to do it in the right moment or choosing the right thing to do. Never forget to implement the possibility for the player to develop skills over time as much as possible. With this in mind I made made 2 different shots with different colors. 1 red and 1 blue. So in order to block a shot you need to use a shield of the same color. So not only did I find a solution for the shield but at the same time upgraded the weapon system with a new feature.

Missiles



A missile

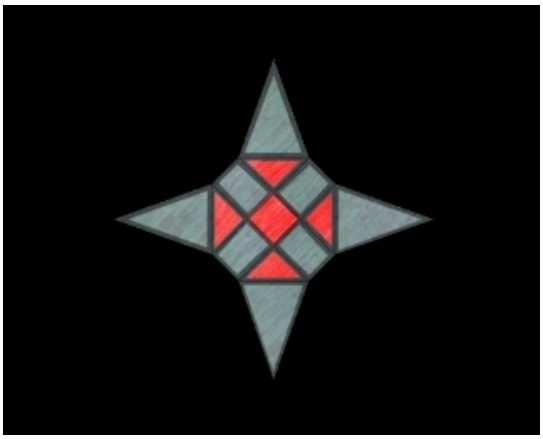
The difference between a shot and an ordinary missile usually is that the missile is slower and explodes on impact, delivering greater damage than a shot. In this case why not the same here. We have already another advantage in this case. The missile gets the feature that it can not be blocked by an energy shield and now the user has to move away to get rid of this threat. But, as usual, why not develop this a little bit further if possible. To put a little bit more skill to both attacker and defender I gave it the feature to self explode whenever the player wants. So it does not even need to hit it's target, only come close enough for the explosion to make a difference. So the defender has to be quick in thought to steer away from the missiles path as early as possible to be sure not to get to close.

There are another kind of missile which is very common. The tracking missile. This one makes it impossible for the player to get rid of it by moving away from it as it always finds it's target. So here we can put in another way for the defender to make safe. Shooting it down with ordinary shots.

Mines

Let's say that you are hunted and while hunted you have a hard time attacking while defending from everything that comes your way. The closer the enemy are to you, the easier he has to hit you. This is where the mines come in. If the attacker get's to close to the defender he can get into serious problems avoiding a dropped mine. But why settle with ordinary mines? Sure they are great but if they can become greater then why not? So I made a similar function as to the ordinary missile but instead of blowing it up on command I made the mines start moving in the direction of the enemy in a straight line. So a mine wont track an enemy while moving, but in it's initial moving face it tracks

the target to go towards it. This can be used to build many sneaky attacks. The mine user can set out a mine and wait for the player to come close enough to have a hard time getting away from it once it starts to move. Another thing is that you can set out lots of mines in lots of places to confuse the enemy into forgetting where all the mines are and thus get to close to one. Also while activating the move functionality every mine on the field starts to move at the same time. These techniques feels very powerful, but with great power comes great responsibility. Because if you put out to many mines, you can get hit yourself. So you can even as the one defending from the mines try to fly in between the mines and the attacker to lure him into shooting himself with them.



A mine

Energy

Now when we got a small but effective set of weapons we need to make a system for the weapons to get a proper use constraint. While it sounds cool to press the buttons as fast as possible to pump out lots and lots of missiles and shots, this will totally destroy the gameplay. So I added an energy system for weapons and shields. First I have an energy bar which always load energy if the ship does not have full energy. Once the bar is full an indicator lights up to more easily show exactly how many bars of energy are ready for use. I made 12 such indicators. Now we can set a price for everything the ship can do.

The shields should drain power as long as they are used. Thus making them the only feature of the ship which is not dependent on full bars.

The ordinary red and blue shots should be the cheapest to use and thus goes of the price of 1 bar. So if 1 indicator is lit up, 1 ordinary shot is available to fire.

All other weapons have different prices set to them and this now becomes a balancing issue.

Balancing the gameplay

Balancing the gameplay is a time consuming but fun part of the game development. There are lots and lots of things to consider when balancing but every game has it's system for balancing. Here are a list of what can be balanced in this game.

- HP of the ships
- Damage from the weapons.
- Area of effect from the weapons
- Speed of the ships and weapons
- Cost of the weapons
- How weapons get destroyed
- AI of the enemy

Sound

When I made the sounds to this game I did mainly 2 things. I searched the internet for license free audio files and rebuild them to my liking and asked a friend to give me some sounds that he thought sounded like shots etc and rebuilt them. The explosion sounds was taken from the internet and

heavily rebuilt. Most shot sounds came from my friend. What I changed was mostly length, pitch, volume and added different effects. Usually I did not use the whole sound that I got but took parts of it that I liked. The humming of the shields was my friends sound which I made into a seamless loop.

Making a loop is automatic in Blenders gameengine. But starting a sound which goes into a loop after a while was not. When I did the sound for the missile I got into some problems because of this. First I wanted a loop sound for when the missile flies. I took the sound of an aeroplane running it's engines at the same spot on the ground to make the sound as easily loopable as possible. I pitched it and changed it's speed so that it sounded smaller. When the loop sounded alright I went for making the initial starting sound of the missile leaving the ship. All this sounded great but making it play was not that easy with Gameblender. The problem was that after a sound started playing there was no way to use a trigger. This simply means that there was no way I could tell the program, after the sound has come to an end or played for X-amount of seconds, do this. But sometimes when there is no way, you do it anyway. I simply started 2 files. 1 was the loop and another the start of the loop. The startup sound was louder than the loop and as they sounded pretty much the same, the startup sound faded nicely over into the loop. So both sounds started at the same time but the startup sound stopped and the loop continued.

Background

I based this game on a simple top down view. It gets much easier to program and it's simpler to create a better gameplay. What we see in this view is ships, weapons and explosions. But if the ships move in the same speed and direction, it seems like they are standing still. So we have to add a background so that we can notice that the ships actually are moving. I modified a photo of space and textured a plane with that and put it beneath the ships. Now we have a background, but as the ships move, we will soon come to the end of the image. To manage this I made the image seamless so that you can tile it forever without noticing the edges. So when you fly to the edge, a new image overlaps the old one.

Explosions

Once a weapon hits or get destroyed an explosion is very nice to add. So I tested some different methods. First test was to make an explosion as an object like the ships. This can be a good idea but to make it look good, you usually have to add many of these objects, have them transparent and animate them. This can become pretty time consuming and a full object also takes a lot of power.

So I did the very simple method of only using a textured plane with the background transparent. I made the texture by combining different photos of explosions my mother took of fireworks. You can always combine lots and lots of planes to fill over each other and have them randomly move. That makes the explosion seem a lot more realistic. This is technique is often used for particle effects in general, such as fog, smoke, fire, etc. As a plane is only visible from the top and not the sides you always have to view it from the top. As this game has a top down view, this becomes no problem. But if we would need to rotate the camera, the explosion would not become visible. The simple solution to this small problem is to track the plane to the camera. This is always done with particle effects in 3D games.

Explosion

Animations

Animations are fun to do and they usually add more eye candy to a game than more graphics, so they should not be neglected. But as this is candy and does not make the game, this still had a pretty low priority. But as I wanted to make a game and manage to create everything myself, I had to do atleast something.

This game is full of movement. The ships move, the missiles fly, but until now they are not animated in the sens that I have done a pattern for how they move or change in shape.

I started to make a very simple animation for the fire on the missile. I had a coneshaped, textured object attached at the missile to resemble the fire. Then I simply made it smaller and bigger in a loop to make it feel more alive.

Another thing I wanted to do was to develop the enemyships rotor. The only thing this would need to do was to rotate so that was more than easy.

Intro sequence

When playing games it might be easy to forget that you actually need a beginning and an end. The player needs time to get ready to start, and to know when finished and what happened. So I added a camera for each ship and so that I could show them before the fight started and added the text “GET READY!” and “FIGHT!”. And when the fight was over I simply made the game show the surviving ship with the text “YOU WON!” or “YOU LOST!”.

What should be noted here is that as I did not have all time in the world to do everything I wanted so I ended up with a boring topview of the models in closeup not to distort the background. What I could have done was make a copy of the ships, set a skybox(*10) around the ships and make a cool looking animated camera view as an intro sequence.

The limits of the engine

Until now I have told of everything that has gone according to plans. But creating a game is bound to have problems occur because of the cheer number of things that is being done. A game is very complex. And making a new game always get you into a new challenge as you probably do not want to do exactly the same game you have done before.

I prepared the making of this game for months, learning the Blender Gameengine. I did lots and lots of different things to learn as much as possible of how things could be done. That way I had a lot of ideas of how to develop things once I knew what to develop. But as it's impossible to cover everything a human mind can come up with, I came to a halt, trying to make something which would seem not being possible to do with the Blender Game engine.

As I did a space shooter I wanted to do laser weapons. Sounds simple enough especially with the thought of what I have already achieved, but this was a mountain I was unable to climb.

I wanted a laser shot which begins from the ship and ends on the target. So all I seemed to need to do was, make a plane, grab the end of it, and move it forward until it collides with a target. That way it would always be connected with it's cannon. But there was no way of pulling the side of a plane. I tried to find a way to move separate vertexes but no. I tried to connect an armament to the vertexes, but when I moved the armament, I moved the whole object. It seems impossible that there would be no way to move the edge only, because how could you otherwise animate an object at all with or without an armament? The thing is that you can animate the edge anyway you want, but creating an animation makes you loose all control over where the edge should stop, making it go straight through the enemy. So I tried to build the laser in segments instead of animating it into size. This would not be a perfect solution but it was a technique used in the old days of doing lasers in spaceshooters with 2D graphics.

But cutting the beam was not the only problem. Another even worse problem showed up. If the

child object has enabled collision detection then the laser starts rotating every update the amount the parent has rotated in the game from it's origin. That's probably because the engine does not update any change of a parent to it's children when the game is running. Why this is connected with the collision detection probably involves the physics engine which is a separate engine used by the Blender game engine. So what happens is that as soon as the ship rotates, the laser start rotating around the ship with great speed.

At this point it seems quite hopeless. The beam have to collide with the ship. But this is a game and not reality, so we do not actually need the beam to collide with something, what we need is a verification of some kind that the beam should have hit it's target. So, I tried to figure out a solution without using collision detection. I found a function called “Ray Sensor”. This function checks if anything is in between the origin and a set range in a straight line. So I had the non-collision detection laser plane and the ray-sensor used as it's collision detection and it worked, sometimes. Why sometimes was of course the question that popped up. This could be a bit tricky to see, but I noticed that the sight I gave the enemy ship to target the player blocked the path of the ray-sensor. As of now I had used an invisible cube for the sight to be able to make it visible and see how it operates to check for bugs as in this case. Blenders game engine has more than 1 way in which to set it's collision. So I turned of everything in order to really make it non collide able. But it still collided. But as an object actually can exist without having any body, I removed the cube around the sight. What you are left with is called an empty, because there are nothing there. This nothing can be used to hold information used in the world but without any body. You can see it as naming an empty space and by knowing that name you can access information about it. So instead of just setting a variable straight into the memory like in any common programming language, in Blenders game engine you also always give it a position and this is called an empty. But after making the sight empty, the ray-sensor still noticed it and there is no way to set an empty to non collide able as there is nothing there. So after all the hard work and figuring out all workarounds, the limit in this case became the engine and there was nothing left that I could do but to leave my precious laser behind.

Bugs

This is where I cant stay away from talking about all the bugs that started to develop. The thing is that the Blender Game Engine is very much not finished and much of it's functionality is not complete or bug free. Also production of this engine has not been highly prioritized. Many great people that is working with Blender has tried to continue the engines work and planned it's development in great detail, but they had to lay everything on ice as they could not afford to develop it. So the bugs in it has been like they are for years. And bugs in an engine is not a good thing. That is, you cant fix them if you do not star to develop the engine yourself. In the case of skilled programmers with a lot of time on their hands, working with Blenders game engine is not a bad thing as it is open source and you actually can do this. Without open source you are totally stuck. You have to try to find workarounds for the problems you get. That is how I did it as I did not have the time to develop the engine (which would be very interesting otherwise).

Crashing when running the executable

One of the first bugs I managed to get was a pretty nasty one. Everything worked fine when starting the game inside the engine, but when I exported it to an executable file it crashed. I checked doing this both in Linux and Windows and both had the same problem. The nastiness of this bug is that it is not often you export the game to see if it works as it should. So by the time you do it, you have worked for weeks, adding and removing content and changed settings all over the place. The bug can be anywhere. So it was only to start removing content until it worked. This is not as easy at

it might sound as much of the content is dependent on other content to exist. Knowing this is very good for new developers so that you try always try to keep things separated and have them work on their own. But I had managed pretty good to keep the code independent so I found where it was. Well, I found out where it was, but even though I knew where it was, it was still hard to figure out why it actually crashed the game. First thing I found was that the ship made the game crash after I had removed everything but the ship. I had removed all logic from the ship. That is the possibility for the player to fly or shoot with it, or make it explode etc, etc. So it was only an object and it still crashed the game. I tried to export it into another totally new game and it worked. But I managed to locate a more exact placement of the bug. When texturing in blender you have a small list of slots in where to put textures. When I put a texture in a certain slot everything crashed which was an engine bug. So for you programmers out there reading this my guess is that it probably was some kind of dynamic array which had forgotten to assign memory for this slot somehow. Well, very hard to say of course, but anyhow, it was a bug which I could not fix but do an easy workaround for.

Crashing when the playership explodes

This was another annoying engine bug which also gave me big problems. I located it to the “Track to” function. When the “Track to” lost it's target, as in when the ship got destroyed, it crashed the system. As I could not fix the engine, I had to find workaround yet again. And as I did not want to stop using the “Track to” function because of the very important use of this fine little function, I had to stop using it the moment it's targets existence ended. This was not easy because of the design of how the engine handle the code. What you can not do which is very unusual is asking an object a question and get an immediate answer. Usually when you program and the code finds a call for a function, the computer enters the function and starts working in the new location. This cant be done here which seems very odd. Let's say that the computer is reading the code from the sight of the AI ship and I want to ask the player ship if it exists, then the question does not get it's answer instantly, but sends it like a letter to the ship. So not until the computer is reading the code of the ship, the ship can then write a new letter back with the answer “Yes I'm alive”. So not until the computer reads the code of the sight again it will be able to get the answer. And when the sight get that message, the ship might already be dead. So the only thing I could do was make the ship send “Yes, I'm alive” all the time so if the sight did not hear a “I'm alive”, it did not dare to start the “Track to” function. But the ship still could die after it had sent the message, so there was no way to actually fix this 100%.

The Blender team did fix so that the “Track to” function did not crash if it had no target 1 week before my presentation. But that update made other things crash which worked before so I was not able to correct it in time. The irony...

Collision and physics

The Blender game engine uses in it's turn a physics engine called “Bullet”. Having physics on an object can do a lot of things. It handles the collision detection, can give an object weight, density and gravity. It can also make an object solid so that when it collides with other objects it bounces off and does not go through them. What Blenders game engine or Bullet can not do at this point is turn off and on collision detection between certain objects. This creates a whole lot of trouble. Let's say that you want the ships to be solid so that they can collide with each other. You also want the shots to be solid so that they can collide with everything too. As you cant exclude something from the collision even the player ship will collide with it's own shots. That makes it impossible to let the shots start from inside the ships cannons to shoot out of the ship.

All this is actually even more complicated as you want some objects being able to bounce of each other and others just to register that collision has been detected. Also all objects needs to have

collision detection activated to get acceleration into their movement.

Also, the mesh or “body” of the object is not what is colliding. What is colliding is called a bounding box. Blender has no ability to show you exactly how the bounding box looks like and you can only change the shape to a number of predefined set of basic shapes. These are: box, sphere, cone, cylinder and convex hull. These shapes will be scaled in X,Y & Z according to the model. The only change you can do to the box itself is scaling the total size of the box. The location of the box is focused around the pivot point of the mesh. The pivot point is the point in which the mesh rotates around. The pivots location is because of this very important. Let's say you have the mesh of a human character and want it to face backwards by rotating it 180 degrees, you want the character to rotate around the center of it's legs. This is no problem as the center of the mesh is probably the same as the center of the legs. But let's say that the character have a very long tail. In that case the center of the mesh would not be in the middle of the legs but somewhere on the tail. So by putting the pivot point where the legs are instead of in the middle of the mesh, the mesh will rotate around the legs as the mesh always rotate around the pivot. But remember now that the collision box centers not in the middle of the mesh but around the pivot point. So when moving the pivot point you move away the collision box from the center of the mesh.

I guess at this point you begin to see that it all feels like a set of domino bricks. If you change 1 thing, it will be connected in some way to something else and lots of things will start to get affected which you did not intend in the first place and it takes a lot of time to go through all the different possibilities.

Workarounds

Some things can be fixed quite easily if you think outside the boundingbox. Luckily you can turn on and off collision detection during a game. This made it possible for me to fix the problem I had that the ship collided with it's own shots. I simply had no collision at all for the shot until it got outside of the collision box.

For my ship I have shields that when turned on can block and absorb the shots supposed to hit the ship. I have a red shield that absorbs red shots and a blue shield that absorbs the blue shots. If you use collision detection on a shield surrounding the ship, then the shield and the ship will push away each other as they cant be at the same space as they are both solid and you cant set either of them not to collide with the other one. What I did to fix this problem was very simple. I used the collision of the ship and turned off the collision for the shields so when the shields were active, I turned them visible and set the ship not to get hurt by the shots.

Letting go

One thing I really wanted to have in the game was the ability to shoot laser. Making a space shooter and not having a laser seems just wrong to me. Most if not all space shooting games have it.

The laser I wanted to create was a simple beam starting from the ship and stopping at collision. So how do I do that? It may sound simple, especially if you've seen it happening before but it all depends on how constraint the engine are you are working with. A nice idea would be to for example, making a box that represents the laser, put the pivot point at the start of the box, scale the box in the direction of choice and when collision occurs, stop scaling. You might already see that this is not possible with the engine I used as the collision box is bound to the pivot point so by putting the pivot at the end of the laser, you would hit things behind you and only half of the size in the direction it was pointed.

So if I cant scale the box, then maybe I can drag the end of the box by using an armature. Armatures are used to build skeletons for characters. You can add part of a mesh to an armature bone so by putting the end of the laser to a bone I could then drag the laser towards it's location to

scale it. The thing was that there is not way in the engine to move an armature by itself so that did not work.

I started to think how did they do it in the old days of all the 2D shooters. What they did back then was to segment the laser into small cubes. Think of this like Domino. All the bricks are the laser segmented. When you shoot, you simply let the first brick fall and in the case of the laser, the first becomes visible. The brick/segment hits the other brick/segment and tells it to fall/show and this will continue until it collides with something and thus stopping the laser and getting a collision at the same time. While this solution was not a preferred choice as it would be noticeable that it was built in segments and that the motion would be choppy, it at least would be a usable workaround to get the well needed feature into the game which would enhance the gameplay a lot. What happened here was that I found another problem when parenting the segments to the ship that is setting their location according to ships location so that the segments will follow the ship. When the ship rotated, the location of the segments started to update their location each frame for the amount of rotation the ship had done since it's origin instead of it's new location. What this means is that the segments started to spin around the ship. The more the ship had rotated since start, the faster the segments spun. As there was no way to tell the the game engine to update each segments location from the ships new location instead of it's old, also this idea had to go.

But I was not about to give up! When one get stuck like this I think it is good to go back to the beginning, forget what cant be done and start focusing on what can and make it really simple to see where it leads to and then maybe go from there. I could set up a mesh representing the laser, covering the whole screen, turn it on and off and stop caring if it stopped at the point of collision. Now I just needed to find another way to detect collision. I found a function in the game engine called “ray-sensor”. What this little function does is to check in in the direction of choice and the length of choice if it hits anything and what. This sounded to me exactly what I needed!

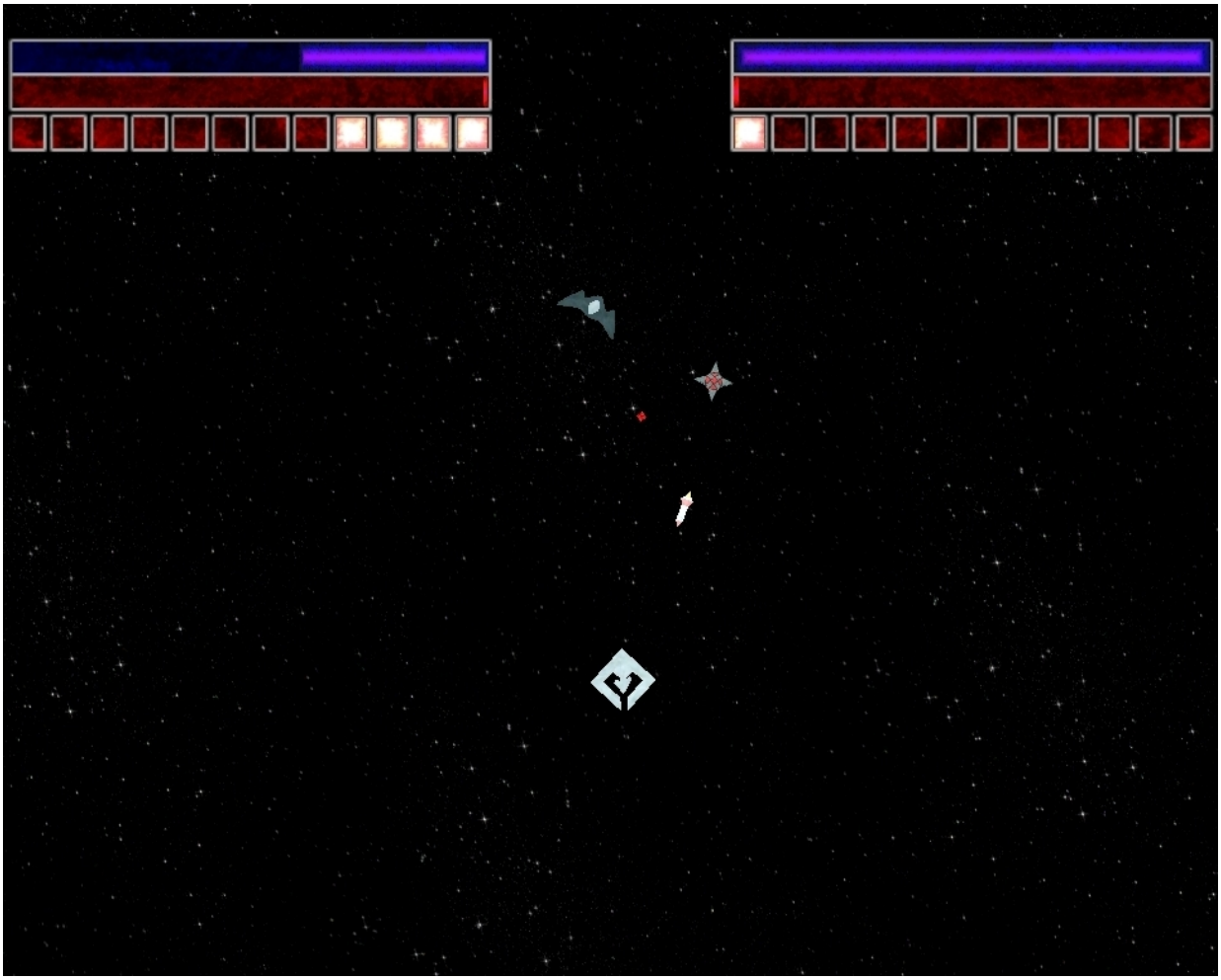
I finally had my laser. It did not stop after collision but it worked. Or so I seemed until I noticed that it did not always work. Well, that was very strange. Sometimes it worked and sometimes it did not. If there is one thing that cant be accepted in a game, it is a feature which do not work the way it is intended and thereby destroys the gameplay. I had to figure this one out or get rid of my beloved laser. Suddenly I knew why it stopped working all of the sudden. The ray-sensor was hitting the invisible targeting system I gave the enemy ship. There are more than one way to detect collision so I turned them all off. This did not help. I had only needed to have a mesh for the targeting system as a reference to understand how it moved around so that I could calibrate it. So I deleted the whole mesh and only had an empty object left which was enough for it to work. The ray-sensor still hit the empty targeting sight. At this point there was nothing I could do to fix it. As the ray-sensor detected any object with or without a mesh, this means that it would detect every single shot or object in the game and thus making it impossible to get around the problem. After a whole week of working with this, I had to let it go.

A complete and working “Proof of Concept”.

When finally the project came to an end I had written the groundwork for a great story usable as a standalone book or the story for my game with one and a half chapter of the beginning already finished and the main storyline and ending already figured out.

I proved that the hardest part of creating the game was possible for me to do and that it would even work as a stand alone game without the big RPG element.

I had successfully gone through most of the important processes experienced when creating a typical computer game production, making all content myself except the music which I already have experience of creating both solo and in a group.



Ingame screenshot

Chapter 5

Final words - English version

My education was a very young one. As most new creations, the first steps are tottering and unsure, but with a new mind comes new ideas and thereby new possibilities.

To be able to go through this whole process has been fantastic. I really learned a lot being able to let free my creativity. Getting a whole year to study by myself is what made all this possible. The more freedom, the more possibility to go further in the studies but it has to be a person with non-ending motivation and a mind with a strong non-static vision. Getting stuck is a danger for anyone and every small thing associated about getting stuck can bring the motivation down or lead the production to a halt. Having a non-static goal is what makes it all possible. Then you can choose the path of least resistance and with highest motivation. That's why the most effective way for a teacher or school to make the students learn is not by forcing them to work, but instead giving as many possibilities as possible for the students to succeed and helping them find their vision.

What I would love to do, is to get sponsored to continue creating prototypes for games. To make the groundwork for a game so that everyone understands the true core of the concept by being able to feel the gameplay instead of just looking at pictures or reading something from a design document. So that a company would know exactly what is needed to get it done and what to do to make it fun. I feel that this is my strength.

What I probably need to do is to create more portfolio material as this production has still been a learning process and thus has produced more knowledge for me than material as proof of what I am actually able to do.

What I hope for is to keep the freedom to create whatever I feel is fun and that the world would want or need. My dreams are not to make money but to make my dreams come true. Thereby I do not crave much more than what I need to survive and to do my work without any pressure. I have a lot of ideas that I really want to develop and that I feel will bring lots of fun and knowledge to the world.

Slutord - Svensk version (Final Words - Swedish version)

Min utbildning var väldigt ung. Digitala Spel klassen var den andra av sitt slag. Som de flesta nya skapelser var de första stegen vacklande och osäkra, men med nya sinnen kommer nya idéer och därmed nya möjligheter.

Att jag haft möjligheten att gå igenom hela processen har varit fantastiskt. Jag lärde mig väldigt mycket genom att få låta kreativiteten flöda. Att få studera ett helt år för mig själv är vad som gjort allt det här möjligt. Ju mer frihet, ju mer möjlighet att gå djupare in i studierna men det behöver vara en person med oändlig motivation och ett sinne med en stark, icke-statisk vision. Att fastna är en fara för vem som helst och varje liten sak associerad med att fastna kan blockera motivationen eller få produktionen att stoppa. Att ha ett icke-statiskt mål är vad som gör det hela möjligt. Då kan man välja vägen med minst motstånd och högst motivation. Därav är det mest effektiva sättet för en lärare eller skola att få studenterna att lära sig att inte tvinga dem att arbeta utan istället ge så många möjligheter som möjligt att lyckas och hjälpa dem finna deras vision.

Vad jag skulle vilja göra är att bli sponsrad att fortsätta skapa prototyper för spel. Skapa grundstommen för ett spel så att alla förstår kärnan av konceptet genom att kunna känna spelglädjen istället för att se bilder eller läsa något ifrån ett design dokument. Så att ett företag skulle veta exakt vad som behövs för att skapa produkten och få den skoj. Jag känner att detta är min styrka.

Vad jag troligen behöver göra är att skapa mer portfolio material då detta trots allt varit en inlärningsprocess och därmed gett mig mer kunskap än bevis för vad jag har kapacitet att skapa.

Vad jag hoppas är att få behålla friheten att skapa vad jag än känner är roligt och som världen vill ha eller behöver. Mina drömmar är inte att tjäna pengar utan att få drömmarna att bli verklighet. Därav kräver jag inte mycket mer än vad jag behöver för att överleva och arbeta utan press. Jag har många idéer som jag verkligen vill utveckla och som jag känner kommer att bringa mycket glädje och kunskap till världen.

References & Links to external information or applications

Applications

Blender - www.blender.org
Blender is the free open source 3D content creation suite, available for all major operating systems under the GNU General Public License.

Gimp - www.gimp.org
GIMP is the GNU Image Manipulation Program. It is a freely distributed piece of software for such tasks as photo retouching, image composition and image authoring. It works on many operating systems, in many languages. (more...)

Audacity - audacity.sourceforge.net
Audacity is free, open source software for recording and editing sounds. It is available for Mac OS X, Microsoft Windows, GNU/Linux, and other operating systems.

Documentation & Miscellaneous information

Google Search - www.google.us
Google search is a Web search engine owned by Google, Inc., and it is the most used search engine on the Web.

Blender wiki - wiki.blender.org
This wiki is the primary documentation for Blender

Blender Summer of Documentation: Learning the Blender Game Engine
http://wiki.blender.org/index.php/BSoD/Introduction_to_the_Game_Engine

Documentation for the GameLogic Module of the Blender Game Engine
http://www.blender.org/documentation/pydoc_gameengine/PyDoc-Gameengine-2.34/index.html

Blender 3D: Noob to Pro
http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro
A wikibook for Blender.

Blender Underground - blenderunderground.com
Homepage dedicated to Blender made by Apollos. Has some of the greatest video tutorials and a great community.

BlenderArtists.org - blenderartists.org
BlenderArtists.org is the most popular community site dedicated to the free, open-source 3D animation program Blender. On the site, users of the program can discuss their artwork, ask questions and find resources such as textures and tutorials.

BlenderNation - www.blendernation.com
Fresh Blender News Every Day.

Apendix

Wordlist

*1 - Sidescrolling - The game is seen from a third person perspective view from the side with the camera following the player steered object.

*2 - Shooter - A computer game where you shoot lots and lots of stuff entering the screen.

*3 - 3D content - Graphics which can be viewed from any angle.

*4 - Unwrapped (Unwrapping) - When making a 3D model you start creating the form of the model. To be able to paint a texture onto the model, you usually unwrap it. This is also commonly called “skinning” because of the likeness of skinning an animal and flattening the hide. Doing this to a model makes it easier to paint a texture onto it.

*5 - Textured (Texturing) - Is the art of painting a texture onto a 3D model.

*6 Twitching - Twitch gameplay is a type of computer or video gameplay that tests a player's reaction time and precision. Action games such as first-person shooters are typically composed of twitch gameplay. However, other genres can involve twitch gameplay as well. For example, Tetris is a puzzle video game that gradually speeds up as the player makes progress.
Twitch gameplay keeps players actively engaged with quick feedback to their actions, as opposed to turn-based gaming that involves waiting for the outcome of a chosen course of action. Twitch can be used to expand tactical options and play, test skill in various areas (usually reflexive responses) and generally add difficulty (relating to the intensity of "twitching" required).
A game that rewards reflexes over tactics, strategy, and problem-solving is sometimes called a twitch game.

*7 - "Melee" has been adopted and popularized as a gaming term to encompass all close-quarter fighting. For instance, gamers will refer to "melee attacks" as opposed to "ranged attacks" in the context of first person shooter, real-time tactics, computer role-playing games, and paper RPGs. Basically the term is used to describe directly striking an opponent at ranges generally less than a few feet with fists, feet, knives, swords, hammers, maces, the butt of a rifle or any other sharp or blunt instrument with the intention of causing harm.

*8 RPG - Role Playing Game.
An electronic role-playing game is a broad genre of video games. These games are originally derived from traditional role-playing games, especially Dungeons & Dragons, and use both the settings and game mechanics found in such games.

*9 - AI (Artificial Intelligence) - a system that perceives its environment and takes actions which maximize its chances of success and thus simulating real life intelligence.

*10 Skybox - This is a textured box which surrounds the content shown. It creates the illusion of a big environment usually with a horizon and a sky. The texture of the box is drawn in a way made to seem round and not square. A Skybox can also be sphere shaped.

Projektplan

Rickard Remelin – rire05 – DS05 – Projektplan för kandidatarbetet våren 2008

Spelkoncept / Speldemo

Grundläggande plan:
Projektet går ut på att utgå ifrån ett spelkoncept och se vart man hamnar. Man bör se hela processen som ett träd där man utgår ifrån roten av trädet och hela tiden väljer den gren som är starkast och har bäst riktning för att nå så långt upp som möjligt.

I beroende på hur spelkonceptet utvecklar sig förändrar sig inriktningen på projektet enligt den gren man hamnar på. Om konceptet har en god grund för att bli en stor tung titel kommer all tid för projektet att gå ut på att skapa ett koncept. Om det visar sig simpelt för en enkel produktion kommer jag att övergå till att producera titeln. Gameplay element kommer att skapas och testas vid ett mindre koncept för att kunna demonstrera konceptet för en projektgrupp och vid nog av tid sättas ihop till ett speldemo.

Ju mer man sätter upp en struktur för projektet, ju mer låser man alla möjligheter för dess utveckling. Det är av yttersta vikt att projektet får ta den riktning projektet behöver för att alltid kunna nå den del som ger mest utdelning. Dock är hela projektet baserat på en specifik arbetsteknik och denna kommer att följas till 100%

Detta projektet planeras allt eftersom det utvecklas. Ju mer det arbetas på, ju mer kommer att dokumenteras för dess framtid. T.ex. efter första veckan finns mer information om vad konceptet går ut på och då går det att skriva mer på mallen för projektet. Projektet kommer alltid att ta fasta på sluttiden och basera alla mindre delar efter detta. Det viktigaste är dock inte dokumentationen över vad som ska göras, utan vad som har gjorts. Att dokumentera vad som gjorts ger en effektiv och realistisk uppfattning för hur långt man hinner komma med det som återstår. Inte förren ett projekt är klart kan man fastslå exakt vad som skulle gjorts till 100%, men ju närmare man når slutet, ju exaktare kan man få resultatet till vad som ska åstakommas.

· Syftet med projektet
Att lära mig utveckla ett spel så långt som möjligt dvs från koncept till färdigt spel. Ett klart spelkoncept är det enda kravet, allt annat är en bonus.

· Motiveringen till detta projekt.
Det här projektet har jag förberätt sedan höstterminens start 2007. Allt jag gjort fram tills nu har varit baserat på att klara av att skapa ett spel på egen hand. Allt har gått enligt min plan tills vidare och nu börjar jag äntligen arbeta på det jag kämpat för att kunna göra.

· Vad studenten för egen del avser att uppnå med projektet.
Då jag arbetar ensam på detta projekt kommer jag att få en inblick inom allt gällande skapandet av spelet. Koncept idéer, koncept skisser, koncept dokument, artifaklista, skapandet av 3d modeller, texturering, ljud, gameplay och allt annat som ingår i skapandet av ett spel. Dock är det planerat att jag ej kommer att nå ända fram men målet kommer att vara att skapa ett eget speldemo ifrån grunden och minimumkravet att få fram ett färdigt spelkoncept. Jag kommer även att att agera producent/projektledare genom att vid behov söka material till spelet utifrån. Det vill säga, jag kommer att hämta arbetskraft och material

från andra personer och blir den som samordnar och strukturerar.

· Projektmetod
· Projektets arbetsgång, hur ska mål och syfte uppnås – hur studenten och gruppen ska gå tillväga, hur arbetet planeras att fortskrida under våren för att få en kunskapsutvecklande karaktär.
Så länge jag gör vad jag ska kommer mitt arbete att vara enormt givande. Då jag arbetar ensam har jag ingen begränsning av någon annan utan allt är upp till mig att se till att jag arbetar hårt.

· Kriterier för delexaminationer – vad ska examineras, när ska examinationerna ske.
Det är inget jag kan svara på just nu. Jag kommer att lämna in för varje vecka en rapport och då vill jag ha information om vad jag ska redovisa av vad jag gjort av det som står i rapporten.

· Tidplan för projektet
· Delmoment, delexaminationer och andra avgörande tidpunkter för arbetet.
En tidsplan är ej möjlig att ge då jag inte har ett fast mål utan flera. I beroende på vad jag lyckas uträtta förändras totalt tiden för projektet. Hur den än förändras kommer jag ändå alltid att ha olika delmoment färdiga. Vilka detta är har jag ingen aning om förren jag gjort dem.

· Observera att tidsplanen behöver vara relativt exakt för att medieteknik ska ha möjlighet att knyta eventuell extern handledare till projektet.
Jag behöver ingen extern handledning. Allt jag gjort hitintills har jag klarat av på egen hand och det är inte lite.

· Eventuell ekonomi för projektet
o Eventuell ekonomisk värdering av kostnader för projektet.
0Kr

· Behov
o Behov av extern handledare eller annan extern hjälp m m.
Inget behov. Jag vet att jag klarar det jag valt att skapa.

· Risker med projektet
o Hur gruppen avser att hantera de risker för projektet som kan eller bör förutses.
Inga risker. Då jag inte har någon annan än mig själv, finns det ingen risk att någon annan inte arbetar som den ska. Jag har även satt ut ett flexibelt mål som är mycket simpelt för mig att klara och samtidigt helt omöjligt att uträtta fullt ut. Ju hårdare jag arbetar, ju längre kommer jag, och jag vill komma långt.

Tidsplan över de 14 arbetsveckorna.
En ungefärlig tidsplan för hur projektet kan se ut:

Vecka 1:
Spelkoncept

Vecka 2:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo

Vecka 3:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo

Vecka 4:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo

Vecka 5:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo

Vecka 6:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo

Vecka 7:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo

Vecka 8:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo

Vecka 9:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo

Vecka 10:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo

Vecka 11:

Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo

Vecka 12:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo

Vecka 13:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo

Vecka 14:
Alternativ 1: Spelkoncept
Alternativ 2: Spelkoncept & Gameplay demo
Alternativ 3: Speldemo