BACHELOROPPGAVE:

Android phone Game - Alien Destruction

FORFATTERE:
Håvard Kindem
Kristian Nordhaug

DATO:
Vår 2010

Sammendrag av Bacheloroppgaven

Tittel:	Android mobil spill - Alien Destruction	Nr: -
		Dato: Vår 2010
Deltakere:	Håvard Kindem	
	Kristian Nordhaug	
Veiledere:	Simon McCallum	
Oppdragsgiver:	Høgskolen i Gjøvik	
Kontaktperson:	Erik Helmås, erik.helmas@hig.no, 61135000	
Stikkord	Bachelor, Android, 3D, Spill, OS, Telefon, Mobiltelefon	
Antall sider: 90	Antall vedlegg: 8	Tilgjengelighet: Åpen

Kort beskrivelse av bacheloroppgaven:
Prosjektet var å utvikle et spill for det nye operativsystemet Android. Spillet er laget for mobiltelefoner med innebygget aksellerator sensor, hvor man bruker bevegelse av telefonen for å styre i spillet

Summary of Graduate Project

Title:	Android phone Game - Alien Destruction	Nr: -
		Date: Vår 2010
Participants:	Håvard Kindem	
	Kristian Nordhaug	
Supervisor:	Simon McCallum	
Employer:	Høgskolen i Gjøvik	
Contact person:	Erik Helmås, erik.helmas@hig.no, 61135000	
Keywords	Bachelor, Android, 3D, Game, OS, Phone, Cellphone	
Pages: 90	Appendixes: 8	Availability: Open

Short description of the main project:
This project was to develop a game on the new Android OS platform. The game is made for cellphones with built in accellerator sensors, where one utilize movement of the phone to control the game.

Abstract

This bachelor project was to make a working game on the new Android OS for mobile phones. We started out creating a game idea (see Appendix:F). When we felt confident we had a high quality concept to work with, we went to the implementation stage. We had no experience in programming for Android, hence we spent a large amount of time learning the system. The needed functions of a game engine was made. A lot of work went into investigating how to optimize the code to run on system. This report will cower these methods used in our game implementation, and our experiences using this new platform. In conclusion, mobile games using this Android platform will only get better and better, as the community develop better utilities such as true OpenGL ES game engines. Developing games for these devices are very similar to normal PC games, but the restrictions to GPUs, screen size, and memory are really tough constraints. Overall we are pleased with the end result, given the limited resources for the project.

Preface

We would like to thank Simon McCallum and Jason Mackie for all help during the bachelor work.
A thank to Helge Nordhaug for all the feedback on the report.
And last, a thank to Øyvind Nordstand for feedback during pitch presentations.

Contents

List of Figures

1 About the Report

The target audience for this report is mainly our supervisor, Simon McCallum, and the external examiners who are to evaluate this project. As such, the report will be quite technical and the reader is expected to have knowledge in the area of software development.

The Report is written in in latex, using a bachelor template [1] by McCallum.

1.1 Main Chapters

This report covers an introduction to Android and our game story, and our objectives with the game. It includes our scope of the work, and the plans made for execution of the project as a whole. Further, it continues with our quality assurance setup, and the main chapter, implementation, covering all code we have written. We have also provided a chapter on optimalization for the Android system, and at last we have a summary of the whole developement prosess for the project.

1.2 References

For references in latex we have used the program JabRef [2] reference manager, that uses BibTex as its native format.

1.3 Appendixes

- Log: individual daily work.
- Log: Weekly group meetings.
- Log: Meeting with supervisor and principal
- Pre Class Diagram, written before the implementation had started.
- Pre Project Report with Gantt Diagram
- Game Design Document
- Class Diagrams for the whole code base.
- Gantt Diagram for Finished project

2 Introduction

Mobile phones in general are not created for gaming support, with its lack of processing power and memory. But the recent few years the mobile phones have taken a huge leap, and are now finally starting to get powerful enough. After the release of HTC Dream in 2008, the market for these types of smart phones running Android has skyrocketed. Today there are over 65 000 phones running Android sold every single day [3]. The goal for this project where to develop a functional 3D game for this new platform, and to improve our overall programming experience and expertise.

2.1 What is Android?

Android is an open operating system for cell phones. The source code was acquired by Google when they brought the small company Android.inc in 2005. In 2007 there was a consortium made by many well known companies, formed as the Open Handset Alliance (OHA) led by Google. It consist of more then 60 mobile and technology companies, such as HTC, Intel, LG, Marvell Tech Group, Samsung, Motorola, Nvidia, Sony Ericsson and Texas Instruments, etc. Their main goal is to develop open standards for mobile devices. Android is the first mobile platform to use this new standard. Android is built on the Linux kernel, that gives the system most of its core services such as memory management, network, security, basically it provides a stable environment for its architecture. The Android OS has three main parts. The underlying kernel providing the API for the applications. On top of that is a middle layer, providing functions that makes the activities running able to coordinate with each other and exchange data. The last layer provides the key elements for the mobile phone, programs like address book, browser, map applications.

2.2 Why choose Android?

The most appealing thing about this system is its open nature. The Android API, SDK and NDK is free and available for anyone who wishes to develop applications for these mobile devices. A second thing is the useful architecture. All programs are of equal rights and level, meaning everything in the system can be replaced by other third party programs. Further the ability to share data between the applications, that makes it possible to link pictures to geographic locations, or use the scroll bars implemented in one application directly into another application, without rewriting any code. Also, the SDK includes everything needed for Android applications, as well as plug ins for the Eclipse development tool, which also is free to use. Last, Android code is written in Java, which is one of the best documented programming languages there is.

2.3 Objectives

The main objective is to develop a mobile phone game of high quality that will attract a lot of new users and strengthen our marketing position within the mobile phone gaming industry.

This thesis is specifically focused on the underlying graphic engine's functionality, and does not focus on other areas such as design, modeling and sounds, as this truly is exceeding our knowledge in those areas. Though it is worth mentioning that some effort has been used to create the game design document, and made our own 3D models.

2.4 Performance

The game performance objective is to make a game that will be robust, and run with good graphics without any issues. It must also meet the requirements for Android applications and support the standards for correctly installing, starting, running, pausing and exiting the application. The game should also be designed to enable future releases and more content to be added, without rewriting the code base of the application.

2.5 Learning

Our goal is to gain knowledge of the Android system, and mobile phones in general, and to increase programming skills. Android applications are written in Java, which we have some experience with, but there are so many differences from a regular Java program to an Android program ,that we are starting close to scratch. Regardless, we worked hard to complete the development process in the best way possible.

2.6 Target Audience for the Game

We are aiming for players from 10 to 30 years of age. We have kept the requirements set for the ESRB E1 rating.

2.7 Game story intro

It is late fall 2012 and it is the players first day at the COAT(Center of Alien Technology) training facility, orbiting earth. COAT has secretly been exploring space since the early 1980s, when they discovered a way to break the time-space continuum. No one knows that a gigantic invasion is impending, and as the aliens manage to break through the outer defenses, it is up to the player whether the population of earth will survive the day!

3 Scope of work

Our task was to go through the whole software development process, from planning to release of an Android application. We focused on using the same methodologies as the gaming industry, to secure a stable and high quality product, with good documentation and re-usability.

3.1 Developement

Planning

The goal was to make frequent small releases as the project is divided into iterations, as we use an agile developement process [4]. We also scheduled meetings, as iteration planning starts each iteration. We performed testing of the iterations and got feedback from friends and classmates.

We established and followed our game design document during the whole process. A Gantt diagram E was made by the end of our learning phase, with planned schedule and progress of the project. As we were using an Agile methodology, we also had the possibility to modify the plan as we went when we felt something had to be done in a different way.

We have adhered mostly to the initial plan, but have some deviations, such as increased time to solve the User Interface that were introduced to achieve the final goal.

Design

The focus was on simplicity, we tried our best to keep everything generalized, understandable and explainable. It was also important not to try implementing models before they are scheduled, as it quickly could end up as a time drain. While designing we where conducting spike solutions. This means we tried implementing the most risky parts as separate programs, before writing code right into our project.

Coding

As we where only two people, we integrated often. We did not have many options in testing environments, as we own the same type of cellphones. This gave us the disadvatage not to be able to tests our game on different cellphones with different Android OS versions.

Testing

Testing was performed on each module to make sure it is safe to go further on in the project. We also had testing when reaching each milestone.

Architecture

The architecture is based on modularization and we split up the development into multiple modules to have better control over the programming process. These was as followed, in this order:

- Data Structures

- Game Menus
- Model and Image loading
- GUI
- The Game World
- Input Handling
- Physics
- Artificial Intelligence
- Top scores
- Campaign

3.2 Refinements

As we had very limited amount of time for the development of our game, there where some things that had to suffer. As our goal was to make a fully functioning game, we focused as much as possible on this. However, it is a very time consuming job. Therefore, we had decided that the development of the mentioned modules had to be considered when the other more important activities was completed. As such, we had to skip some functionality:

- upgrades of ship
- aditional game modes
- integration with Facebook

3.3 Constraints

Developers have reported that making software for the different operative system versions creates a large amount of issues concerning the different aspect ratios on the cell phones. For this project we gave it our best to make the game work for all phones, but the only hardware we have for testing, is the HTC Hero. This phone is running Android 1.5, HTC had promised Android 2.1 for the phone in the early spring, but this did not happend. This is why the game can only run on Android 1.5 at this moment.

The delivery date for the project is set at 25.05.10.

3.4 Monitoring and Reporting

We have tried to have status meetings every Monday to discuss last weeks work and and review the targets for the coming week. We present a weekly status rapport, which contain the topics discussed. We also log almost every working hour on the project with all the things we have done. This is only for us to have the possibility to go back and see what we left out, and what we used our time on.

4 Quality Assurance Setup

4.1 Development Environment

We have been working with Eclipse for Java, with Android and SVN plug-in installed. This is the default method used for developing Android applications. With the Android Software Development Kit, we are able to quickly run the program on our phone. This makes the testing and debugging of the application much faster and easier, as we also get debugging information out of the phone, directly into Eclipse. This also enables us to test every new implementation with the gyro-sensors only found in the phone itself.

4.2 Developement Tools

Today, there are two different methods of developing applications for the Android platform, using the Software Developement Kit(SDK) or the Native Developement Kit(NDK). The SDK uses well documented Java code, while the NDK uses Ansi C or C++ and is loaded through the JNI API. In terms of speed, see:7.1

4.2.1 Installing the SDK

Setting up the SDK is fairly simple and is well documented throughout the internet. The setup itself takes about 15 minutes to complete and all the needed executeables work well with eachother.

First of all, you need to install the Java Developement Kit and Eclipse. When this is done, download and install the Android SDK [5]. When this is done, you need to add the install directory of the SDK to the PATH variable, found in:
$Computer \rightarrow Properties \rightarrow Advanced\ System\ Settings \rightarrow Environment\ Variables.$

Now you need to add the ADT plugin for Eclipse. This is done, by going to the Help tab and selecting Install New Software and adding $https://dl-ssl.google.com/android/eclipse/$ to the sites and marking the ADT plugin for installation.

The final step is to set up one or more devices to develop on, this is done through the AVD manager, by running the $setup.exe$ in the SDK install directory.

4.2.2 Installing the NDK

Setting up and using the NDK is poorly documented on the internet and takes some. After a large amount of research we found a guide that does it all right. Android NDK [6] First of all, you need to download the NDK [7]) and extract this to a suitable location. If your are running windows, you need to download Cygwin [8] and choose to install the make utility when asked to install Cygwin, if you run a linux OS, then you probably have this from before. Finally, open a terminal

(Cygwin on windows), change directory to where you extracted the NDK and run the command $make./build/host - setup.sh$. The NDK is now ready to use through the JNI API.

4.3 Standards and Source Code

We followed the standards of Android OS build 1.5, as many of the Android phones have not yet got their update, including our test platform, HTC Hero. The commenting was done by the code's author. This way, we made sure not to forget important parts, and run into problems later in the process.

4.4 Configuration Management

Our group used the Tortoise Subversion tool for configuration management, as the IT-Services at Høgskolen I Gjøvik provide these for no extra cost. The documentation however, was automatically maintained by the JavaDoc[9] application, as it is all written within in the code, hence this saves us both time and effort.

4.5 Project organization

4.5.1 Responsibilities

While both of us had to join in on basically everything, we set up four main responsibilities. These acted more like guidelines and to set the group member's focus on specific tasks, rather than limit the work in certain areas.

- Group Leader: Kristian Nordhaug
- Lead Game design: Kristian Nordhaug
- Lead Programmer: Håvard Kindem
- Web master: Håvard Kindem

4.5.2 Other Roles

Key persons:

- Supervisor: Simon McCallum
- Principal: Øyvind Nordstand (Kunnskapsparken In Hamar)

5 Design

Description on the game design itself is located in the Game Design Document, found in the Appendix: F. This chapter briefly cowers the small part conserning the initial stage of our implementation.

We started out doing basic Android programming tutorials provided from NeHe Productions [10] and others that had been rewritten at InsanityDesigns [11] to be applied to an Android OS. This was a nessesity we could not skip, as we had no previous experience with Android. In total we spent about two weeks, but at the same time, we wrote down class diagrams 8 for our own game idea. This was done with only the tutorial knowledge, and compared to the end product 9 it was more work then we anticipated.

6 Implementation

6.1 Sounds and Music

The sounds in the game is controlled by our own small implementation of the existing functions for media in the android platform. This had to be done as we need to preload everything before the game starts, to minimize any garbage collection that Java start automatically. The sounds are placed in a SoundPool class, and can be started-stopped and played over each other, as it is controlled by the OS, and has its own threads. The only issue with this is the game's own rendering threads have more priority then the sounds. so if the game is struggling, trying to play the sounds really fast, will force the OS not to play the sounds until it has free time.

To have a constant music playing without lag is even more important then having the frame rate drop down, so the music is totally controlled by Android, and the threads have higher priority then the game graphics itself.

6.2 Scene Graph

A scene graph is basically a node tree of different matrix operations, used to represent the world we are drawing. We implemented the following kind of nodes:

- Default (Empty) Node
- Translate
- Rotate
- Scale
- Apply Matrix
- Draw Model
- Draw Skybox
- Draw Waypoint
- Enable Mode
- Disable Mode

All of these nodes pushes the matrix before doing their specified operation and pops the matrix stack after its children have been parsed. This means that you can put two models together, translate one of the models above or behind the other and it will stay at this position, following its parent, unless specified otherwise.

This gives us an easy way to add visual weapons, engine particles, sprites and lights to objects in the world, not having to be concerned if the object is placed at the correct spot.

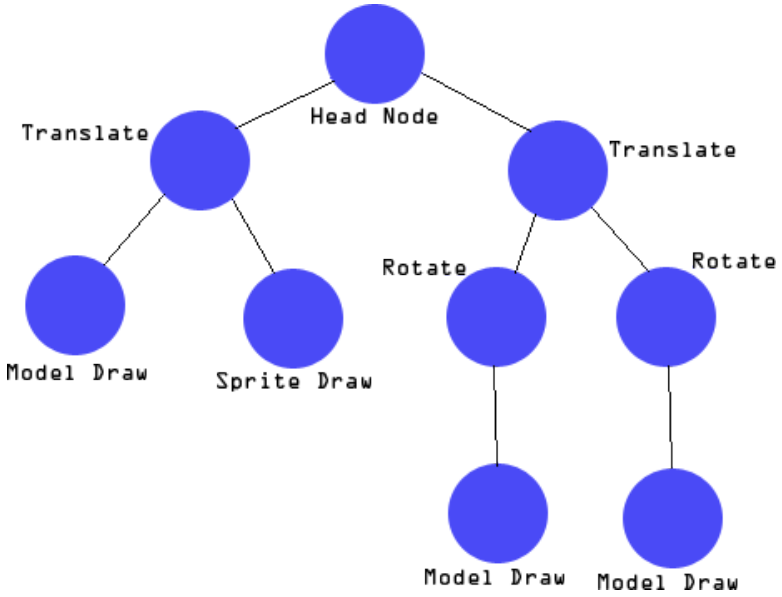


Figure 1: An example scene graph

Parsing

Our scene graph is parsed depth first, to apply all the desired matrix modifications before eventually reaching the leaf node, drawing the object.

```
public void parse(){
        perform();
        for(int i = 0; i < children.size(); i++){
                children.elementAt(i).parse();
        }
        cleanup();
}
```

6.3 Input Handling

Gyro sensors

Pitch indicates the tilt of the top of the device, with range -90 to 90. Positive values indicate that the bottom of the device is tilted up, negative means the top is tilted up.

Roll indicates the side to side tilt of the device, with range -180 to 180. Positive values indicate that the left side of the device is tilted up, and negative means the right side. [12]

Yaw is the compass heading in degrees, range from 0 to 360 degrees, whereas 0 is North, 90 is East, 180 is South, 270 is West.

The gyro sensors is a tricky thing to get working right. It drains huge resources of the CPU, so a limit on the times it actually updates had to be made.

6.4 Flight Controls

There are mainly two different ways of controlling flight, which depends on whether you are within an atmosphere and the current gravity. In space you have vector flight, as the gravitational forces are minimal and you have no aerodynamic limitations. Within an atmosphere on the other hand, you are bound to the aerodynamic principles and must compensate for the gravitational forces.

Vector Flight

The concept is not as well known outside the groups of interest, so this would require a longer learning phase than the normal gravitational flight. As you travel in one direction, you will continue to do so until another force is applied. The main formula for calculating movement here is vector multiplication and kinetic energy calculations. The formula for the kinetic energy is as follows:

$$E_v = Mv^2$$

This means that the energy needed to stop an object, with the mass of 100kG traveling at $50\mathrm{m/s}$ is: $100\mathrm{kG} \times 50^2\mathrm{m/s} = 25\mathrm{kN}$ in the opposite direction of the current heading.

To calculate the new direction, after adding additional force, we need to multiply the vectors where we set the length as the kinetic energy. This is done by adding the new vector to the old one, getting the new speed and direction.

$$Vt = (V_{current} \times E_{current}) + (V_{new} \times E_{new})$$

$$E_{current} = \sqrt{Vt_x^2 + Vt_y^2 + Vt_z^2}$$

$$V_{current} = \frac{V_{xyz}}{E_{current}}$$

Heading Based Flight

Heading based flight, uses the same principles as vector based, but it is simplified to make the control easier. Say that you see a target, you lock on it and press the boost button. That automatically applies enough force to make you orbit the selected target, if the button continues to be held in, otherwise the ship would continue to apply force until the direction of travel was the same as the player specified when he pushed the boost button. As well as a long learning phase, this requires a great deal of reference points around the world, to navigate efficiently and not confuse the player.

Gravitational Flight

Gravitational flight is what we know as normal flight on earth, this is bound by the aerodynamic principals that must overcome gravity. This means that the generated lift must be greater than the gravitational force of the earth at all times, if the plane is to avoid crashing to the ground. Another issue with gravitational flight is that the lift area must be calculated is it is to be correct.

For example: a plane that is turning, will have lower lift than a plane that is level. This is due to the aerodynamic surfaces and their alignment to the gravitational force. The mathematics behind the forces that lift the plane is complex and time consuming without cheating a bit.

When it comes to the control principle, you generally have 3 options: roll pitch and yaw. These behave differently when the plane is level than when its turning, as well and must also be calculated when the vessel moves.

Conclusion

From these three options, we chose to go with a simplified form of gravitational flight. More specifically, we use the pitch and roll to control the spaceship and ignore the gravitational forces. The reason for ignoring the forces, was to save CPU time and avoid complex calculations for each object in the world (as all objects generate a gravitational force).

We found that controlling the spaceship as you normally would a plane, the time of the learning phase was decreased, as it makes more sense to most people to control flight this way. Another argument for using this method, is that we would need a lot of reference points in our world to make the heading based flight to avoid confusion as well as some sort of HUD element to clarify what direction the player would be traveling.

Vector based flight was also and option, but were discarded as it would be very difficult to control the spaceship this way, you would need a way to look around as well as controlling how much force you applied. When we have such a small screen and only can rely precisely on the gyro sensor, as the compass would be too inaccurate and limit the game play when you physically can not move freely around.

6.5 Drawing in OpenGL ES

Introduction

Drawing elements in OpenGL ES requires pre-set buffers for all the desired information that you want to draw. The different arrays can then be activated and the pointer set to the desired buffer before drawing.

Initializing

Initializing the needed arrays for OpenGL ES are done using the ByteBuffer.allocateDirect() function with the specified size(the number of total bytes, hence the multiplication of 4, as this is the size of a single float). When the space has been allocated, the order of the content are specified not to be modified, before the data is inserted and the pointer position is set to the beginning of the buffer again.

```
ByteBuffer byteBuf = ByteBuffer.allocateDirect(vertices.length * 4);
byteBuf.order(ByteOrder.nativeOrder());
vertexBuffer = byteBuf.asFloatBuffer();
vertexBuffer.put(vertices);
vertexBuffer.position(0);
```

Drawing

There are two ways to draw objects in OpenGL ES, using glDrawArrays() or glDrawElements(), but we will get back to this later, in 6.5. The arrays needs to be allocated and the pointer must be set to the buffer, containing the data to be drawn, before any of the draw functions can be called.

Here is an example on how to draw a vertex array with the corresponding normals and texture coordinates.

```
gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer);
gl.glNormalPointer(GL10.GL_FLOAT, 0, normalBuffer);
gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, texCoordBuffer);

gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
gl.glEnableClientState(GL10.GL_NORMAL_ARRAY);
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);

gl.glDrawArrays(GL10.GL_TRIANGLES, 0, vCount);

gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
gl.glDisableClientState(GL10.GL_NORMAL_ARRAY);
gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
```

glDrawArrays() vs. glDrawElements()

The glDrawArrays() function, is a linear draw function, it iterates over all the arrays that are currently activated. This function requires that there are a set of data for each vertex. For example: if we have 4 vertices (a total of 12 floats) that we want to draw and we need both normals and texture coordinates for the object, we would need a total of 4 normals(12 floats) and 4 texture coordinates(8 floats) as well.
This would then take up a total of $N \times 4(3+3+2)$ bytes, where N are the number of vertices.

The glDrawElements() function, uses indicies instead of just iterating over the arrays. This means that it would require less memory, as the vertices, normals and texture coordinates can be used multiple times. The total number of bytes taken is now: $(N \times 4 \times 3) + (N \times 4(3+3+2)) - ((Dn + Dv) \times (4 \times 3)) - (Dt \times (4 \times 2))$ where N is the number of RAW vertices. Dn, Dv and Dt are the number of duplicate normals, vertices and texture coordinates.

Then it comes down to the conclusion. While the glDrawArrays() function takes up more memory on larger models, it makes up for it in the iteration speed. As it does not require to pass any index data between the arrays, it is much more CPU friendly. The glDrawElements() on the other hand, takes up less memory for largest models. BUT: this is not really noticed, unless the models have have a lot of shared data or the complexity of the models are exceptionally high for a mobile device. When it comes to CPU usage, the glDrawElements() seems to take up a lot of time. This is basically because there are a lot of information being passed between the arrays (3×4bytes) for every vertex being drawn.

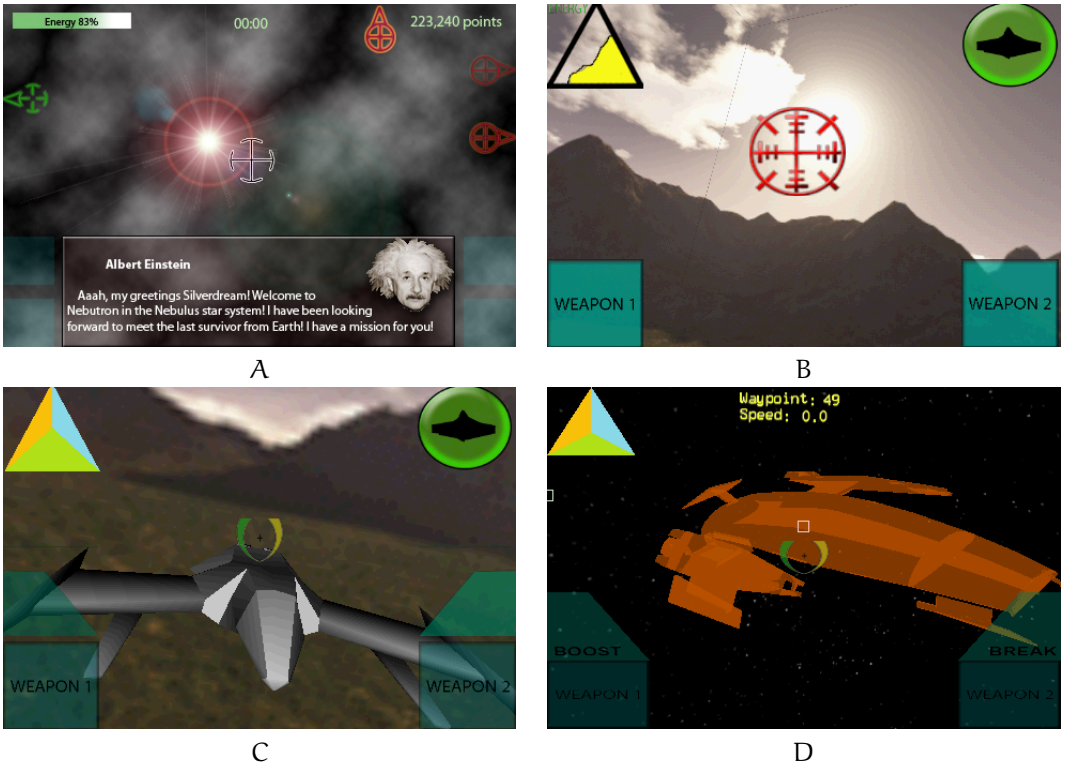


Figure 2: Images of the GUI during development. A this is our concept art made in Photoshop. B taken 2th April, C taken 27th April, D taken 18th May

We chose to use glDrawArrays(), as we intentionally made our models with a low polygon count and we spent a lot less CPU time, drawing them.

6.6 Graphical User Interface (GUI)

The GUI have been in constant development throughout the project. Figure 2 shows some work in progress.

Energy

To find out of the point is inside the triangle or not, is not quite easy. We have to compute the barycentric coordinates Figure:3, also known as area coordinates. These are made by making vectors out of ABC, then create dot products, and normalize. Then i'm left with 3 scalar's U,V and S. These correspond to each vector of the triangle side's. To check if the screen touch is close to a corner, its now just to check the size of these scalar's. the closer one is to 1, the closer it is to that corner. to find center, all three would be $\frac{1}{3}$ in size.

```
//Compute dot products
dot00 = v0.Dot(v0);
dot01 = v0.Dot(v1);
```

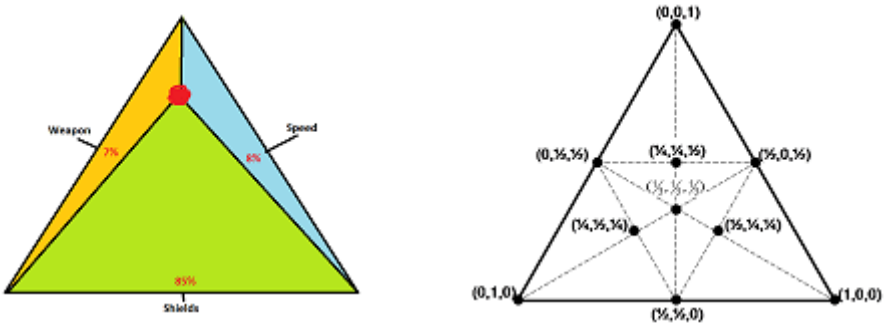


Figure 3: Barycentric coordinates used in the energy bar.

```
dot02 = v0.Dot(v2);
dot11 = v1.Dot(v1);
dot12 = v1.Dot(v2);

//Compute barycentric coordinates
invDenom = \frac{1}{(dot00 * dot11 - dot01 * dot01)};
U = (dot11 * dot02 - dot01 * dot12) * invDenom;
V = (dot00 * dot12 - dot01 * dot02) * invDenom;
S = 1 - U - V;
```

If two of the sides are bigger then 1, the touch is outside the triangle. This gives the game its micro,macro,and automatic ship controls. By making tests on what point is closest to what was pressed, and automatically move it. [13]

Cross hair

The cross hair is the center of focus at all time while playing the game. That is why we wanted to place the most useful information there, the targeted enemy's current total health and energy. This information will be controlled by the tracking system used, same that will control missiles. It will display the current closest enemy. Due to the lack of time, we chose not to implement this further, and rather focus on other parts.

Text in the GUI

The implementation of text in Android can be done in different ways One option is to place a TextView over the SurfaceView. This is really slow and takes extreme recourses from the system.

Second option is to render common strings to textures, and simply draw those textures. This is by far the simplest and fastest, but the least flexible.

Third, and our option was to write our own text rendering code based on a sprite. This is very tricky, as there are allot of different features that should be in such a class. These classes are usually given off-the-shelf or open-source libraries, but for Android there still are none available. Therefore our text class does not support line-breaks, variable width, max width, as it is not needed for our game. As of now, you can change the font image that is loaded, but this have to be hand-made, to be supported by the Android 1.5 system, that can only take images with size powered in 2. ie. $64*64$, $128*512$. In the future Android OS versions this has been changed, and

it can load any images of any size.

Since everything has to be preloaded to minimize Java garbage collection, the font class generates and saves texture coordinates for all 256 characters possible from the bitmap file.

This is also done for the premade texts vertex points, but we experienced some nasty memory corruption trying to avoid all possible uses of 'new command' in the loops, so the text that has to be changed aka. waypopint info and speed, is updated each loop.

6.7 Model Loading

First, we started off by using RAW models, this was implemented and used in the beginning. After a while, we started having issues with the loading time of our models, as there was a lot of calculations needed to generate all the normals. As well as the loading speed, the RAW format does not store the texture coordinates of the models, so we were stuck without working textures. Due to the reasons above, we chose to use the OBJ format instead. Now, all the needed calculations was done beforehand, we had the needed texture coordinates and we saved a lot of space on the size of our models. The OBJ format stores the vertices, texture coordinates, vertex normals, faces and materials. As the work to make a OpenGL ES model loader is so complex, we have not yet implemented the loader with ability to get materials.

|

6.8 Waypoints

As navigation in space can be very disorienting, we decided to implement waypoints to make it easier to navigate between the objectives. The waypoints are represented as a green geosphere and are shown one at a time. The waypoints are loaded into a dynamic array at startup and are loaded in sequence, so that the waypoint first mentioned, will be the first waypoint to visit. The update function does a squared range check on the current waypoint and if it is within range, it sets the next waypoint as the selected one, before it updates the translate node to the new location in the scene graph.

Although, having waypoints made the navigation easier, it was useless if we could not see the waypoint (ie. it was off the screen or outside the max range) so we implemented the tracking(Section 6.9) for it at a later stage. |

6.9 Tracking

Introduction

In our game, we decided to track the closest enemy and the next waypoint on screen. Because of this, we needed to implement functions to translate the world coordinates to eye coordinates, relative to the player. As OpenGL ES 1.0 does not have the ability to save the current view matrices, we needed to to this mathematically instead. The logic behind this is quite simple, as for the implementation, it was a bit worse.

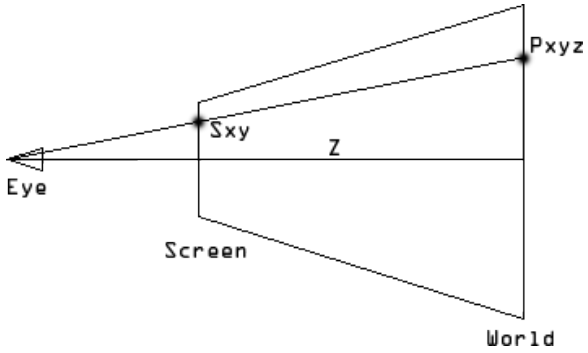


Figure 4: World to screen coordinates

Implementation

The first thing that needs to be done, is to calculate the eye coordinates of the point in the world. First, we store the length of the vector from the player position to the object we want to track, in world coordinates. Now we calculate the dot product between the desired axis and the normalized vector from the player to the object. Multiply this vector with the original length of the vector from the player to the object and you have the eye coordinates of the object. This is now repeated for each of the axis.

Now we have the eye coordinates of the object, all we need to do now is to divide both the X and Y coordinates with the Z coordinate, hence moving the coordinates one unit away from the current eye position. Finally, divide by the X and Y coordinates by the aspect ratio, so that the values range from [-1, 1], do our calculations for determining if the point is on screen or not and do a line interpolation to the screen if its not.

Conclusion

This the basic source code, used in our project.

```
public void track( Player player, Point3 target, int color ){

        Vec3 tempVec = target.oMinus(player.getPos());
        float targetLen = tempVec.length();
        tempVec.normalize();

        float dotProduct = (float)tempVec.Dot(player.getDirection());
        float eyeZ = (float) (targetLen * dotProduct);
        dotProduct = (float)tempVec.Dot(player.getDirectionNormal());
        float eyeX = (float) (targetLen * dotProduct);
        dotProduct = (float)tempVec.Dot(player.getDirectionUp());
        float eyeY = (float) (targetLen * dotProduct);

        float testX = eyeX/eyeZ;
        float testY = eyeY/eyeZ;
```

}

6.10 Triggers

Although the triggers are not implemented in our release of the game, the structure for it is planned out and designed.

Concept

The triggers are in the game to control completion of objectives, handle world events and GUI messages, based on the location, player health, enemies dead etc. The idea behind the trigger system is to use a digital version of the the cause and effect theory. This gives us a total of three classes:

Trigger Classes

Cause:
This class will have a lot of subclasses, that has different checks for the player location, enemy location, enemy status, time, objectives completed and so on. The check function returns true or false on the check.

Effect:
The subclasses of this class will have a perform function, used to start events, like killing the player, complete objectives, finish or fail the mission, spawn enemies etc.

Triggers:
This class holds one instance of a cause subclass and one instance of an effect subclass. It will contain a check function, that checks the function of the cause instance and performs the desired effect.

Usage

The different cause and effect combinations are loaded from the XML spreadsheet and imported into an array of Triggers. The triggers array will then be parsed and checked on each update of the world, then handled accordingly.

6.11 Quaternions

Introduction

In our ship control, we started having strange issues with the ship, when it rotated around the y axis, specifically. After some research and discussion, we figured out that this was because of something called Gimbal lock. The Gimbal lock occurs when two of the 3 gimbals end up at the same position, so that it can only compensate for 2 axes instead of all three. This is the same problem as Apollo 11 had on its decent towards the moon.

The solution to this in the real world is to add a forth gimbal, we did this in our game by using quaternions instead of Euler angles.

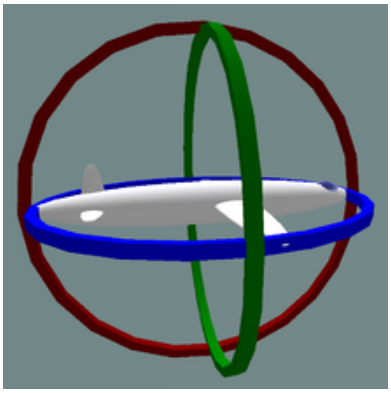


Figure 5: No Gimbal lock

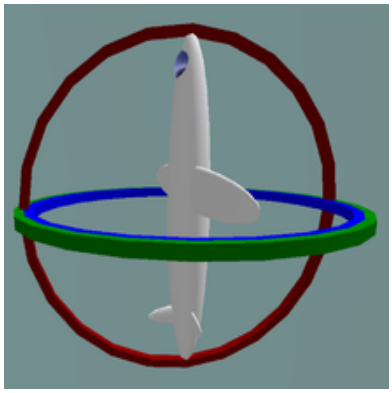


Figure 6: Gimbal lock

Usage

One of the nice things about quaternions is that they can simply be multiplied to get the total rotation as a result with only 24 operations and the total matrix can be made from just 32 operations. Although the math for using quaternions is simple, understanding quaternions can be a bit worse. We applied a quaternion based system on all the objects in our world, generating rotation from the gyro sensors for the player.

6.12 Parsing XML

There are multiple ways of reading XML in Android, the two different parsers are SAX[14] and the XMLPull[15]. We chose to use the XMLPull reader, as we found it easier to use.

The Code

```
XmlResourceParser xrp = context.getResources().getXml(R.xml.testlevel);

while(xrp.getEventType() != XmlResourceParser.END_DOCUMENT){
        if(xrp.getEventType() == XmlResourceParser.START_TAG){
                if (xrp.getName().equalsIgnoreCase("EXAMPLE")) {
                        // HANDLE THE START TAG
                }
        }
        else if (xrp.getEventType() == XmlResourceParser.END_TAG) {
                if (xrp.getName().equalsIgnoreCase("EXAMPLE")) {
                        // HANDLE THE END TAG
                }
        }
        else if (xrp.getEventType() == XmlResourceParser.TEXT) {
                // HANDLE THE TEXT BETWEEN TAGS
        }
        xrp.next();
}
xrp.close();
```

Comments

While the parsing itself seemed to run at a decent speed, we soon figured out that it got very messy, keeping booleans for each tag, to know where the text we were currently reading, fit in. As all the tag names in XML are strings, every if test needs to check for atleast parts of the string, reading larger documents would slow down the performance severely.

7 Optimization

7.1 Speed Comparisons

After a while of development, we started having efficiency problems with our game. It was ignored at first, but as our code got more complex, it got worse. We did a lot of research and testing before we finally found out that the trigonometry part of the math library, used in Android 1.5 was basically broken.

To figure out how bad it really was, we made a test project to compare the trigonometry up to other options.

Testing Environment

The application we made was set up with three options: Java's own cosine function(SDK), the cosine function in the C math library(NDK) and finally we implemented a cosine lookup table function. The way we used the functions, was the following code:

```
float check = 0.0f;
for(int i = 0; i < 100000; i++){
        check = (float) Math.cos(i);
}
```

Testing Specifications

The devices we had access to was a HTC Hero(running Android 1.5) and a HTC Desire(running Android 2.1). Here are the specifications:

HTC Hero

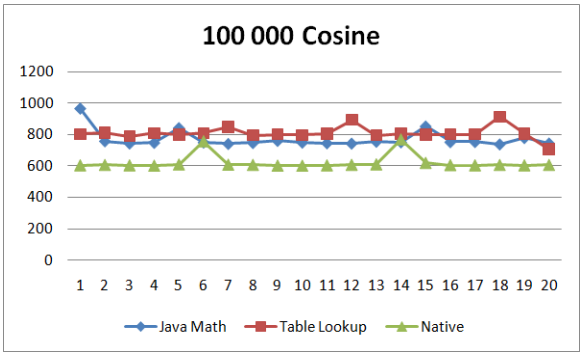
- CPU: Qualcomm®MSM7200A™, 528 MHz
- RAM: 288 MB
- Display: 320×480 HVGA resolution

HTC Desire

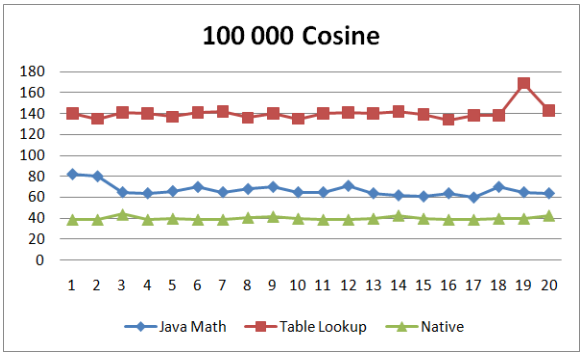
- CPU: 1 GHz
- RAM: 576 MB
- Display: 480×800 WVGA

Test Results and Conclusion

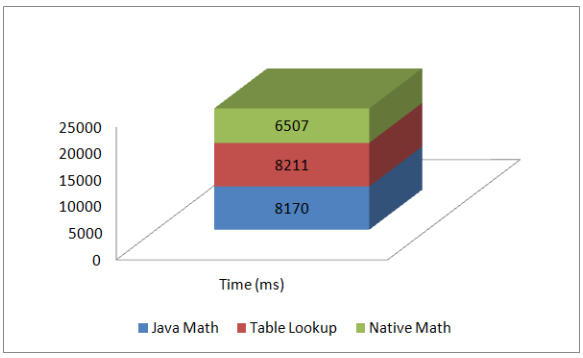
As you can see from the figures 7, the Java math and the lookup table on the Hero, has no significant difference in performance. Compared with the NDK on the Hero, we clearly see that there is an issue with the trigonometry functions, as the lookup table seems to be inefficient as well. On the Desire, we see that the time needed to calculate the same procedures take a lot less time. Notice the difference between the Java math, lookup table and the C math on the two devices. The ratio between each are almost identical on the Hero, while the lookup table performs much slower on the Desire. Although the Desire has a better processor, We can conclude that Android 2.1 has been optimized a lot, compared to Android 1.5 and it's Java version. It is also worth mentioning that the newest test results from Android 2.2 has shown a stunning 5.5x increase of speed [16]. This is expected due to Android being switched to a JIT-compiler. Java byte-code is compiled to native code once before execution, as opposed to interpreting the byte-code every time that part of the code is run. The issue is that floating point on the old NDK does not utilize the VPU (float point processor) of the new ARM CPU, while 2.2 Java code does. Only future will tell if a 2.2 NDK will be faster then a JIT compiled Java program.



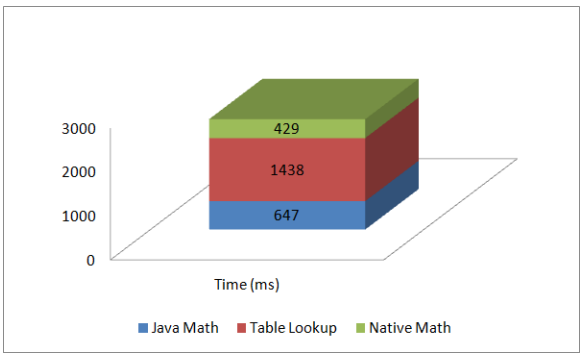
A



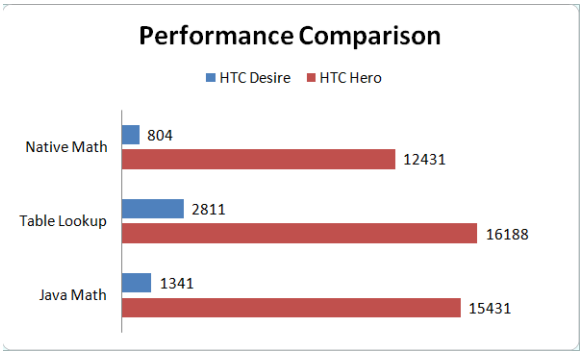
B



C



D



E

Figure 7:
A: 20x 100 000 Cosine on the HTC Hero
B:20x 100 000 Cosine on the HTC Desire
C:1 000 000 Cosine on the HTC Hero
D: 1 000 000 Cosine on the HTC Desire
E: This show the speed comparison between our two devices.

8 Summary

8.1 Assignment Evaluation

We think that we have developed a good idea with a good market potential. Looking back on the difficulties we have had with the project such as the efficiency issues, we could have solved these by programming with the NDK instead of the SDK. This would have solved most of the issues we had, but would have caused us to spend more time on the programming, which then would have hindered us from developing the game to the stage that we have.

Some of our code sections are hard to understand, as we had times that we were in a good flow and flew through a lot of code quickly. Most of this is now documented and cleaned up, but there are still a few sections like the XML parsing that are very messy, due to the massive amount of tests that needs to be checked.

We never got as far as the Facebook integration that is mentioned in the design document as we did not have enough resources to implement all the features. What we have is a nearly complete 3D game engine for Android that can easily be used to further develop our game.

8.2 Evaluation of the group work

Our group has been a great one and we think that the difference in our personalities have given us a much broader perspective on the possibilities and goals of out bachelor thesis. Mainly we have worked on different sections at any given time and had a continous conversation on what should be done next and shared tips on how to do things, as well as given to eachother feedback while we worked.

During the time we have worked on this project, we have had much help the staff at HiG. Simon McCallum, Jason Mackie and Nils Fjeldsø have all been helping us out whenever we were stuck on a problem, as well as given us a lot of good ideas and feedback. As Simon has been our supervisor, we have had weekly meetings as well as almost daily communication on the status of the development. Øyvind Nordstand have been giving us good feedback on the two pitch presentations as well as suggestions and priorities for the Hamar Game Challenge.

8.3 Further work on the project

A future version of the game, would need to be rewritten to match Android 2.1. This OS version have many new functions such as multi touch and better efficiency, which we could use to improve the game play.

In the input handling, using Yaw in a game like this, was quite a brain puzzle, and we did not find a working solution to the problem, this is why the game only support pitch and roll for now. This makes the whole space experience much lower then what it could have been. This would be the top priority to be changed in a future release of the game, to ensure all axis control in a full 3D space where one need it the most.

As of the specific game changes we would like to see, would be to implement a 3rd person perspective, that could be used to get a better view of your surroundings. at the moment it is hard to maneuver around the world.

There would also be a universal tracking system, with a 3D map of all planets, and objects detected while traveling around the galaxies. This would also display the objective location, friendly planets where the player could use a hyper drive system to reach. sThere many more ideas in the Design Document, so for a full list of improvements see it in the Appendix F.

8.4 Beta testing

As the game demo we have made have little content, we have had very little use for external testing. Although, we have had friends and family test the gyro-flight system to get feedback on how they like the system. The general reply was the lack of turning sideways 6.3). This is one of the problems we hope to solve in future releases 8.3.

We have also used feedback on the energy bar. We chose to schedule implementation for micro, macro and fully automatic control, as some testers found it to advanced 6.6.

8.5 Conclusion

As this is our first large scale project we have been working on, we have learned a lot when it comes to project management and the importance of planning. Even though we spent a lot of time planning in the beginning, it was barely enough. Our programming skills have improved a lot, especially when it comes to writing understandable and easy to read code, as well as organizing a larger object oriented project.

We have proven to ourself that we have the knowledge and skill to start out from scratch, knowing nothing about the Android API, and ending up with a good product in a short amount of time. Our learning phase have shown us the importance of a well organized API and good documentation as well as tutorials. When it comes to the 3D part, we have improved our knowledge about OpenGL, not only by using the ES version, but also by always having to think about optimization and the usage of system resources. Because of this, we have had to compare the efficiency of different solutions in different scenarios, as there are still very little information on this elsewhere at this point.

Communication is a vital part of developing software as well as skill, the bachelor thesis has been a good practice for further work within a game of software development company. We learned that cooperation between design, programming, analysis and administration is a vital part

of developing, and that with or without this, projects may shine or be discontinued.

We conclude that with knowledge, devotion and determination with a good base in theory and structure will result in a good product that we are proud to present. The experience from previous courses and assignments have all contributed to our ability to develop and focus on the right assignments at the right times, and that we have the knowledge and confidence to move on to larger scale projects.

Bibliography

[1] McCallum, S. 2010. Game technology lab. http://gtl.hig.no/index.php.

[2] Jabref reference manager. http://jabref.sourceforge.net/. JabRef is an open source bibliography reference manager. The native file format used by JabRef is BibTeX, the standard LaTeX bibliography format.

[3] AFP. 65,000 android phones shipping every day. http://www.google.com/hostednews/afp/article/ALeqM5jtZT_1rdNJFpfU_fwWMiugrx8JMw.

[4] Wikipedia.org. Agile software development. http://en.wikipedia.org/wiki/Agile_software_development.

[5] Google.inc. Android sdk installation. http://developer.android.com/sdk/index.html.

[6] Tez, E. Android ndk tutorial. http://earlence.blogspot.com/2009/07/writing-applications-using-android-ndk.html.

[7] Google.inc. Android ndk installation. http://developer.android.com/sdk/ndk/index.html.

[8] Cygwin.com. http://www.cygwin.com/.

[9] Microsystems, S. Javadoc. http://java.sun.com/j2se/javadoc/.

[10] Gamedev.net, N. P. 2010. Nehe productions website. http://nehe.gamedev.net.

[11] Gamedev.net, N. P. 2010. Nehe productions android ports. http://insanitydesign.com/wp/projects/nehe-android-ports/.

[12] http://developer.android.com. Sensor manager in android api. http://developer.android.com/reference/android/hardware/SensorManager.html.

[13] Wikipedia.org. 2010. Barycentric coordinates. http://en.wikipedia.org/wiki/Barycentric_coordinates_%28mathematics%%29.

[14] SAX-Project. http://www.saxproject.org/.

[15] XML-Pull. http://www.xmlpull.org.

[16] Androidguys.com. Benchmarks android 2.1 vs 2.2. http://www.androidguys.com/2010/05/12/benchmarks-put-android-22-wicked-fastlike-450-faster-21/.

A Work Logs

Kristian

Date	Hours	Description
11.01	2h	Meeting Simon, setting up android development environment
12.01	3h	Brainstorming game ideas, discussing ideas
13.01	3h	Game design document and ideas
14.01	5h	Game design document, Discussing game ideas and features
15.01	4h	Game design document discussing Dev. methods.
16.01	2h	Testing out android GUI programming.
17.01	4h	Writing up on log, testing GUI in android, reading on AI. Started working on the Pre-Report.
18.01	3h	Game Design document, Pre Project report, making group schedule and adding to Google Calendar.
19.01	4h	Game design document, Pre project report.
20.01	2h	Game Design document, pre-report, and sketch of GUI
21.01:	5h	Game Design, Sketch of the space map
22.01:	4h	Alien ship details in game doc.
24.01:	4h	Group meeting, Pre project report
25.01:	5h	Pre project report, rooting htc phone, Power point presentation, Meeting with Simon
26.01:	10h	Pre project report, Power point presentation, feedback on pre report
27.01:	3h	Pitch for the principal from Hamar.
28.01:	5h	Completed the Pre report, Did a read through on the design document and fixed errors so it could pass as an appendix, got the bachelor contract signed and delivered, 1 day in front of schedule
01-13.02:	10h	Android Tutorials from NeHe Android Ports, as well as their old OpenGL tutorials.
14.02:	3d	models, made a few very low poly count models to try out in the game.
15-16.02:	11h	Gyro sensors, trying examples from net, didn't work, Simon made an example and i continued on that.
17.02:	8h	Math Tools, Made Vec3 and Point3 classes with all needed functions.
18.02:	10h	Gyro sensor is working but it is not stable enough. But we can move around and see the Sky box is working
22.02:	11h	World is moving with gyro sensor. H?vard rewrote the code to work better with the quaternions, but it has to be redone once more.
24-28.02	35h	Started with GUI, problems with what kind of Ortho2D commands to use, to set a 2D view on top of the world.
01.03	10h	GUI is working. I have now buttons and loaded images, placed at their correct spots on the screen.

01-07.03:	30h	basic GUI, getting bitmap loading to work properly, as well as making them transparent.
07.03-18.03	45h	energy triangle, tried many different implementations of how the Energy would work best on the screen, and implemented the one that worked best. Let many test it to get feedback.
18-22.03	13h	Working on end report, installed Latex programs, and ported the report from word into TeXnicCenter.
17-18.03:	7h	Working on End Report
19.03:	6h	Adding week reports and logs into Latex
26.03-5.04	10h	Easter Vacation: Made new 3d Model
06.04	8h	Made space Sky box textures, changed boost/break buttons to get rid of line
07.04	5h	Installed NDK, could not get it to work, started working on GUI again
08.04	6h	Worked on Making the GUI without any Magic numbers
09.04	6h	Same as yesterday
12.04	5h	Started on Triggers, continued on GUI
13.04	11h	Implementing Text in the game with OpenGL
14.04	7h	Text in GUI
15.04	4h	Triggers, and figuring out how to implement it, no real progress there...
16-20.04	8h	Made GUI energy work with ship speed, optimizing text in GUI
21-22.04	12h	Making Weapons fire projectiles, and optimizing them.
23.04	10h	Loading new Ship, + projectiles, designing new models.
24-27.04	26h	Preparing for Hamar Game challenge: improving in game text, tracking distance to way points, current speed, and boost level.
27.04-04.05	46h	Writing on End Report.
04.05	8h	Working on NDK implementation
05.05	8h	End Report
06-10.05		Reading for AI exam
11-13.05	5h	End Report and NDK
14.05	8h	End Report, and Fixing Bugs in the Game. Gyro, and Text now use less CPU power.
15.05	5h	End Report
16.05	6h	End Report fixing parts after feedback
17.05	2h	End Report Read through rewriting to propper English
18.05	9h	End Report rewriting with comments from Simon
19.05	11h	End Report and Source Code optimalization

Håvard

Date	Hours	Description
11.01:	2h	Meeting Simon, setting up android developement evironment
12.01:	5h	Developing website, discussing ideas
13.01:	3h	Developing website
14.01:	5h	Discussing game ideas and features, Developing website
15.01:	4h	Writing Gantt Diagram, Installing MS Office, researched developement methods(agile, xp, waterfall)
18.01:	3h	Technology Research, Game Design Document

19.01:	4h	Pre rapport
20.01:	3h	Pre rapport
21.01:	5h	Pre rapport, website(feedback)
22.01:	4h	Pre rapport, security testing of the website
25.01:	5h	Working on the pre rapport, researched debugging on the phone rather than emulator, tested the text-to-speech functions in Android 2.0
26.01:	6h	Powerpoint presentation, statistics, pre project report
27.01:	8h	Powerpoint Presentation, feedback, meeting
28.01:	6h	Gantt chart, pre project rapport, signatures and delivery
01.02:	4h	Android tutorials, Weekly rapport, class diagrams
02.02:	12h	Android tutorials, menus, layout, OpenGL
03.02:	4h	Tutorials
04.02:	3h	Tutorials, texture loading
05.02:	5h	Texture loading optimization
08.02:	13h	Model loading
09.02:	8h	SceneGraph, Guidance Meeting, revised model loader
10.02:	14h	Finished scene graph, remade model loader to .OBJ
12.02:	0h	Sick
13.02:	2h	Sick, rebuilt application
15.02:	4h	Math, Inout, Menus
16.02:	10h	Entities.java, Quaternions, Matrix4ljava, status meeting
17.02:	4h	Quaternions, Menus
18.02:	8h	Level Loading
19.02:	4h	Level loading to visual
22.02:	10h	PNG loading, level loading, XML parsing, entities
23.02:	9h	Quaternions and rotation
24.02:	14h	QUaternions to glRotate and matrices
25.02:	8h	Quaternions and rotations
26.02:	12h	Finished quaternions and implmented it. Loading the XML level with player location and scenegraph setup.
01.03:	10h	Remade the whole Gyro class.
02.03:	9h	Correctly calculating the gyro and implmenting it
03.03:	10h	Edited player to work with the new controls and fixed NaN bug from the Sensors class
04.03:	11h	Level loading and object classes (Ships mainly)
05.03:	9h	Fixing scenegraph for demo
08.03:	8h	Adding objects and polishing for demo
09.03:	6h	Demo and polish + release
10.03:	4h	NDK research
11.03:	9h	NDK research and presentation for master students
12.03:	6h	NDK testing, failed!
15.03:	5h	NDK testing, still fail!
16.03:	10h	finally got NDK to work.
17.03:	7h	speed comparison between SDK and NDK
18.03:	9h	further implementation of NDK source
19.03:	7h	AI research
22.03:	8h	AI Research
23.03:	14h	AI A*

24.03:	12h	A* Path finding
24.03:	8h	AI, states
06.04:	8h	Waypoints and entities
07.04:	4h	Entities and loading
08.04:	5h	Waypoints and models
12.04:	8h	Waypoints (visual), Updated Scenegraph
13.04:	7h	Finished and debugged waypoint system
14.04:	6h	World coordinates to screen research
15.04:	9h	Began tracking system, finished draw to GUI
16.04:	8h	Finished tracking system and resolved eye coordinates
19.04:	6h	Tracking optimization and finalization
20.04:	5h	Tracking multiple targets
21.04:	9h	Code cleanup, pre-release test
22.04:	7h	Demo level
23.04:	5h	Presentation
26.04:	9h	Presentation and market research
27.04:	5h	Hamar Game Challenge
30.04:	5h	Latex research
03.05:	10h	Report
04.05:	10h	Report Waypoints and triggers
05.05:	9h	Report Made an application to compare speeds
06.05:	7h	Report Speed Comarison
07.05:	8h	Report XML Parsing and Scene Graph
11.05:	7h	Report Finished quaternions
12.05:	7h	Report Flight controls
13.05:	6h	Report Developement tools and logs
14.05:	7h	Report week reports
18.05:	5h	Report Summary
19.05:	8h	Report Summary and updated Gantt chart

B Weekly Reports

Week 1

- What was done last week:
  1. Website - 75
  2. Game design document - 20
  3. Pre-Rapport - 25
- Goals for this week:
  1. Work on the Pre-Rapport
  2. Work on the game design document

Week 2

- –What was done last week:
  1. Website - 100
  2. Game design document - 75
  3. Pre-Rapport - 85
- Goals for this week (in descending priority)
  1. Finish the Pre-Rapport
  2. Feedback from Simon
  3. Work on/finish the game design document
- Details for this week:
  1. Game design document
  2. Items
  3. Name the locations, describe them
  4. Name of the races
  5. Details of the enemies
  6. Write down the campaign and objectives
  7. Music and sound effects
  8. Fix the story intro and write more
  9. Sort out the appendix

10. Pre-rapport
11. Footnotes
12. Gantt-chart
13. Write more on: Planning, monitoring reporting, goals, introduction

Week 3

- Done last week:
  1. Pre project rapport - 100
  2. Design document ready to start programming
  3. Gantt chart 100
  4. Wrote and delivered contract
- Goals for this week:
  1. Look through android tutorials
  2. Getting started on the Class Diagrams
  3. Researching OpenGL ES

Week 4

- Done last week:
  1. Android Tutorials
  2. Implemented some basic functions that we found in the tutorials, such as menus and rendering system.
  3. Set up the basic class diagram
- Goals for this week:
  1. More Android tutorials
  2. implement Model Loading, and Gyrosensors

Week 5

- Done last week:
  1. We have implemented a basic model loader.
  2. Gyrosensors is working but not wery well.
  3. Started on Scene Graph
- Goals for this week:

1. Rewrite Gyrosensor to work better
2. Continue on Scene Graph
3. Vector and Point classes for the Scene Graph and more.

Week 6

- Done last week:
  1. Rewrote .OBJ loader
  2. Finished Scene Graph

Week 7

- Done last week:
  1. Math classes, Vector Point etc.
  2. Quaternions fully implemented
  3. Level Loading
  4. Entities and other basic classes.
- Goals for this week:
  1. Quaternions class working

Week 8

- Done last week:
  1. Image loading
  2. level loading
  3. XML parsing
  4. Matrices in Scene Graph
  5. expanded Quaternions

Week 9

- Done last week:
  1. Rewritten gyro class
  2. Fixed player class
  3. fixed Not a Number bug that came from the sensor.
  4. Set up Scene Graph for Demo.

5. Loading of Object classes.

Week 10

- Done last week:
  1. Added a Demo world
  2. NDK research
  3. Presentation for the Master and other bachelor students.
- Goals for this week:
  1. Research the new NDK revision 3.
  2. Continue the work on GUI
  3. Start working onn the end report

Week 11

- Done last week:
  1. performed a speed comparison of NDK vs SDK.
  2. Further tutorial implementation on NDK source code.
  3. We tarted researching Artificial Intelligence.
- Goals for this week:
  1. Make low poly 3D models for use in the game.
  2. Get the models working in the game world.

Week 12

- Done last week:
  1. Research on A* and AI states.
  2. 3 new Ship models was made and all gets loaded
  3. Still problems with materials and textures.
- Goals for this week:
  1. ENJOY VACATION!

Week 13

- Done last week:
  1. VACATION!

2. A new 3D model, Normandy ship.

Week 14

- Done last week:
  1. Waypoints system, and loading of the waypoints now working in game.
- Goals for this week:
  1. Implement Waypoints.
  2. Start working on presentation for Hamar Game Challenge.

Week 15

- Done last week:
  1. Visual effect of the waypoints now working, scene graph also improved.
  2. Started working on tracking system and done research on how this could be implemented.
  3. Presentation for Hamar Game Challenge.
- Goals for this week:
  1. Prepare a demo video and a powerpoint presentation for Hamar Game Challenge
  2. improve Collision detection
  3. Triggers
  4. Tracking of way points

Week 16

- Done last week:
  1. We went to Hamar with our project and did a fairly good presentation of our game idea. Unfortunately we lost, mostly due to our groups where so bad balanced with a real group leader, designers, animaters, programmers etc. Allthough they did say they liked all the game ideas very well.
- Goals for this week:
  1. Continue with the end report.
  2. Comment code that has no comments.

Week 17 - 20

These weeks we have been working on the end report, writing javadoc comments on our code base. Also, the gyro class has been redone, again, due to deprication of the Android API.

C Progress meetings with supervisor

26/01-10 Meeting With Simon

- Creativity
  1. Find link between elements
  2. Inspiration from sci-fi
  3. Think outside the box
- Include basic sales techniques in the game
- TTS vs. Voice Actors
- Try to base the game on TTS in the beginning
- Find out on the performance requirements of the TTS module
- Keep a consistent TTS language for different races, might be used in developing the community further
- It is possible to get voice actors for a fair price
- This might be an option for further development of the game.
- Positional audio Could be implemented in the game, needs research. Imagine turning your phone around the room, and you can hear confirmation sounds when you point in the correct direction could be used to let the player know in what direction the incoming fire is originating from.

02/02-10 Meeting With Simon

- Talk to Phone recognition, speak with AI
- SMS / Phone calls from in game Your ship / base / faction / clan member is under attack etc. Request challenge objective is in close proximity to you, send warning to player
- face book holds top scores, best equipped user etc. Send challenges from face book etc, to challenge vs mini games.
- Multi player area / battlegrounds When the player is "off line" aka app is not active, and AI is in control of the ship, The ship is placed in a stack of inactive players, and can be used as enemies and objectives for quests/missions.
- If a mission is made, to take down that player, he will receive a sms about being targeted. if there is urgent threat to the ship, the AI can make a phone call to the player to get the attention needed. Then the player log on, and he will have to give some fast commands, to

flee/fight the opponent.

- Loose /win areas, safe zones, controlling planets give credits faction standing with different planets
- In game credit system, works on phone and face book buy and upgrades on XP / stats for a given time, micro payments.
- upgrades to be bought for sms / phone call warnings
- missions, not campaign, or just a main quest line to complete to visit new planets.
- Players in a clan can buy planets, and have exceptionally good boosts and credit gain.

09/02-10 Meeting With Simon

- thesis on Bounding boxes vs bounding spheres time spent on different phones
- How and where to save game data.
- Make a beta form on web page -require login etc make character.
- Lime-form for beta feedback

13/04-10 Meeting With Simon

- Auto Mode AI controlled Ship control all energy use. If you fly without taking damage or shooting it will turn more energy to engines. And if taking rapid damage, it will boost shields
- Simple Model 7-8 points that the shield can be placed on, center top sides etc. and ambient bars
- Advanced mode Free movement + Slide bars for Ambient Energy
- Where to place the energy triangle? ideas:
  1. Top of triangle pointing upwards.
  2. Sideway with point in.
  3. top pointing to bottom of screen,
  4. Not show energy bar at all, except for a button to enter energy mode, where the triangle shows on the whole screen and makes it easier to set. This would enable us to just use a normal energy bar to display total energy.

20.04-10 LaTeX guidance by Simon

- Showed how to add images to the latex system, and how appendixes and references are to be handled.
- Installed JabRef to handle the references.
- Talked about how to get the correct layout, using a premade template designed for Master

Thesis's.

Meeting with Principal 1

- Notes:
  1. Get better at speaking to the audience.
  2. The idea was good.

Meeting with Principal 2

- Notes:
  1. We had more data about market research.
  2. Progress is going well.

D Pre Class Diagram

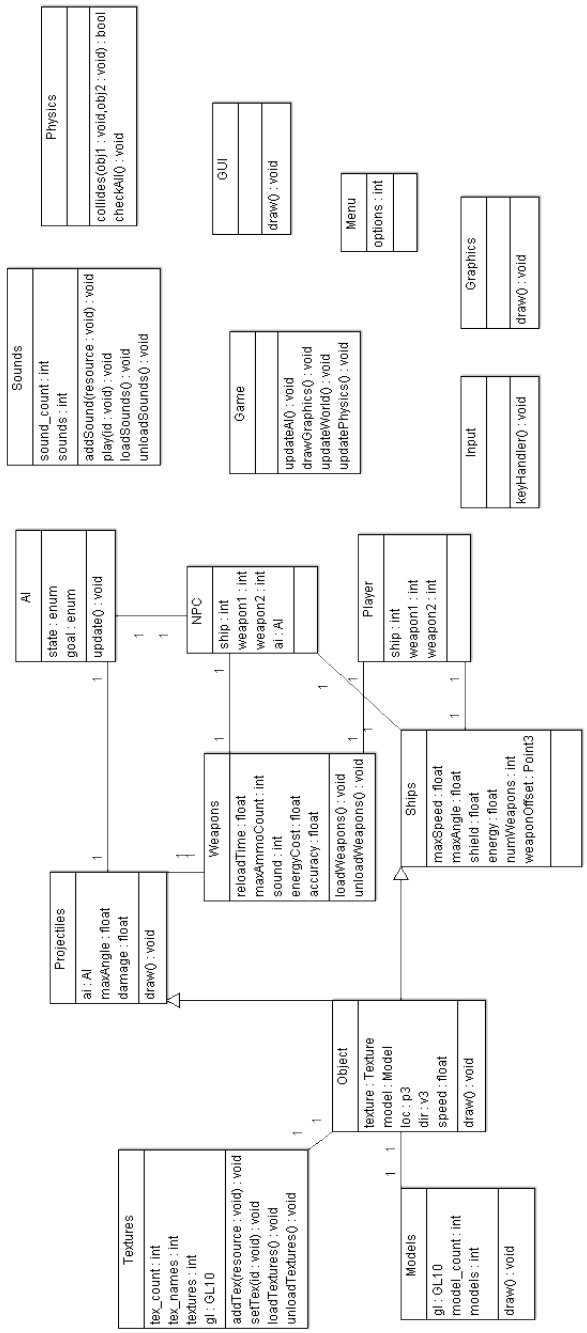


Figure 8: Pre Class Diagram from the inital stage of the implementation

E Pre-Project Report

28.01.2010

Pre-Project Report

Alien Invasion Game

For Android OS. Phones

Håvard Kindem

Kristian Nordhaug

Table of Contents

1.0 Objectives

The main objective is to develop software of high quality that will attract a lot of new users and strengthen the principals marketing position within the mobile phone gaming industry.

Game

The game takes place in different parts of space. It is late fall 2012 and it is your first day at the COAT (Center of Alien Technology) training facility, orbiting earth. No one knows that a gigantic invasion is impending. COAT has been exploring space since 1998, when they discovered a way to break the time-space continuum.

As the aliens manage to break through, it is up to the player whether the population of earth will survive the day!

Android

Android is an operating system for cell phones designed and made by Google.inc, who bought the small company Android.inc in 2005.

In 2007 there was a consortium made by many well known companies, formed as the Open Handset Alliance (OHA). Some of the most famous names where Google, Intel, LG, Marvell Tech Group, Samsung, Motorola, Nvidia, Sony Ericsson and Texas Instruments.

Google and OHA made the whole Android, with all source code and SDK, open and free for everyone to use under an Apache License. This makes it a brilliant platform to make software for, and is the reason why our group settled for it.

Performance

The performance objective of the project is to make a game that will be robust and bug free and run with its best graphics without lagging. It must also meet the requirements for android applications and support the standards for correctly installing, starting, running, pausing and exiting the application.

The game should also be designed to enable future patches with more content can be added, without rewriting the code base of the application.

Learning

Our goal is to gain knowledge of the Android system, and mobile phones in general. Android applications are written in Java, which we have some experience with, but there are so many differences from a regular Java program to an Android program that we are starting close to scratch. Regardless we will be working hard to complete the development process in the best way possible.

Target Audience

We are aiming for players from 10 to 30 years of age. The ESRB rating we should be getting is E[1], as we have no blood or gore of any kind.

2.0 Scope

Development process

Our task is to go through the whole software development process, from planning to release of an Android phone game. We will focus on using the same methodologies as the gaming industry, to secure a stable and high quality product, with good documentation and reusability.

Phases:

Planning

Design document are written, and we create the release schedule. The idea is to make frequent small releases as the project is divided into iterations. We will also schedule meetings, as iteration planning starts each iteration.

Designing

The focus is on simplicity, we will try our best to keep everything generalized, understandable and explainable. It is also important not to try implementing models before they are scheduled, as it quickly can end up as a time drain.

While designing we will be conducting spike solutions. This means we will try implementing the most risky parts as separate programs, before writing code right into our project.

Coding

As we are only two people, we will integrate often, and we will both have the same testing environment set up, as we use emulators.

Testing

Testing will be performed on each module to make sure it is safe to go on in the project. We will also have testing when reaching each milestone (GUI,

Modularization

We will split up the development into multiple modules to have better control over the programming process. These will be as follows, in no particular order:

- Data Structures
- Game Menus
- GUI
- The Game World
- Artificial Intelligence
- Input Handling
- Physics
- Model and Image loading
- Top scores
- Campaign

Refinements

As we have a very limited amount of time for the development of our game, there are some things that might suffer. As our goal is to make a fully functioning game, we will focus as much as possible on this. We have discussed that we might add upgrades for our ship, make some extra game modes and getting this all integrated into Facebook to get people to compare for top scores, try challenges etc.

This however, is a very time consuming job, if everything was to be implemented. Therefore, we have decided that the development of the mentioned modules will have to be considered when the other activities are completed.

Boundaries

The delivery date for the project is set at 20.05.10.

Developers have reported that making software for the different phones creates allot of issues concerning the different aspect ratios on the cell phones. For this project we will give it our best to make the game work for all phones, but the only hardware we have for testing, is the HTC Hero. This phone is running Android 1.5, but HTC has promised Android 2.1 for the phone in the early spring, so we will aim for using at least the Android 2.0 SDK for our game.

3.0 Project organization

Responsibilities

While both of us will have to join in on basically everything, we have chosen to set up four main responsibilities. These will act more like guidelines and will set the group member's focus on specific tasks, rather than limit the work in certain areas.

- Group Leader Kristian Nordhaug
- Lead Game design Kristian Nordhaug
- Lead Programmer Håvard Kindem
- Webmaster Håvard Kindem

Meetings and Working Routines

Status Meeting:
Monday: 09:00 – 09:30

Project Lab Work:
Monday: 09:00 – 14.30
Tuesday: 09:00 – 15:30
Wednesday: 12:00 – 13:30
Thursday: 09:30 -15:00
Friday: 09:00 – 12:30

Other roles

Supervisor – Simon McCallum

Principal –Øyvind Nordstand (Kunnskapsparken In Hamar)

4.0 Planning, monitoring and reporting

Planning

We will establish and follow our game design document during the whole process. A Gantt diagram will also be finished by the end of our learning phase, with planned schedule and progress of the project. As we are going to use the Extreme Programming, we also have the possibility to modify the plan as we go if we feel something has to be done in a different way.

Monitoring and reporting

We will have status meetings every Monday to see how much of the last week's targets we have met and review the targets for the coming week. We will write a weekly status rapport, which will contain the topics discussed, the achieved completion and any corrective actions required.

As well as our weekly rapports, we will log every working hour on the project with all the things we have done. This to have the possibility to go back and see where we left off, what we used our time on etc.

5.0 Quality Assurance

Development Environment

We will be working with Eclipse for Java, with Android and SVN plug-in installed. This is the default method used for developing Android applications. With the Android Software Development Kit, we are able to emulate an Android system on our computer. This makes the testing and debugging of the application much easier, as we can do it on the fly, not having to update it on the phone all the time.

As well as testing and debugging on the computer, we will do the larger scale testing on the HTC Hero, as we have to do so to see if the gyro sensors are working correctly and know if the GUI is suitable for the smaller screens on mobile devices.

Standards and source code

We will be following the standards of Android OS build 2.0, but make sure the code can run on Android 1.5 as well, as many of the Android phones have not yet got their update. The commenting will be done by the code's author, prior to coding. This way, we will minimize the possibility of having chunks of code that is not documented and poorly commented.

Documentation

For documentation of the source code we will use JavaDoc, as this is already automatically generated through Eclipse. It will be written before the code itself, as explained above. We will not release the documentation on our website, as it might be a concept worth developing further.

Configuration management

Our group will use the Tortoise Subversion tool for configuration management, as the IT-Services at Høgskolen I Gjøvik offers these for no extra cost. The documentation however, will be automatically maintained by the JavaDoc application, as it is all written in the code, this saves us both time and effort, trying to work out the documentation another way.

Risk Analysis

Scope

- Project not completed in time
  - If somehow the workload gets too hard or we should experience immense complications on some parts which would totally obscure our timeline.
  - Risk evaluation: Low
  - Mitigation: reduce scope or support from supervisor
- Driver problems
  - As the layout of the Android phones are so different, we could have problems getting the application to fully function on different phones, as we only have the HTC Hero to test with.
  - Risk evaluation: High
  - Mitigation: release for HTC Hero only
- Inappropriate system requirements
  - If the application can't run smooth enough on the mobile device and can't stop lagging, we have to reconsider the graphics and other resource demanding operations.
  - Risk evaluation: Medium
  - Mitigation: As we are programming the HTC Hero (Generation 2), we will not lower the graphics to support Generation 1 Android phones, unless we can make an option to choose quality on each device.

Schedule

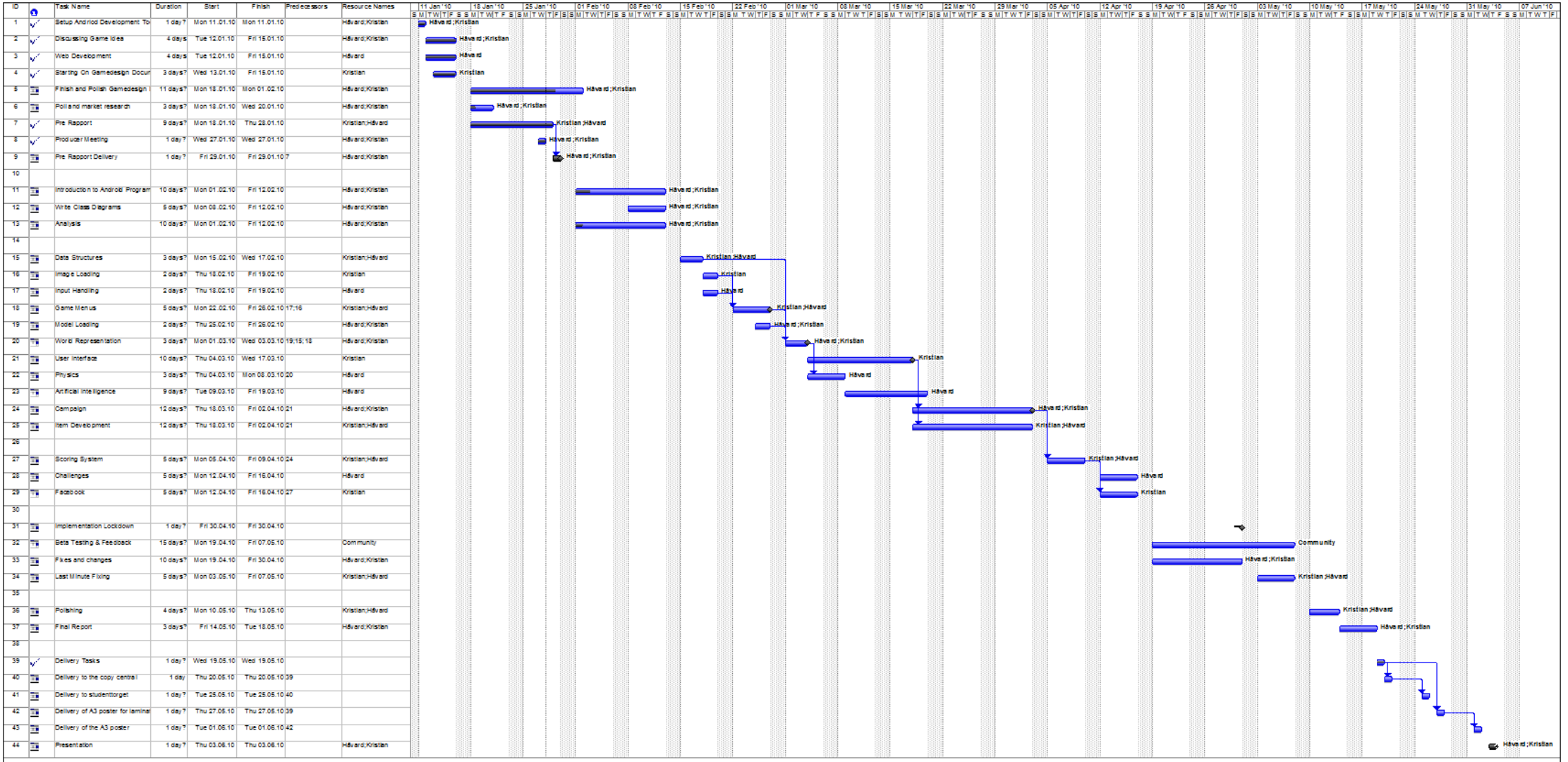
The planned development process for the application as described in appendix 1 has some uncertainties. Although we know what we need to do to finish the project, we are learning a new programming language as well as developing with it. Due to these two factors, there are some uncertainties when it comes to the time we will use on each module. Because of this, the Gantt chart of tasks should be considered a rough estimate. We are using the Extreme Programming development method, so we have the time and opportunity to get more accurate estimates as we go.

7.0 Footnotes

1. Descriptions at http://www.esrb.org/ratings/ratings_guide.jsp
2. http://www.androidguys.com/2010/01/04/the-average-android-user/

8.0 Appendix

1. Gantt Chart
2. Game Design Document



ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	Setup Andriod Development To	1 day?	Mon 11.01.10	Mon 11.01.10		Håvard;Kristian
2	Discussing Game Idea	4 days	Tue 12.01.10	Fri 15.01.10		Håvard;Kristian
3	Web Development	4 days	Tue 12.01.10	Fri 15.01.10		Håvard
4	Starting On Gamedesign Docur	3 days?	Wed 13.01.10	Fri 15.01.10		Kristian
5	Finish and Polish Gamedesign I	11 days?	Mon 18.01.10	Mon 01.02.10		Håvard;Kristian
6	Poll and market research	3 days?	Mon 18.01.10	Wed 20.01.10		Håvard;Kristian
7	Pre Rapport	9 days?	Mon 18.01.10	Thu 28.01.10		Kristian;Håvard
8	Producer Meeting	1 day?	Wed 27.01.10	Wed 27.01.10		Håvard;Kristian
9	Pre Rapport Delivery	1 day?	Fri 29.01.10	Fri 29.01.10	7	Håvard;Kristian
10						
11	Introduction to Android Program	10 days?	Mon 01.02.10	Fri 12.02.10		Håvard;Kristian
12	Write Class Diagrams	5 days?	Mon 08.02.10	Fri 12.02.10		Håvard;Kristian
13	Analysis	10 days?	Mon 01.02.10	Fri 12.02.10		Håvard;Kristian
14						
15	Data Structures	3 days?	Mon 15.02.10	Wed 17.02.10		Kristian;Håvard
16	Image Loading	2 days?	Thu 18.02.10	Fri 19.02.10		Kristian
17	Input Handling	2 days?	Thu 18.02.10	Fri 19.02.10		Håvard
18	Game Menus	5 days?	Mon 22.02.10	Fri 26.02.10	17;16	Kristian;Håvard
19	Model Loading	2 days?	Thu 25.02.10	Fri 26.02.10		Håvard;Kristian
20	World Representation	3 days?	Mon 01.03.10	Wed 03.03.10	19;15;18	Håvard;Kristian
21	User Interface	10 days?	Thu 04.03.10	Wed 17.03.10		Kristian
22	Physics	3 days?	Thu 04.03.10	Mon 08.03.10	20	Håvard
23	Artificial Intelligence	9 days?	Tue 09.03.10	Fri 19.03.10		Håvard
24	Campaign	12 days?	Thu 18.03.10	Fri 02.04.10	21	Håvard;Kristian
25	Item Development	12 days?	Thu 18.03.10	Fri 02.04.10	21	Kristian;Håvard
26						
27	Scoring System	5 days?	Mon 05.04.10	Fri 09.04.10	24	Kristian;Håvard
28	Challenges	5 days?	Mon 12.04.10	Fri 16.04.10		Håvard
29	Facebook	5 days?	Mon 12.04.10	Fri 16.04.10	27	Kristian
30						
31	Implementation Lockdown	1 day?	Fri 30.04.10	Fri 30.04.10		
32	Beta Testing & Feedback	15 days?	Mon 19.04.10	Fri 07.05.10		Community
33	Fixes and changes	10 days?	Mon 19.04.10	Fri 30.04.10		Håvard;Kristian
34	Last Minute Fixing	5 days?	Mon 03.05.10	Fri 07.05.10		Kristian;Håvard
35						
36	Polishing	4 days?	Mon 10.05.10	Thu 13.05.10		Kristian;Håvard
37	Final Report	3 days?	Fri 14.05.10	Tue 18.05.10		Håvard;Kristian
38						
39	Delivery Tasks	1 day?	Wed 19.05.10	Wed 19.05.10		
40	Delivery to the copy central	1 day	Thu 20.05.10	Thu 20.05.10	39	
41	Delivery to studentorget	1 day?	Tue 25.05.10	Tue 25.05.10	40	
42	Delivery of A3 poster for lamina	1 day?	Thu 27.05.10	Thu 27.05.10	39	
43	Delivery of the A3 poster	1 day?	Tue 01.06.10	Tue 01.06.10	42	
44	Presentation	1 day?	Thu 03.06.10	Thu 03.06.10		Håvard;Kristian

F Game Design Document

Appendix #2

Design Document for:

Alien Destruction

The Phrophesy

Written by
Kristian Nordhaug
Håvard Kindem

Version # 0.2

Tuesday, May 04, 2010

Table of Contents

Design History

In this document we want to give the reader our ideas for the game. We will describe our intentions with the game, and how we would like to see it completed.

Version 0.1

This is the initial stage after we settled on our game idea, and we did a quick run trough of the document to add as much ideas as possible.

Game Overview

Philosophy

Philosophical point #1

In this game we are trying to learn the new Android OS, which we have never programmed with before. It is a whole new way of thinking in how to make a program, so we dare not expect our game to be revolutionary in any way, but rather be a fun game testing out some of the capabilities of the Android system that we find most interesting.

Common Questions

What is the game?

This is a sci-fi themed game, set outside our atmosphere in space, at the top secret united nation alien counter attack base, that protects earth from total destruction.

The game will describe all the secret technologies that exist today, that none of us normal humans know about. Also that we do regularly have contact with other alien races, that are kept hidden from us by the governments around the globe.

We do have spaceships with lasers, and which are capable of traveling light speed, and even faster…

Why create this game?

The idea of making this a space game, is to have as much open space as possible to move around, and make best use of the gyro stabilization, and the compass inside the Android phones. This makes it possible to move the phone around in front of you, and it will detect all movement done. This we will use to move the camera around in outer space, such that you can aim your weapons at the right targets.

What do I control?

The player will be in charge of one of COAT's latest edition space fighter ships. The ship will start with a laser and rocket launcher that are controlled by the thumbs at the bottom corners of the android phone. It will also have a throttle boost for extra speed and a break button.

The player will have to create a profile for his character, with a unique name that will be used for logging into the Android application as well as the face book profile, and the system on our webpage that will handle the database of each character, and their gained score, items and progress.

What is the main focus?

The main focus on the game is to explore the universe as an adventure game, seeking distant planets, millions of light-years from home, and to seek and destroy the alien race that sent our planet Earth to oblivion. Along the way there will be encounters with other alien races that can provide aid, new technologies, and information on how to find the right planet.

What's different?

On the planets that are discovered in the universe the player will encounter Abducted Humans from different times on earth, we would really like to find Elvis, Napoleon and Einstein and Cleopatra!

The game will be controlled by the gyro sensors in the phone.

Feature Set

General Features

Big Universe
3D graphics
Interface with intelligent tracking system for other encounters in space.

Multiplayer Features

The game itself will not be multiplayer, except for the fact that everyone will be competing against each other's top score, and achievement points, in an endless struggle to unlock every aspect of the game, gather most weapons and learn of most ancient technologies from the different planets.

Game play

Gyro sensors
Meet famous humans and aliens
Experience new technologies in cell phones.

The Game World

Overview

The game will change as you progress through the different planets in different star systems. This will be shown with different background of the space, with star images, and closer images of the different galaxies. For example, at earth you may see orisons belt, Pegasus and the other star images. But when travelling to Andromeda galaxy, you can see the Milky Way in the background. The planets will also chance in color and size, their suns glowing with different colors.

World Feature #1

When communicating with the other alien races there is already a unified system that translates all communication, this is implemented in the space craft, and will seem naturally for the player. The translated words will be displayed in the status message field at the bottom of the screen. There will also show an image and name with the one you talk to, so the player easy will recognize them. They will however hear the sound as the aliens normally talk, to give the real impression that they are in fact, aliens.

The Physical World

Overview

There will be a restricted area around each planet you travel to, so the player stays in orbit of the planet, and don't get lost along the way. When hyper driving close to the planet of destination, the player will gain manual control of the space craft, and have to fly into range for communication and combat.

When in manual control, the player has a full 360º angle of the world and can shoot everywhere.

Key Locations

Planets systems surrounding famous stars, like Polaris, Betelgeuse, Alpha Centuri, Nebula systems, Corellia, Andromeda,) Black holes, and Supernovas.

Travel

The Spacecraft will travel around in manual mode with high tech rocket booster, and when traveling to other system, it will initiate hyper drive.

Scale

The scale of the world or the restricted area will not be that huge, it is a small device, with small screen and it would be troublesome to have too much area to travel around, but it will be big enough to give the impression that you are floating around a planet.

Objects

New weapons, space ships and other technologies.
See the "Objects Appendix" for a list of all the objects found in the world.

Weather

There will be radiation from stars and supernovas, which makes the missions around those planets extremely hard. The radiation will burn the ships energy shield fast, and there will be timers for how fast the player must complete the missions.

Day and Night

The stars will shine upon their planets correctly, so the player will have a hard time maneuvering on the backsides of the planets. It will be the player's job to stay out of trouble.

Time

Apart from the missions where time is essential for survival
The time in the game is being counted, and loaded up with the profile. There will be high score tables for best time runs of the campaign, but the main top score table will not take into account for the time. As we want the player to explore the universe as much as possible.

Rendering System

Overview

The game will be rendered on Android OS. These cell phones all support OpenGL, which we will use to make our 3D graphics.

2D/3D Rendering

Unless we find some useful engines, we will create our own. Most likely we find some support in the Google.inc made program Google Sky, which has the system of maneuvering around with the phone, exactly like we intend to use it.

Camera

Overview

Camera will work like the phone is a window into the other world. When you look at the phone your holding in front of you, you will look into the world at the same angle as the phone. If you lift it up and it's pointing towards the sky, you will then look in that direction in the game. This is how the player will be able to find its enemy targets.

Camera Detail #1

There are also situations that the player is unavailable to move around with the phone like a madman. So the same features will be added to the phones that have a trackball, which will serve the same purpose.
Some players might end up only using this feature, as moving the phone around would be too much of a hassle, but we aim to use the gyro sensors as the main way of targeting, as this is a completely new feature.

Game Engine

Overview

The game engine will be built as an activity, as all programs in an android phone. The general technology, supported by android is OpenGL ES 1.0 (equivalent to OpenGL 1.3).

Environment

We will place a skybox at an infinite distance from the player, using images for the location. From this, we will calculate where the different light sources will have to be placed.

Collision Detection

As the game is placed in space, we are going to do the collision detection with bounding boxes. This is because in a large 3d environment, we won't have many collisions, except for projectile impact, so the detail of the pixel perfect collision is not needed and will only waste further system resources.

Lighting Models

Overview

We will use an ambient light, enough to make everything visible. Other than that, we will set up multiple diffuse lights for the nearby stars. As OpenGL ES 1.0 doesn't support GLSL shaders, we will keep the lighting effects basic. We will however, set up a glow effect on the appropriate projectiles.

Ambient Light

Due to the darkness of space, we break reality to let the player actually see what is going on, hence the ambient light. This effect will be used on the space ships and other object, not producing its own light.

Diffused Light

The stars will be getting this effect, casting it on the other objects in the game world. If the system resources can handle it, we will also add some diffused lighting to the projectiles.

Glow

The glow effect will be used on the projectiles, as well as other objects, capable of producing its own light.

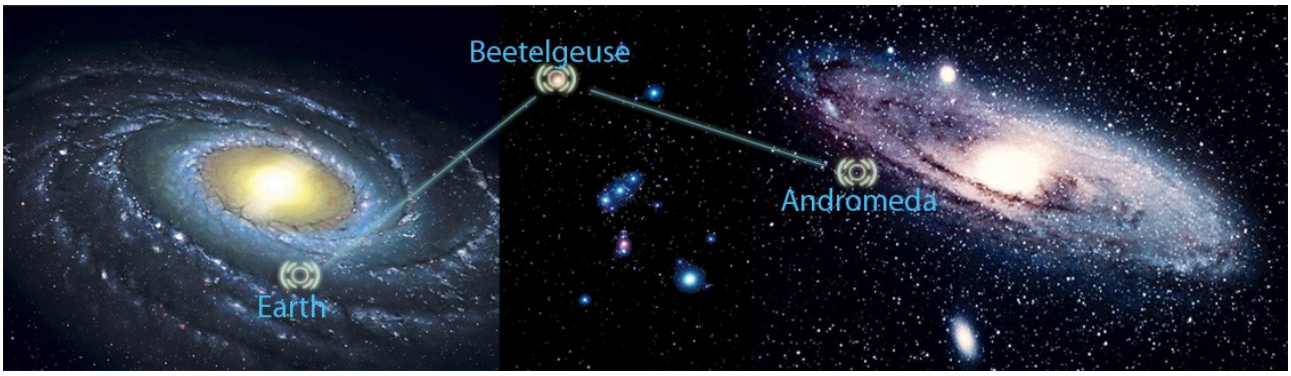
The World Layout

Overview

The world locations will differ from each other in mission objectives, enemies and surroundings. The game world is basically a vast empty space, crawling with enemies. You might also encounter other alien creations, such as ship factories, artificial planets etc.

Locations

As the story takes us to different locations in space, we want to replicate those locations as close as possible. Some places will have red, giant suns; some might have what is left of a supernova, while others might have what is known as a white dwarf.



Images are from these respected sites in the following order:
http://www.phys.ncku.edu.tw/~astrolab/mirrors/apod_e/image/0501/milkyway_garlick.jpg
http://startheory.files.wordpress.com/2009/08/orion1.jpg
http://www.definity-systems.net/~apw/astro/images/andromeda_big.gif

Game Characters

Overview

The player will need a username and password that also will be linked with Facebook and the webpage. That account will be linked with a player
They will only be able to play one character throughout the whole game.

Game Characters Detail#1

The character they start will have the story of being the most promising space fighter of the decade; right out of the academy he/she is placed with a top of the line space craft, to be the defense of earth.

Creating a Character

After logging in with username and password, that matches the webpage and Facebook application. There will be a character profile setup, where you choose a name, gender, age and a profile picture. The pictures can either be taken at the moment with the camera, or the user can browse his cell phone, or find one on his Facebook profile.

Creatures and behavior / AI

The enemies will be alien space ships, in different sizes and colors. If the player has to communicate with anyone, it will be displayed at the bottom of his screen, in the status message window. It will display an image / 3d animation of whomever he is speaking with, along with the name and translated speech.

Ship Details#1

Hit points
Motion detection range /radar range
LOS
Patrolling
Detect movement
Weapon max/min range
Weapon accuracy in % on min/max
Damage inflicted
Boost Power
Armor/Shield Ratings
Combat Actions
Target system

Drone description:
Drone guard will be found outside Polaris, it is a weak enemy, designed for the first stage of learning the game. Only carry laser weapons.

Assault Shuttle
Normal patrol unit

Drop Ship
Slow ship with high HP but low combat power

Personnel Shuttle
Fast ship with low HP and low combat power

Troop Shuttle
Medium sized ship, equal to a frigate, but without the combat power, but good armor

Fighter
Best close combat fighter, best laser weapons

Frigate
Larger then a Fighter, but have good combat and armor capabilities

Destroyer
Destroyer is the most common combat ship for the missions, larger then any of the previous, but still no match against Battleships, Cruiser or Dreadnaught.

Cruiser
Cruiser will be the standard support ship that will follow the Destroyer in most cases.

Battleship
Large ships, dedicated to combat

Battle cruiser
Fast battle ship, with much less armor, but same quality weapons

Dreadnaught
Largest ship in all of the alien fleets.

Events:

- If the units detect the player it will start to approach the enemy until it has the best possible fighting position. It will fire its weapons to maximize its damage output.
- When it's firing weapons, it will continue to move towards the enemy, and try avoid taking hits.
- If the player moves to avoid the unit, it will do its utmost to keep the player in its LOS, and chase the player for a given time.
- The unit will, as the player, die (by explosion) when his energy shield reaches zero, at the first shot after that.
- The unit will drop mission specific resources if it is a part of the current object.

Details #2
Ship AI:

- Female voice
- Takes control of ship if too close to a planet, or travelling to far away from mission area.
- Is meant to be in control of all the aims on the screen, AI upgrades will provide more aim accuracy.
- Gives warning with red blinking on the screen edges when taking damage and sound
- Tracks the current mission with a green arrow on screen
- Warning when energy level reaches critical low – 10%.

User Interface

Overview

- Two weapon buttons (Lower left and right), and a boost and a break button
- Status message window that will come up from the bottom between the buttons.
- Aim in the center for weapons.
- Ship computer which opens the status message window, and also controls the tracking.
- Tracking will be small aims with a directional arrow pointing to the enemies.
- Green arrow tracking current mission and/or object.

User Interface Detail #1

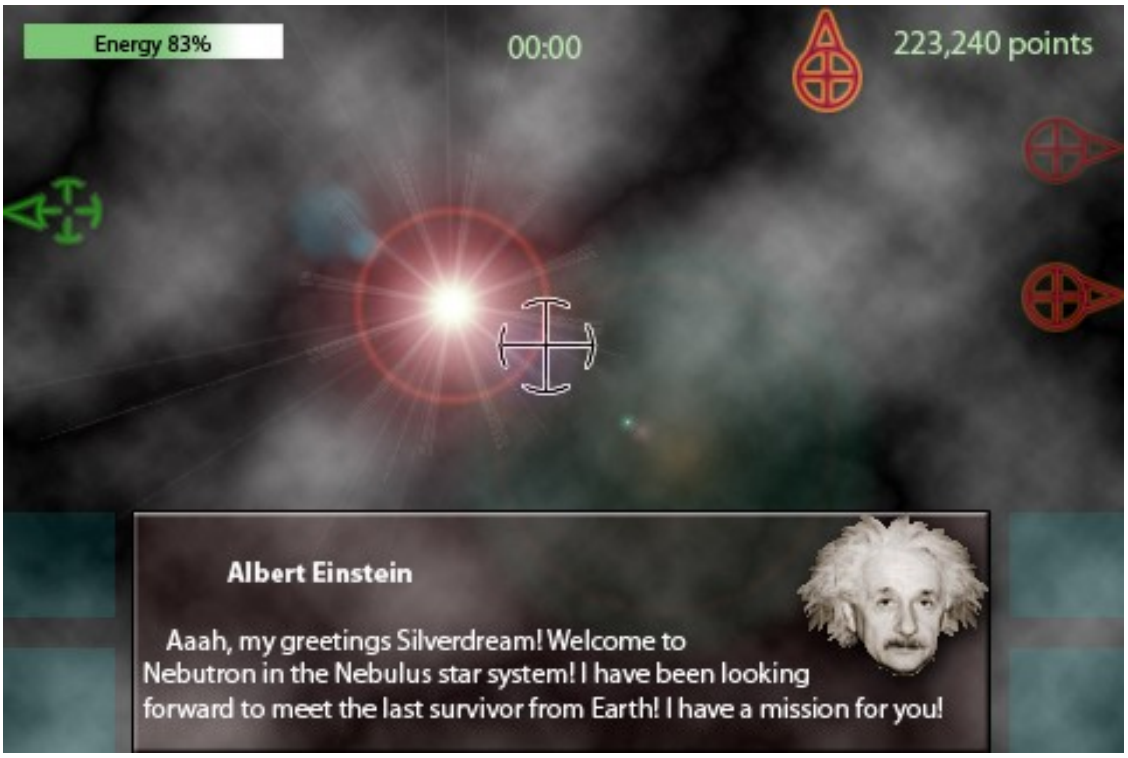
On the top line of the screen, we will display the energy shield level, placed at left side. Middle will be reserved for the timer that will be enabled in some missions. At the right side there will be a high score counter.

User Interface Detail #2

There will also be the tracking system of enemies (red arrows in picture). These are floating aims, with varying color from yellow to red and blinking danger, as to how close the enemies are. This will be the main system for the player to find where to shoot and locate the enemies. It will also help him to choose which one to aim for first. More red and blinking will be more dangerous opponents.

User Interface Detail #3

The real aim will be center placed at the beginning of the game, and can't be moved until later, when the player gains some onboard ship computer upgrades. Then it can move around with a bit of AI to aid the player.



Weapons

Overview

- Lasers
- Heat seeking missiles
- Radar triggered missiles
- Ultimate's
    - Heat Seeking Missile
    - Gravity missile
    - Radiation bomb
    - Supernova bomb

Laser

The laser will be your primary weapon, as it uses energy from the suns it will have an endless amount of ammunition, although: it will have a recharge time between each shot that can be decreased by upgrading your laser.

Radar triggered missiles

Radar triggered missiles will be the normal secondary weapon button skill. This will have a cooldown of 10-20 seconds. The actual time will have to be beta tested.

Ultimate's

Heat missiles will be a close combat "ultimate" that will aid the player when all hope seems lost.
This item can be unlocked or bought for real money trough the webpage. The same goes for gravity bomb, supernova weapon, and radiation bomb.

Gravity missile, when exploded, will drag all enemies in the area to a single location, enabling for a quick heat or radar missile to take out all enemies at once.

Radiation bomb will work as an AOE over time, and can be used to "clear an area". Everyone entering the radiation gas will get energy drained by the load on the ships shields.

Supernova bomb will explode the entire current solar system, and can only be used at the end of the game. It will only be unlocked trough the campaign.

Musical Scores and Sound Effects

Overview

The Sound effects need a special coding.
JET coding

Red Book Audio

If you are using Red Book then describe what your plan is here. If not, what are you using?

3D Sound

Talk about what sort of sound APIs you are going to use or not use as the case may be.

Sound Design

Take a shot at what you are going to do for sound design at this early stage. Hey, good to let your reader know what you are thinking.

Single-Player Game

Overview

Describe the single-player game experience in a few sentences.

Here is a breakdown of the key components of the single player game.

Single Player Game Detail #1

Aliens traded humans for technology and galactic information.

The good aliens warned the humans, that the galactic war would reach their solar system at 21.Dec.2012. For that reason, all governments have struggled to develop technology for that day. But the secret itself has been held for thousands of years.

Single Player Game Detail #2

Story

The game is to go through the campaign, starting on the space defense base orbiting Earth.
The player is given control of the newest spacecraft made, the first ever built with a hyper drive engine, capable of entering hyperspace.

The camera will zoom closer to the base, as the intro text is written on screen:

COAT(Center for Alien Technology) Earth date: 21. Dec. 2012

Welcome to COAT, lieutenant! You are late!!!
Today's training has already started!
Get back to your squadron ASAP, the practice drones are incoming.
Your ship AI will handle the communication from here.
Over and out!

On the first day out of the academy, the introduction mission for the game will lead the new pilot into his first real fight ever. The galactic enemy race Graqu attacks, and although the training, the player will not succeed in the defense. Eventually the earth is destroyed, and everyone except the young pilot, will die. He only survives as the ship drift off in the wrong direction of the fight due to his problems handling the ship (or as it might turn out, the ship did not want to get destroyed).

As the intro ends, the Aliens will attack and destroy Earth, and the campaign to find them begins.
The path leads the player trough the following star systems:

- Andromeda
    - Home planet of Elidyans
        - Planet is green, with a yellow sun
- Polaris
    - Home planet of Neruflux
        - Planet is light blue, with close to a white star.

- Betelgeuse
    - Planet is green/brown, with a green star
    - Home planet of Birantu
    - (Idea #1 may not be included) Here the player will find another famous human, Elvis! He is utterly depressed since he has no way to style his hair! On this planet the aliens do not have any hair, and nether any hair gel. His quest for the player is to locate some hair gel so he finally can look cool on stage. For the help, the player is free to choose one ship upgrade item from Elvis' space ship garage.
- Nebula system
    - Home planet of Ciodonts
        - Planet is red/purple with a lot of purple gas around, and many medium sized suns.
    - This is where Albert Einstein has set up his huge science laboratory. After hearing about what has happened to earth, he give the player a quest to obtain a last item for the ultimate weapon that he has been working on. This is the Supernova weapon, but it will not be activated before the end of the game, since it has to charge up its energy.
- Corellia
    - Home planet of Kezzyl
    - (Idea #2 may not be included) In Corellia the player reaches a planet where he meets Queen Cleopatra of Egypt. She was abducted by the aliens two thousand years ago to save her in eternal youth. On Corellia the aging process of humans is close to zero, so she still looks as beautiful as ever. The player will learn they are in the middle of a galactic war with another alien race, and the player joins the fight when he finds out it's the same race attacking them, as those who destroyed Earth. Cleopatra and the player have fallen in love, and the player will promise to return to Corellia once the war is over.
- Alpha Centuri
    - Home planet of Graqu
    - This is where the home planet of the attacking aliens is located, and where the final battle for survival is held. This mission is planned to "boss mode" with a countdown timer, where the object is to survive until the timer is up, and the supernova weapon is fully charged.
- Supernova
    - When the Supernova explosion starts, the game is "over" and the game goes to video mode.
- Black holes
    - As they end up making a supernova out of the attacking alien's home solar system, and by that the player is pushed by the force of the blast, in the wrong direction, and ends up being sucked into a black hole! That will be the end of part one of the game.

Hours of Game play

We estimate that the player will spend about 10 minutes at each planet, leaving the total game time to about 80 minutes total.

Victory Conditions

The player ends the came when he finishes the campaign and enters the black hole.

Extra Miscellaneous Stuff

Overview

The score will be calculated on how well the player completes the different missions. We may implement a gold, silver and bronze system for how well the player does the job. In the missions that are based on time, the score will be calculated based on how little time spent.

Ideas we are working on…

The Background of the game will consist of real space images, with NASA images of Andromeda, The Milky Way etc, so it will look real.

Planet shops for items and upgrades.

G Class Diagrams from Eclipse

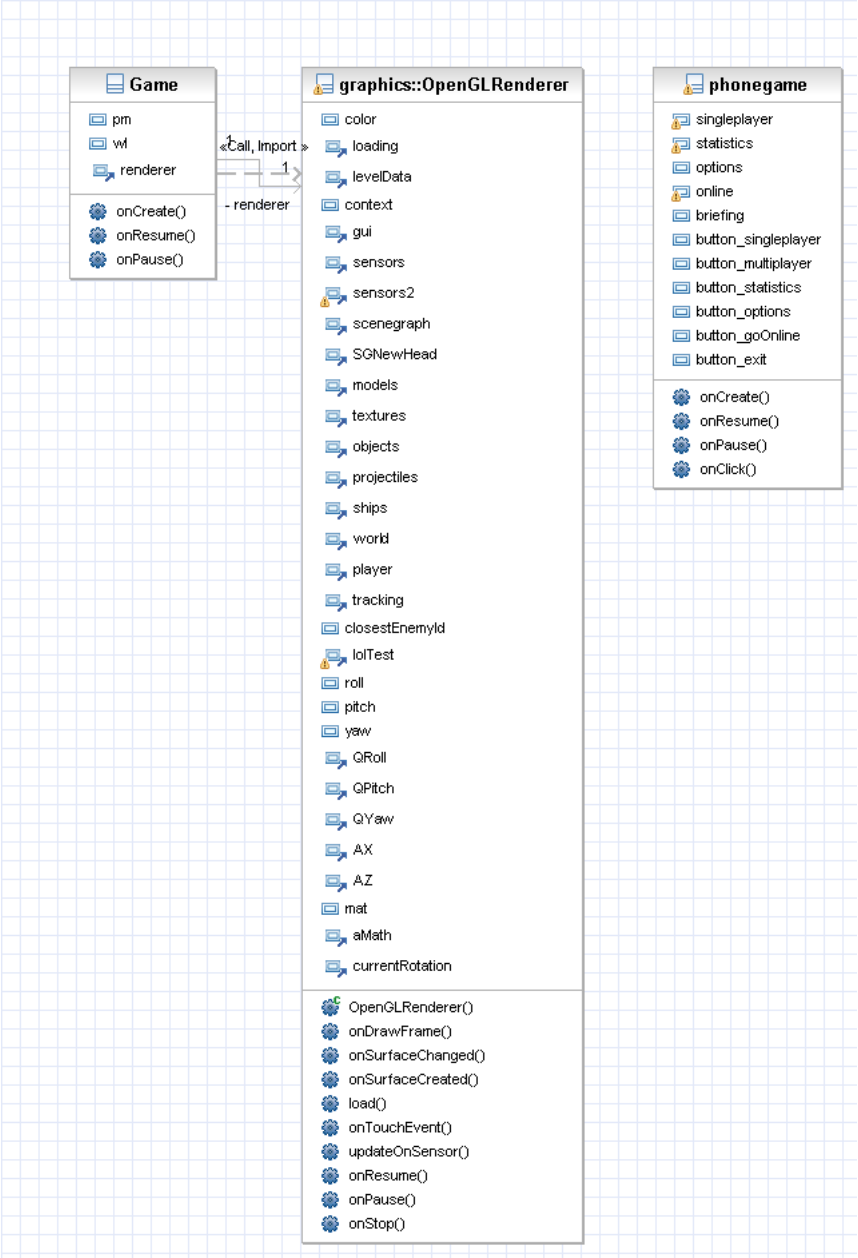


Figure 9: package: phonegame

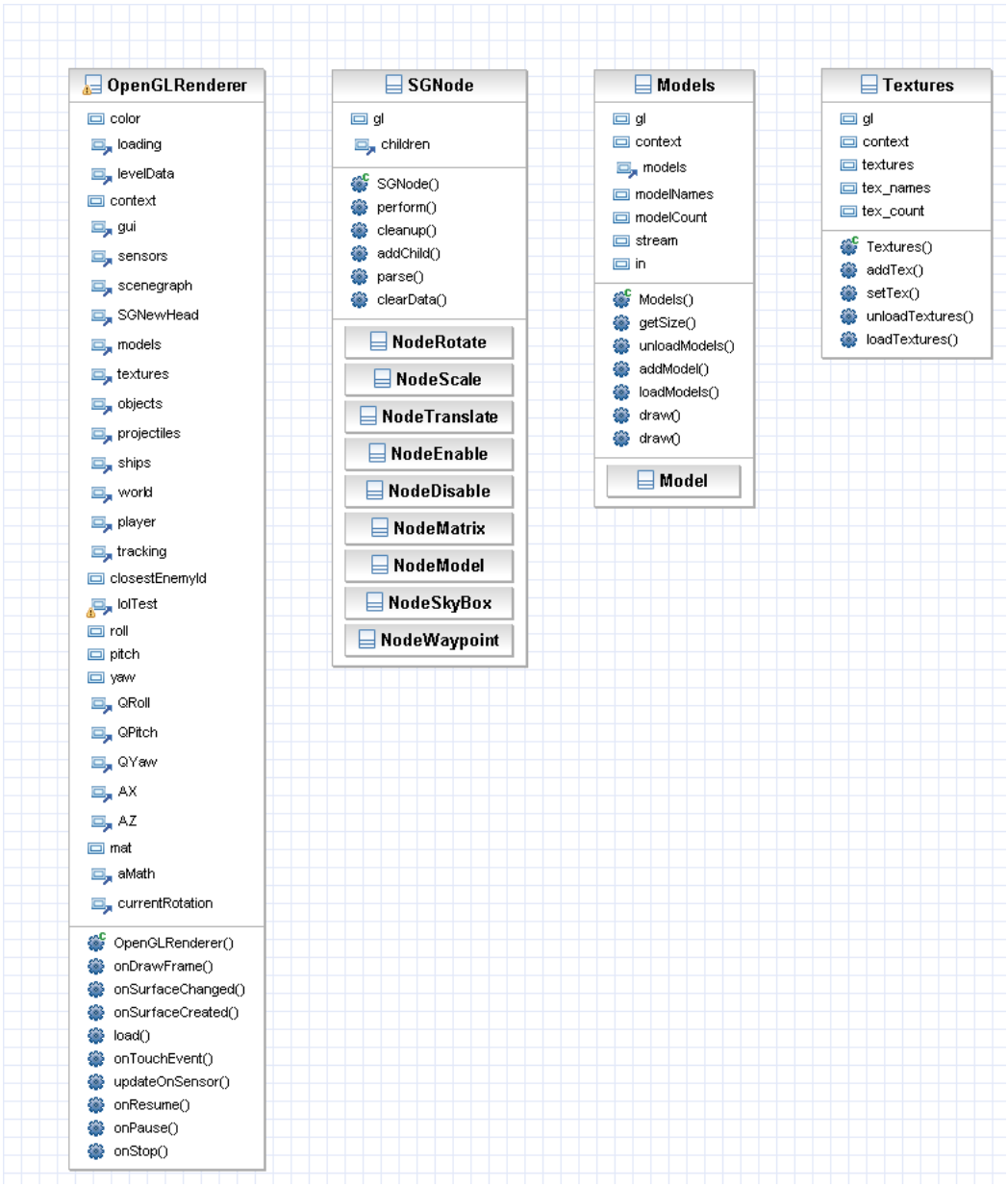


Figure 10: package: graphics

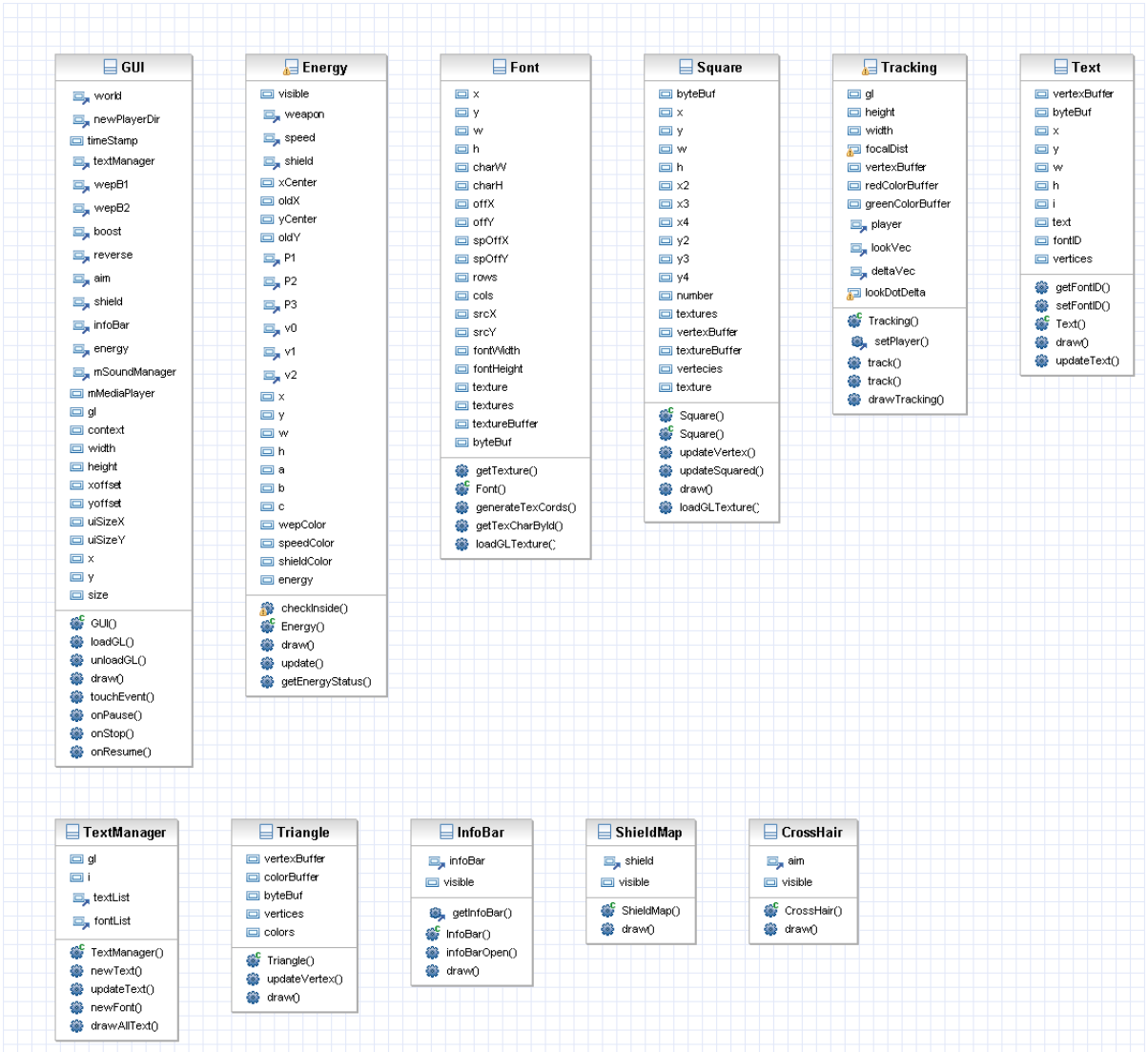


Figure 11: package: GUI

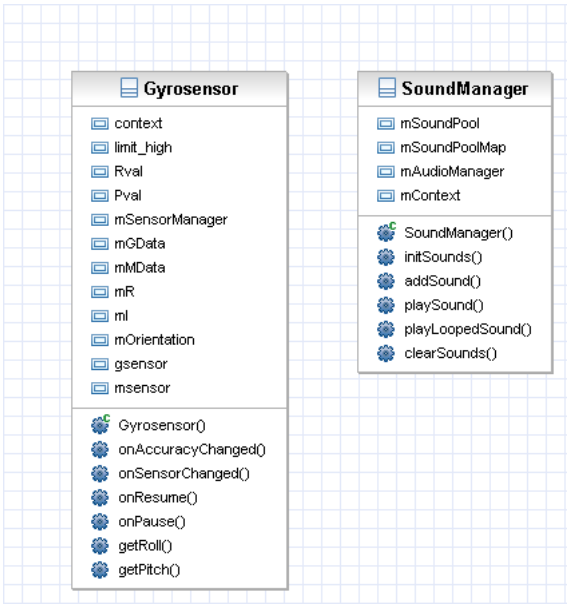


Figure 12: package: hardware

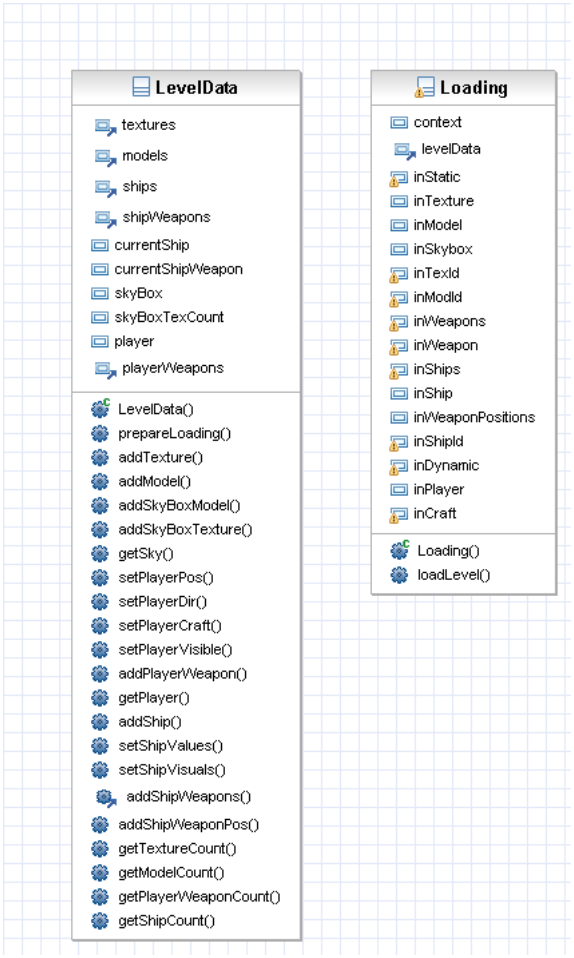


Figure 13: package: loading

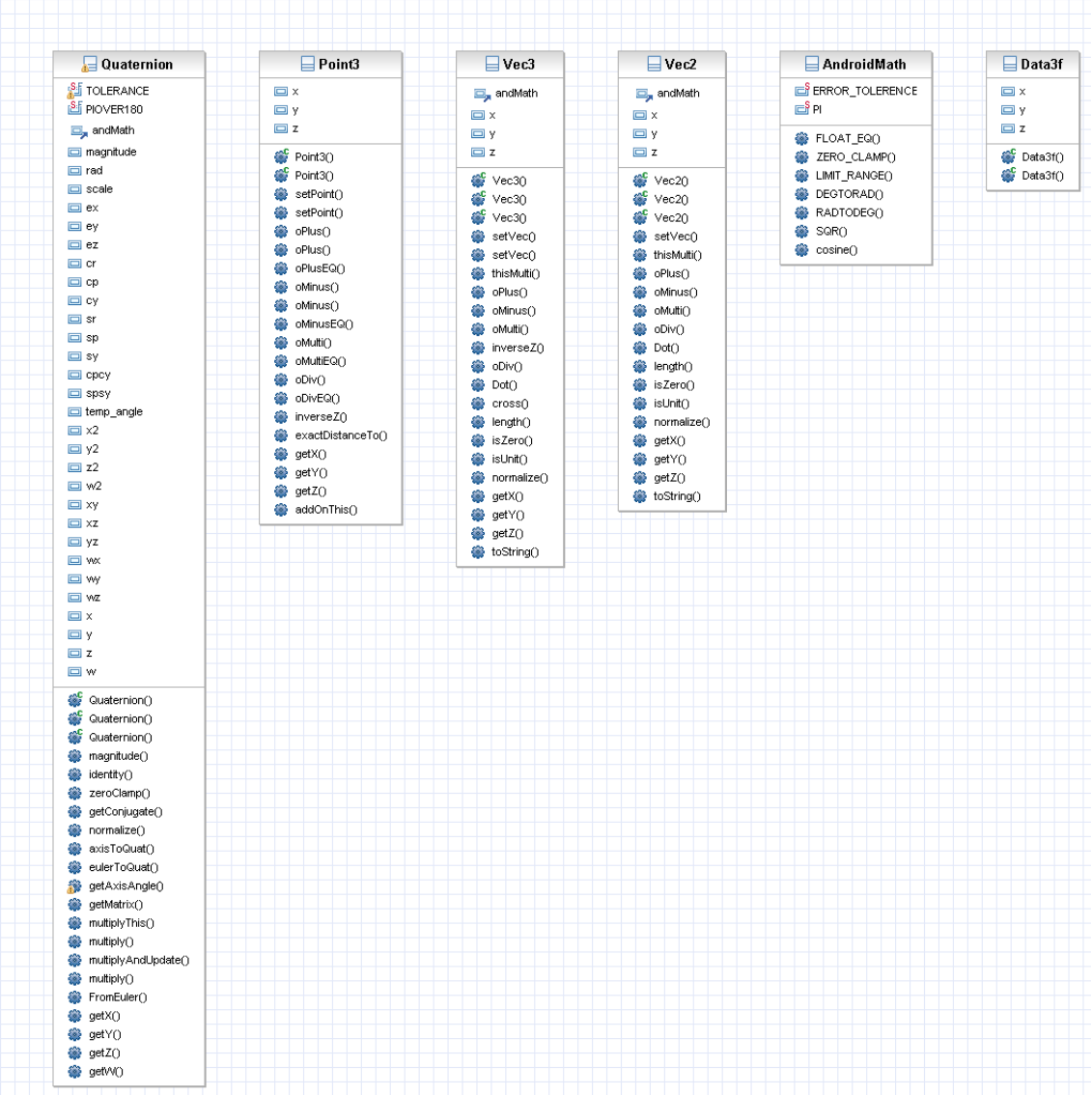


Figure 14: package: math

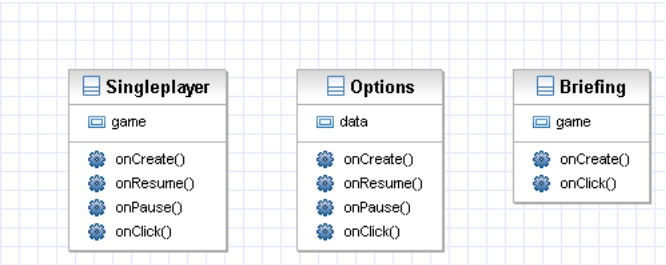


Figure 15: package: menus

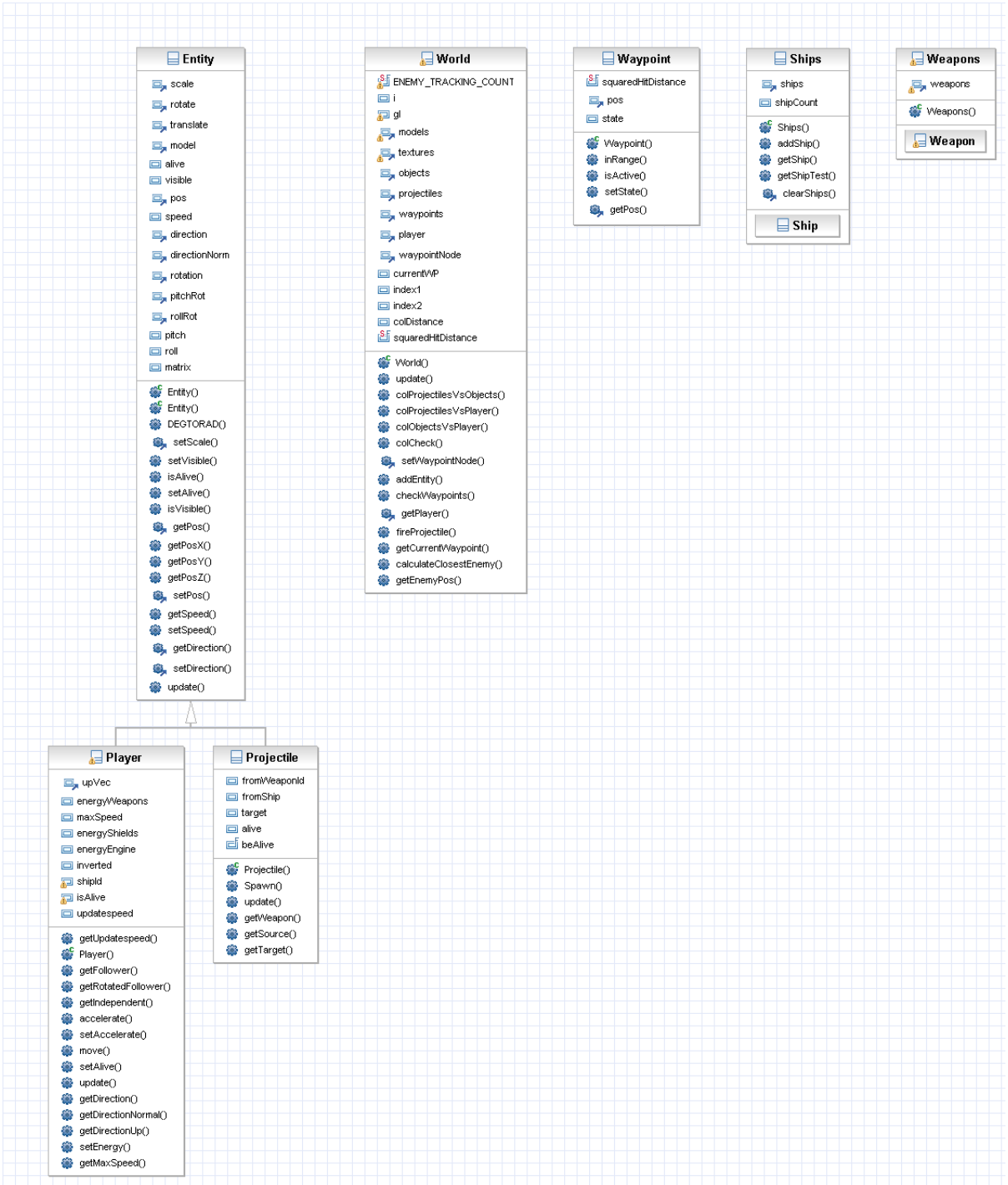


Figure 16: package: menuObjects

H End Gantt Diagram

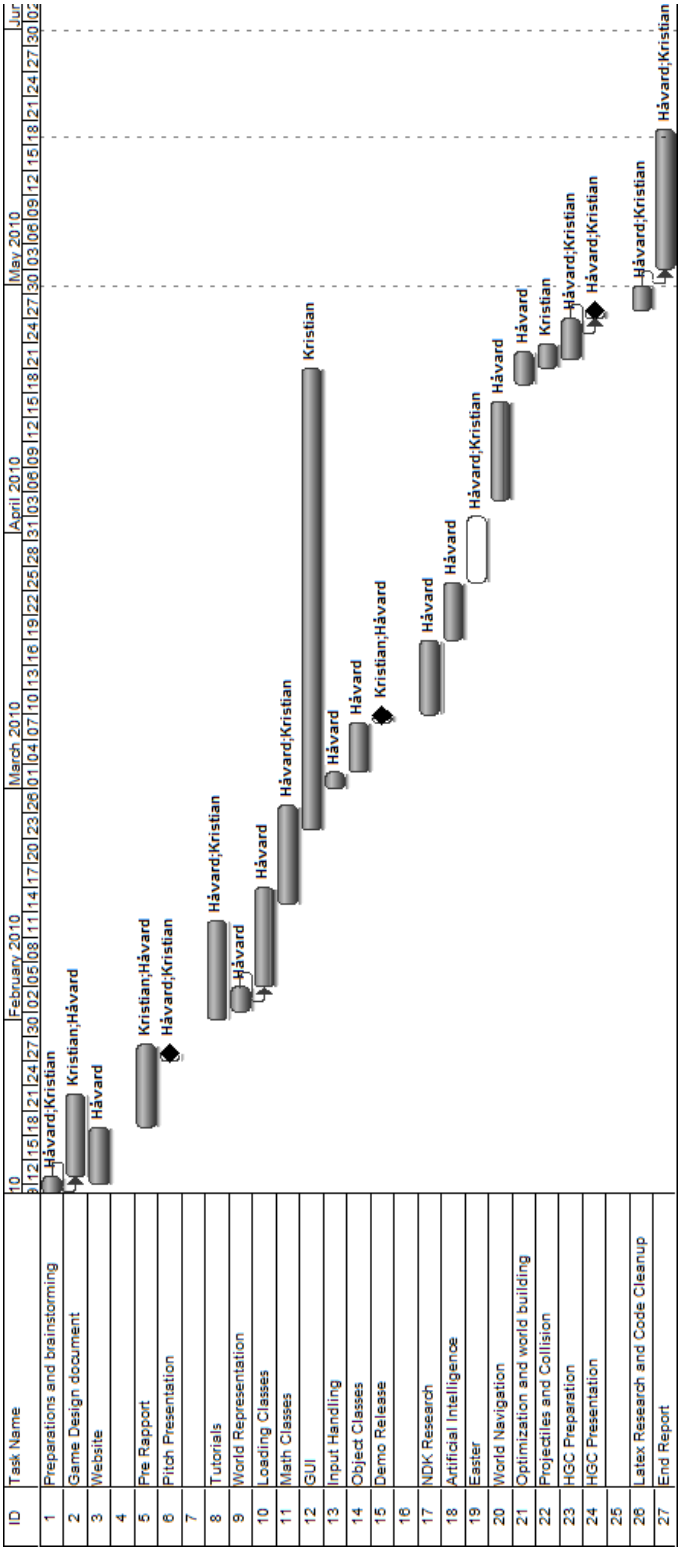


Figure 17: End Gantt Diagram