AC3DS

Aquaculture 3D Simulator
A project for Sintef Fisheries and Aquaculture

Ole Kristian Grashei   Magnus Westergaard

Samson Valland   Per Øyvind Øygard   Audun Bjørkøy

Norwegian University of Science and Technology

November 19, 2009

Preface

This report is written during the course TDT4290 Customer Driven Project at the Norwegian University of Science and Technology (NTNU). The course is a part of the fourth year study in computer science.

The purpose of the course, is to give students a first-hand experience of real-world software engineering. Students are divided into groups, and assigned an external customer who have a project they would like solved. The tasks are usually very diverse, and often on a subject unfamiliar to the students. This, along with requirements specifications, project management, implementation, etc., are part of the challenges.

During the short time the course has existed, it has garnered praise from both industry and students alike, and is by many considered one of the most important parts of the Computer Science program at NTNU.

We would like to thank everyone that have contributed to the project. Extra credits goes to our great customer, SINTEF Fisheries and Aquaculture with Finn Olav Bjørnson, Carl Fredrik Sørensen and Gunnar Senneset, and of course our supervisors Meng Zhu and Bian Wu.

November 19, 2009

Ole Kristian Grashei　　Magnus Westergaard

Samson Valland　　Per Øyvind Øygard　　Audun Bjørkøy

Abstract

SINTEF Fisheries and Aquaculture AS is the leading centre for technological research focused on fisheries and aquaculture in Europe today. One of Sintef's recent investments was the ICT laboratory SSO (Surveillance, Simulation and Operation) which will be used for large-scale research, computation and simulation.

Our task was to develop a 3D aquacultural demonstrator, to show the potential of the new SSO laboratory. Through the use of sonar map data, we were to create a navigatable 3D environment in which the user can place equipment and perform simple simulations and calculations.

In cooperation with the customer we created a backlog of tasks for the project. The implementation was split into four iterative sprints, and for each sprint a suitable task load was chosen from the backlog, based on the customer's priorities.

The final product achieved all the primary goals set forth during the course of the project. Both the customer and the team were very satisfied with the end result. The customer has expressed great interest in the future of the field of aquacultural modelling and simulation. While our product is just the first step towards software that can be useful in this field, it showcases the possibilities of the technology and serves well for demonstrating purposes.

Glossary

3D Studio Max Autodesk 3ds Max is a comprehensive animation, 3D modeling, and 3D rendering software used for game development, television, film, web graphics, multimedia, and marketing communications. [2].

ACE ACE (AquaCulture Engineering) will, by combining science, technology and experience, unite the international aquaculture industry, engineering companies and researchers in a common arena of development to find solutions for the challenges of today and tomorrow.[29].

IDE Integrated developer environment, a program that provides the programmer with tools aimed to boost productivity.

Latex Latex is a document markup language and document preparation system for the TeX typesetting program.

LOD Level of detail is a technique that progressively render objects with less detail the further away they are.

OGRE Object-Oriented Graphics Rendering Engine is a scene-oriented, flexible 3D engine written in C++.

OLEX OLEX is a unique Norwegian developed system for navigation, fisheries plotting and mapping. Using the boat's depth sounder and GPS, the OLEX system collects depth data and continuously calculates and adjusts the measurements. [34] The OLEX data file is a text file containing time, depth, longitude and latitude for squares of about 5x5 meters..

PATS Program for Advanced Telecom Services. PATS is a research agreement between the Norwegian University of Science and Technology (NTNU) Department of Telematics, Ericsson, Telenor and Hewlett-Packard.[35].

Plugin Usually a small piece of software that adds features to a larger piece of software.

Polymorphy Polymorphy is a programming language feature. It allows diffrent objects to be handled uniformly through an interface.

Qt Qt is a cross-platform application and UI framework.

Scrum Scrum is an agile process for software development. [33].

SeaLab SSO Facility for Surveillance, Simulation and Operation [30], is a computer laboratory equipped with modern technology. The laboratory contains software and hardware for designing technical systems, collection and analysis of environmental and operational data, simulation, operation and monitoring..

STL The Standard Template Library is a software library for C++. It provides, among other things, containers and iterators, and is widely used in C++ development.[37].

Trac Trac is an enhanced wiki and issue tracking system for software development projects. Trac uses a minimalistic approach to web-based software project management.[32].

waterfall The waterfall model is a sequential project development model, where each phase will be more or less completed before the next one is started. [16].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we give a short introduction to the customer driven project course as well as the customer for this specific project, the project itself and finally the structure of the report.

The first section gives a short description of the customer driven project course. The course is part of the Computer Science programme at NTNU and is considered one of the most important parts of the programme.

SINTEF Fisheries and Acuaqulture is introduced as the customer of this specific project in the second part of the chapter. SINTEF F&A is the leading centre for technological research focused on fisheries and aquaculture in Europe today. Tight connection to the University of Science and Technology (NTNU) is considered one of the most important reasons for making SINTEF Scandinavia's largest research institution.

In section three, the Aquaculture 3D Simulation (AC3DS) project is introduced. A graphical 3D demonstrator is to be developed, showing the potential of the new SeaLab SSO laboratory at SINTEF F&A at Brattøra in Trondheim. This section also introduces the stakeholders in the project as well as budget and constraints. Among the most important constraints are multiple interests (sensor and customer), limited time, limited experience and limited access to resources.

Finally, an introduction to the report is given together with an overview of the chapters in the report. The report will document the work done during the project in addition to serving the customer with enough technical information to use the software and make further development.

1.1 Customer Driven Project Course

The purpose of the course TDT4290 Customer Driven Project, is to give students a first-hand experience of real-world software engineering. Students are divided into groups, and assigned an external customer who have a project they would like solved. The tasks are usually very diverse, and often on a subject unfamiliar to the students. This, along with requirements specifications, project management, implementation, etc., are part of the challenges.

During the short time the course has existed, it has garnered praise from both industry and students alike, and is by many considered one of the most important parts of the Computer Science programme at NTNU.

1.2 SINTEF Fisheries and Aquaculture

The customer for this project is SINTEF Fisheries and Aquaculture, abbreviated SINTEF F&A.

SINTEF Fisheries and Aquaculture AS is the leading centre for technological research focused on fisheries and aquaculture in Europe today. They drive technological research and development along the entire marine chain[10]. The institute is located at SINTEF SeaLab in Trondheim, in tight connection to the Norwegian University of Science and Technology. SINTEF is Scandinavia's largest research institution.

SINTEF SeaLab contains a modern process hotel for the the distribution of marine raw materials and a processing laboratory for marine fry production. The department also has a flow tank for testing of fishing gear in Hirtshals in Denmark. The project has offices in Ålesund and Vietnam, and a subsidiary in Chile. They cooperate with universities and research institutes in Norway and abroad as well as other SINTEF institutes. SINTEF is also a host institution for the Center for Research-based innovation in aquaculture technology (CREATE).

> CREATE - the Centre for Research-based Innovation in Aquaculture Technology - conducts research to assist in the innovation of technology, products and solutions specifically to improve the grow-out phase of marine fish culture. CREATE is a centre for research-based innovation, established and 50% founded by the Research Council of Norway. SINTEF Fiskeri og havbruk AS (SINTEF Fisheries and Aquacluture) is the host institution for

> the centre. The three Norwegian industry partners involved in the centre, AKVA Group AS, Egersund Net AS and Erlig Haug AS, are all world-leading suppliers of equipment and technology in their respective market segments. Five internationally recognised research institutions are active research partners within the centre: NOFIMA Marin, the Institute of Marine Research (IMR), Centre for Ships and Ocean Structures (Centre of Excellence), the Department of Engineering Cybernetics at the Norwegian University of Science and Technology, and SINTEF Information and Communication Technology. The centre also has research collaborations with the Open Ocean Aquaculture (OOA) Engineering group at the University of New Hampshire, USA (UNH). [7]

1.3 AC3DS Project

The project name was given by the customer, and is Aquaculture 3D simulator. Our group is using the abbreviation AC3DS.

The group has also developed a logo for the project. The logo is neither an official symbol for the software nor a part of any official SINTEF product. The logo, in its full size, is placed on the front page of this report.

One of Sintef's recent investments is the ICT laboratory SeaLab SSO (Surveillance, Simulation and Operation) in their offices at Brattøra. This is a flexible infrastructure for development and testing, and contains machines for heavy computations, servers, graphical workstations, HD projectors and a Smartboard. SeaLab SSO communicates with their large scale research aquaculture plant ACE (Aquaculture Engineering). Realtime measurement data and video are transferred from ACE to SeaLab SSO. Through cooperation with Telenor, SeaLab SSO is also connected to the PATS (Program for Advanced Telecom Services) laboratory at the Telenor office at Tyholt [24].

The project is to develop a graphical 3D demonstrator showing the potential of the new SSO laboratory. The 3D software is supposed to generate an ocean seabed from sonar data and then give the user ability to place aquacultural equipment into this environment. The final solution should preferably be able to measure the amount of equipment used and have the possibility to export the 3D model to an external program for realtime simulation.

The original project description can be found in appendix A.

SINTEF has not put any restrictions on framework or programming language for this project. We may freely choose our development tools. The customers has requested we have productivity in mind when picking tools. In cases where the customer have preferred technologies, these are taken into consideration in the preliminary studies in Chapter 2.

1.3.1 Objectives

In the list below, a summary of the project objectives are stated.

- The project should have a result with a working prototype; a 3D application to place and view aquacultural equipment placed on the water surface, in the water or on the seabed.
- Preferably the result should be an application that is very easy to modify by adding and removing modules/plugins.
- The group will provide the customer with a good report to help further development in the future.
- A written report to be used by the customer for further development, as well as it will be read by the sensor for evaluation of the team and their process.

1.3.2 Stakeholders

SINTEF F&A, the supervisor and the project team are obvious stakeholders in this project. Table 1.1 lists the three persons at SINTEF F&A that have been involved in this project.

Table 1.1: Customer contacts at SINTEF F&A

Name:	**Title:**	**E-mail:**	**Phone:**
Finn Olav Bjørnson	Researcher	finn.o.bjornson@sintef.no	977 26 490
Finn Olav is working to develop a framework that will support the linking the data from farms and numerical models together to form a basis for decision support systems for aquaculture.			
Gunnar Senneset	Senior researcher	gunnar.senneset@sintef.no	920 26 107
Gunnar is the project manager for the ’Simulation and Optimization Framework (SimFrame), a subproject within the CRI CREATE (Center for Research-driven Innovation). He is also responsible for construction and operation of ICT laboratory SeaLab SSO (Facilities for Surveillance, Simulation and Operation).			
Carl-Fredrik Sørensen	Senior researcher	carl.f.sorensen@sintef.no	406 47 344
Carl-Fredrik works on a daily basis with the development of and research into architectures and technologies related to electronic traceability of food and other application areas related to electronic data capture and the “Internet of things”.			

The group was provided with two supervisors during the project. Every week the group had meetings with them and gained useful experience and knowledge. The supervisors are listed in Table 1.2.

Table 1.2: Supervisors at NTNU

Name:	**Title:**	**E-mail:**	**Phone:**
Meng Zhu (Supervisor)	PhD Candidate	zhumeng@idi.ntnu.no	73 55 11 89
Meng Zhu is a PhD candidate in IDI, NTNU. He is now working in software engineering group and his research area is pervasive game technology.			
Bian Wu (Assistant Supervisor)	PhD Candidate	bian@idi.ntnu.no	73 59 17 26
Bian Wu is an teaching assistant in this course and mainly supervises the process of the students team work with the main supervisor. His research area belongs to the game methodology in pedagogical context.			

The team members in the group are listed in Table 1.3. Even though the team members did not know each other before the project we were able to have good group dynamics and work well together.

Table 1.3: Project group from NTNU

Name:	**Title:**	**E-mail:**	**Phone:**
Audun Bjørkøy	Student	audub@stud.ntnu.no	958 05 613
Magnus Westergaard	Student	westerga@stud.ntnu.no	993 57 558
Ole Kristian Braut Grashei	Student	grashei@stud.ntnu.no	486 08 628
Per Øyvind Øygard	Student	peroyvo@stud.ntnu.no	986 82 719
Samson Valland	Student	samsonv@stud.ntnu.no	975 88 926

In addition SINTEF plans on using the software together with the following stakeholders.

Breeding company: Will use the program to design new constructions (usually individuals on a regional level, which is responsible for this)

Equipment suppliers: May receive blueprints and provide offers of equipment, help with dimensioning of chains and mooring.

Government: Should receive a final plan for archiving.

Researchers: Design and take out finished models and use the forward in simulation tools.

1.3.3 Budget

The estimated workload is 24 hours per student per week. As our group consists of five students, the total workload throughout the duration of the project is approximately 1440 hours.

The project kick-off is the 25th of August 2009 and the projects continues until the final presentation at 19th of November 2009.

1.3.4 Constraints

The project is facing several important constraints. The most important are listed below.

Multiple interests: One of the essential constraints to be aware of, is the fact that both the report and the software must be useful and logical for both the sensor and the customer. This means that we will have to compromise between the content of the report and the functionality of the software.

Limited time: Another constraint is the limited time on the project. Each member of the team is supposed to use 24 hours each week, giving the whole project about 1500 hours of work. Producing a well working prototype and a written report requires the team members to work efficiently and prioritise the important areas of the work. There are a lot of meetings and seminars making the actual coding/writing time even less.

Limited experience: Some of the team members also have limited experience with 3D programming. A lot of time will be used for getting to know the 3D engine and the programming language chosen for the project.

Resources: Limited access to resources such as printers, facilitated rooms for the group to work and powerful computers to run the 3D software may as well be considered as constraints. The team has free access to rooms at NTNU Gløshaugen as long as they are not occupied by other students.

1.4 Report Structure

In this report we document the work done by us during the course of the project, in the form of pre-studies, requirements specification, implementation, testing and evaluation. In addition to this, the project mandate and plan serves as a guideline for work and interaction in the group during the course of the project. The following list gives a short description of each chapter.

Introduction will give a short introduction to the customer driven project course as well as the customer for this specific project, the project itself and finally the structure of the report.

Project Plan presents the relevant information about the project, customer, task, goals, etc, in addition to routines and guidelines.

Quality Assurance contains a set of rules and routines to make sure the quality of the process and the final product is as expected.

Preliminary Studies documents the work done in preparation for the implementation, including studies of relevant technologies and software solutions.

Requirement Engineering describes the task outline, based on the problem given to us by the customer.

Software Architecture presents the core architecture chosen to support essential features in the program.

Sprint chapters contains a short description of each sprint, including the goals, design, implementation, testing and evaluation.

Evaluation contains our own evaluation of the product, the process, the results, the customer and the task. We have also given our evaluation of the course, and work remaining for the product to be useful for the customer in the future.

User Documentation contains information about installation and how to use the software.

Conclusion is a short conclusion on the project as a whole and the final product.

Chapter 2

Preliminary studies

In this chapter, the result of the pre-study work is presented from problem to solution perspectives. Research and decision on the development process was the first thing to get done. This is important for the planning of the rest of the project. Two alternatives were discussed; waterfall and Scrum.

Clearing out the customer domain knowledge was also done at an early stage. Knowing what resources the customer has available is important, both in consideration of help during the project and for better understanding of what the final product is supposed to do.

Finally relevant technology the team might use for the project is studied. Different candidates are presented in several categories, evaluated and a conclusion is made. These conclusions was a product of both the teams research and the customer's wishes.

2.1 Development Process

2.1.1 Waterfall Model

The waterfall model is a sequential project development model, where each phase will be more or less completed before the next one is started [16]. When using the waterfall model, the developing team has little or none contact with the customer during the implementation of the code. All requirements and demands are set before starting implementation.

When the implementation is done, the software is tested according to the requirements set at the beginning of the project, and bugs are fixed. Then the final product is delivered to the customer. Figure 2.1 illustrates how the waterfall method works.

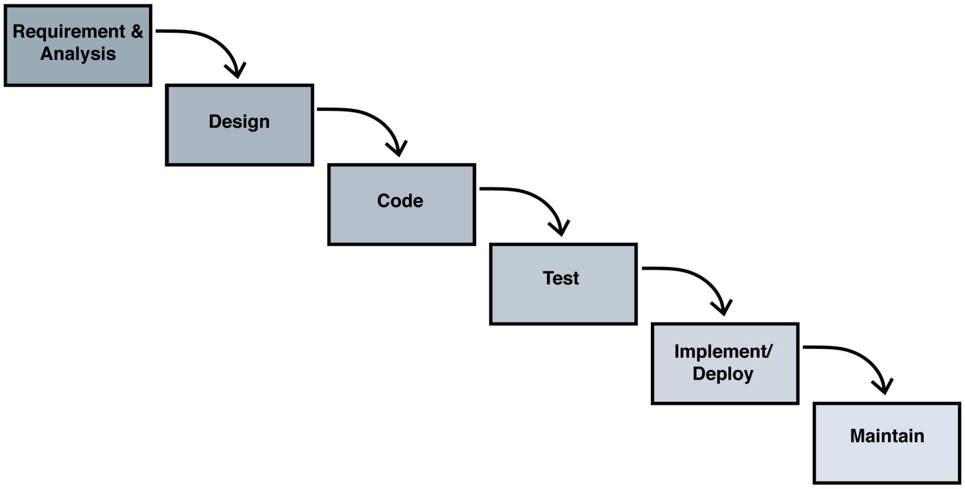


Figure 2.1: The waterfall method

Using the waterfall model reduces the opportunity for the customer to change requirements during the implementation. Hence, the software can turn out to be what the developing team wants, and not what the customer had in mind. Therefore it is crucial to use a lot of time on pre-study and planning with the customer, to ensure that the team has the right idea of how the software is supposed to behave.

2.1.2 Scrum - agile method

Scrum is an agile process for software development [33]. Projects start off with some initial project planning, creating a product backlog and then progress through series of iterations called sprints being 2-4 weeks long.

The product backlog is the core of the entire project. This can most easily be explained as a prioritised list of requirements from the customer. The team will use this list as the basis for the process and development. For each sprint, it is an estimation phase where team members choose tasks (often called stories) from the back log, starting with the tasks with the highest priority. Each item in the backlog is then divided into subtasks, and time for each of these are estimated. All team members participate in the time estimation of each task. The mean of all suggestions are usually a good estimate. If a member believes a task will take much longer than the other team members, he must explain to the others why he thinks so. Team members will select an appropriate number of elements in relation to the time that is intended for the sprint. This process is done in a meeting called the sprint planning meeting.

Figure 2.2 illustrates how the Scrum framework is implemented.

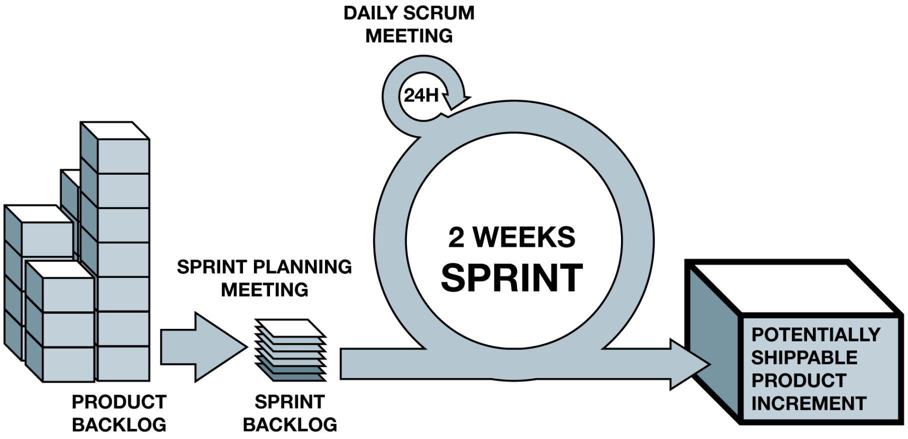


Figure 2.2: The Scrum framework

When using the Scrum framework, the team should have a stand-up meeting of about 15 minutes each morning where each member quickly go through what he or she has done since the last meeting and what the plan for the day is [13]. Each team member also gives a quick brief on problems that might arise, but the meeting is not intended for discussion and problem-solving. However, the meeting initiates problem solving, as members can get a hint of other team members who might help solving the problem. It is crucial to keep the Scrum meeting short and efficient. The best way to do this is making each member stand up when they present their work. This prevents the meeting from getting a casual atmosphere.

Each item in the sprint backlog will have an id number, a priority, a reference to other backlog items and a description. It will also contain a time estimate, and after one task is done the actual work hours used on the task will be added. This helps the team evaluate their time estimation skills and could also be helpful for tracking time spent on the project.

After each sprint, the team will arrange a demo-meeting where a demo will be presented for the customer. Basically everyone can attend on this meeting, but at least the product owner and the developers should be present. This meeting gives a good opportunity for the customer to give feedback to the team. At the very end of the sprint, the team will have a review meeting, where they can share experiences and improve the process for the next upcoming sprint.

2.1.3 Conclusion

Which development model to use, relates to what kind of project and software we are dealing with. This project is quite large, comprehensive and relatively undefined. This means that we are depending on much contact with the customer and quick feedback on our work. Table 2.1 compares the the most important differences between the two methods.

Scrum is probably the best model for this project. By using Scrum we are able to see working software at an early stage, and misunderstandings or changes in specifications will appear in time to be fixed. It was also important for us to involve the customer in the project, as they want to build on our product later.

2.2 Domain Knowledge

Being aware of the domain knowledge of the customer is quite important in every project. The customer often has expert knowledge in the field where the software is going to be used, and can provide valuable or even critical knowledge to the developing team.

Working with SINTEF Fishery and Aquaculture is quite pleasant when coming to

Table 2.1: Scrum vs waterfall comparison [36]

Waterfall features	**Scrum features**
Managemet methodology	Management framework
Project splits into various phases. One after the other.	Project splits into sprints.
Product manager defines work and prioritises them.	Product owner writes story points and prioritises them.
If the schedule is not achieved, we change the schedule.	If schedule is not achieved, we adjust the weight of the estimates in the next sprint.
Planned project.	Planned sprints.
Less flexibility to the customer.	More flexibility to the customer with controlled cost.
Typically weekly monitoring.	Daily Scrum meetings.
Project planning will be done at the beginning of the project.	Planning will be done at the beginning of each sprint.
Project postmortem meeting will be held at the end of the project, and experiences noted.	Review meetings will be held at the end of each sprint, giving the team a possibility to improve during the project.
Project is delivered and closed when all the requirements are met.	The project is delivered in iterations.
Project manager is responsible for the final result.	The team is responsible for the final result.
Customer's involvement is less.	The customer is involved in the project and present on every sprint meeting.
Can apply anywhere and anytime	Can be a challenge when there is large geographical distance between the team and the customer.
A finished product may not be "done"	Every release is a complete "done"

domain knowledge. They are all researchers who are experts in several fields when coming to fishery and aquaculture. They also have great knowledge and experience in coding and know much about the different tools, languages and OpenSource projects available. The most important domain knowledge of the customer for this project is listed below. The developing team will have to communicate with the customer to get access to the knowledge they need.

Map data: Developing a 3D simulator for fishery, requires access to an knowledge about geographical data. SINTEF F&A were able to provide us with high resolution maps and comments about how the map data should be handled. The map data comes from sonar data and is represented in the OLEX-format [34]. The OLEX data file is a text file containing time, depth, longitude and latitude for squares of about 5x5 meters.

Aquacultural equipment: The developing of 3D models and placing of these. No one in our team has any experience with such equipment. Drawings, comments and explanation from the customer is needed to get the models as real as possible. SINTEF F&A have has provided the group with models of different equipment in 3D Studio Max.

2.3 Relevant Technology

The combination of technology used to create and run the application is complex. At the core a 3D engine is needed to present the map data, the water and all the fish farm equipment visually, and to let the user navigate within this 3D world. This engine must be combined with some sort of GUI framework to make a nice interface for the user to manipulate the world and its contents. To let the user save the world's state, a good way of serialising the underlying objects is required.

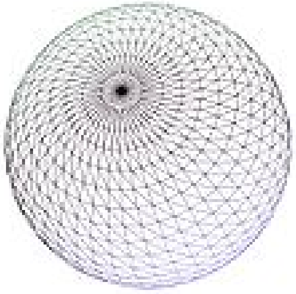
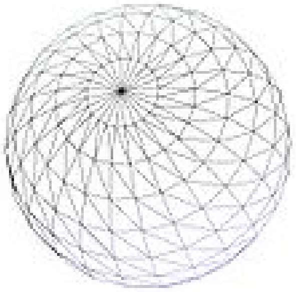
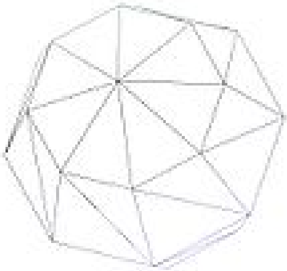
All of these, in addition to tools used for development, are discussed in the following sections.

2.3.1 3D Engine

In this chapter we review several different solutions for the visualisation part of our program. A large number of 3D engines exists, and we have searched for the most complete and sound among these projects. Many engines are still in early development, have no users or document base, or the project may already be abandoned. A handful of the best engines are reviewed here.

The alternative we use must be well documented and be easy to use. Speed is also of particular interest since we will be rendering large outdoor scenes. Features such as level of detail (LOD) and some kind of a spatial organisation of our static terrain will be crucial to obtain sufficient speed.

Table 2.2: Level of detail illustration

Level	High	Medium	Low
Mesh			
Notes	Highest LOD used for close objects		Lowest LOD used for far away objects

2.3.1.1 Candidates

OGRE (Object-Oriented Graphics Rendering Engine)[15] is a open source engine written in C++. OGRE abstracts the underlying rendering mechanisms, and provides a focus on high level classes and world objects. The engine can use both OpenGL and Direct3D, is cross-platform and wrappers exists for several languages such as Python, Java and the .net family. One of OGREs feature is that it's purely a rendering engine, as such it only exists to provide rendering capabilities. Other functions such as physics may be implemented in any way the end-user wants, OGRE doesn't put any restrictions on this. OGRE has a well documented interface and a large community at http://www.ogre3d.org/. Has a large number of tutorials from beginner to advanced levels. Several GUI frameworks are supported by OGRE, CEGUI is the "official" GUI pack. OGRE is scene-graph based, which include amongst other a scene manager for outdoor terrain. Released under the terms of a modified GNU Lesser General Public License.

OpenSceneGraph [5] is a 3D toolkit written in c++ using OpenGL for rendering. OSG runs on a multitude of platforms including Windows, OS X, Linux and more. Well documented at www.openscenegraph.org. OSG is a scene graph, but most features of a 3D engine is provided through Node-Kits which can be loaded at runtime or precompiled in. A Node-Kit for terrain rendering exists and LOD for speed is a

Table 2.3: 3D engine evaluation

Engine / Criteria	**OGRE**	**OpenSceneGraph**	**Irrlicht**
Version	1.6.3/26-07-2009	2.8.2/28-07-2009	1.5.1/05-08-2009
Tutorials	Good, from beginners to intermediate levels	Good	Good
Documentation	Ecellent	Excellent	Good
Updates	About quarterly	Atleast quarterly	Twice a year
Community	Ecellent	Very good	Decent
Scene management	Large quantatiy of scene managers (bsp, octree, terrain etc)	Core feature via scene graph	Hierarchical scene graph
LOD	Via scenemanager	Core feature	Not supported
.NET support	via MOGRE	osgDotNet wrapper	Official port in alpha version

core feature. OpenSceneGraph Public License is based on the Lesser GNU Public License (LGPL).

Irrlicht Engine [12] is a free open source 3D engine, completely cross-platform. Written in c++, but unofficial bindings exists for most popular programming languages. OpenGL, Direct3D and a software renderer are all available. A skinnable GUI system is available, Irrlicht has an internal event system that provides mouse and keyboard events. No external library for input is needed. Plugin support for custom mesh loaders could be useful for loading map data or other custom models. The engine is licensed under the zlib license.

2.3.1.2 Evaluation and conclusion

All the 3D engines evaluated have the functionality required by our program. The differences between them are still evident. OGRE is the most complete package in terms of features and built in functionality. OGRE has the most tools for a complete 3D software implementation. OpenSceneGraph is a very sound tool for the most important part of our 3D part, namely the scenegraph. A scenegraph is the core of a scenemanagers functionality, it is the structure that organise our 3D world into more manageable parts. The scenegraph will speed up the rendering via visible object detection. Reducing the number of visible polygons is very important when rendering large areas like we will be doing.

A core feature when rendering large outdoor areas is level of detail. LOD means objects close to the camera will have high detail and objects further away will have less details. The level of detail should be progressive based on distance. Often the user will not even be able to see that the detail declines on objects in the background. OpenSceneGraph and OGRE has LOD support, but Irrlicht does not.

In conclusion we arrive at OGRE as the most sound package for rendering and managing our 3D scene. OGRE has a large community and the support is massive for this open source project. There are are plenty of tutorials, documentation and demos showing OGREs capabilites. This engine will not hold our project back, but allow us to build the program the customer wants.

2.3.2 Programming language

The programming language in which we write the application must meet several conditions. To be practical and easy to program it should be relatively high-level, support object-orientation and be somewhat known to the team. A large user base and good documentation is essential to exploiting the language to its fullest and to find solutions to common problems easily. Most important of all, it must be supported by the 3D engine that has been chosen.

This section presents the most known and supported languages, and lists some of their key features and differences.

2.3.2.1 Candidates

C++ [6] is a well known and widely used multi-paradigm language today, originally developed at Bell Labs. It is a characterized as a middle-level language because it contains both low-level and high-level features. Originating as an improvement to the C language, it was standardised in 1998 [ISO/IEC 14882:1998].

The language is used everywhere in the software industry today, is thoroughly documented and easy to find resources and get help for. A key characteristic is its support for object-orientated programming, which allows programmers to easily implement designs that are made in a logical and practical way. OOP also encourages the reuse of code and modularity.

Compiling C++ code is easy and can be done on almost any platform without the need to change much, if any, of the code. The resulting code is very efficient, and C++ is known to be a fast language compared to some of its competitors.

Writing code is effective and the result should be easy to navigate and read. It can be divided into several files and linked. If a project is compiled and a small change is made, this ensures only the file where the change was made has to be recompiled.

When using C++, the programmers must know how to allocate and handle memory manually if they are to avoid memory leaks and other similar problems.

C# [17] is a multi-paradigm, compiled language developed by Microsoft and is approved as a standard by ECMA [ECMA-334] and ISO [ISO/EIC 23270]. It has a modern object-oriented syntax based on C++, but adds powerful features to simplify some of the complexities in C++. C# supports generic methods and types, which provide increased type safety and performance, and iterators, which enable implementers of collection classes to define custom iteration behaviors that are simple to use by client code.

C#'s reference compiler is Microsoft Visual C#. Open source compilers and compilers for other platforms exist, for instance Mono, DotGNU and Microsoft's Rotor project.

The code is easy to write and read, and migrating to C# should not take a lot of time for someone who already knows C++, Java or other similar languages.

Java [20] is a programming language developed by Sun Microsystems. It is almost exclusively object-oriented and has syntax similar to C++. The language is very well documented and a lot of resources concerning it are easily available.

The big characteristic with Java is its platform independency. The code is first compiled to Java bytecode, which is then run on a virtual machine (VM). Implementations of the VM are available for all popular platforms.

Because it interprets or compiles bytecode into machine code using a VM instead of compiling directly to machine code, Java suffers when it comes to speed. Optimisations over the last years, like the introduction of just-in-time compilation in version

1.1, have improved the performance.

Java has automatic memory management using its garbage collector. This ensures that the memory used by objects is freed when the objects are no longer in use. There is no direct control of the garbage collector, but it guarantees that actions are taken at certain times.

2.3.2.2 Evaluation and Conclusion

None of the candidates have major drawbacks that make them unsuitable as a programming language. Overall, they offer mostly the same functionality. It is natural to choose an object-oriented language for this project, as there will be a need for containers representing for example the 3D models and the data assosiated with them, such as cost and size. This data needs to be exchanged easily between the 3D engine, the data model and the graphical user interface. All three candidates provide this functionality through objects and functions/methods. Choosing one language comes down to other factors; how well it interfaces with the 3D engine chosen to power the application, and the customer's preference. If the customer has no opinion, the group's experience and preference can be taken into account.

As requested by the customer, the team has chosen C++ as the programming language for the project.

2.3.3 Framework

Using a framework lets the team take advantage of good solutions to common problems, effective code for common tasks, well defined models and interfaces. Coding within a framework might also let other applications use our application's functionality more easily.

2.3.3.1 Candidates

.NET Framework [19] by Microsoft is a framework which is intended to be used to build applications for the Windows platform. It includes a large library, with features for use in making graphical user interfaces, to help access data, web application development, network communications and more. The other part of the framework is the Common Language Runtime (CLR), a runtime environment all programs written for the .NET framework execute in. It provides services like security, memory management and exception handling. It also provides the appearance of an application virtual machine so that the programmers need not consider the capabilities of the hardware of the specific host that will execute the program.

The framework uses a Common Type System (CTS) to define all the possible datatypes, constructs and interactions between them that the CLR supports. This allows for the exchange of datatypes between programs written in different languages, as long as the languages conform to the Common Language Infrastructure (CLI). Most programs written in such languages are first compiled to the Common Intermediate Language (CIL), which is then executed using an implementation of CLI, for instance the CLR.

Qt [23] is an open-source application development framework developed by the norwegian company Qt Development Frameworks, formerly known as Trolltech. Qt includes a wide array of tools for developing cross-platform applications, but is perhaps best known for its widget toolkit which is used extensively by many open-source and commercial software projects.

While Qt is natively written in C++, it has bindings for most major programming languages, including C#, Python, and Java. In addition it supports all major platforms, and offers many tools and libraries to ease cross-platform development, including threading, SQL database access, network support, and file management. The widget toolkit uses native APIs for creating GUI widgets, and as such allows for a very easy way to get a clean and consistent look across platforms.

2.3.3.2 Evaluation and Conclusion

The GUI abilities of these two frameworks are both good. Using .NET enables the application to use Windows Forms and Windows Presentation Foundation, both of which give access to the native Windows interface elements. Qt is less platform dependent, and uses whichever native interface elements the host platform has available. Both candidates are have sufficient GUI capabilities for this project.

In addition, the .NET framework has a Plugin framework to ease the development of plugin-based applications. It uses a model that allows the plugins to be separated from host and enforces a contract to which all communication must abide. The plugins are given specific permissions, and problems are contained within them so that the host application does not crash if a plugin does.

While it's possible to the .NET framework with C++, it requires the use of a special dialect known as C++/CLI, which is significantly different from standard C++. Among the major differences is that C++/CLI is managed, and doesn't require memory allocating. While this is a useful feature, nobody in the group had any experience with this dialect. Considering the challenge of the task, and the fact that the group had litle experience with 3D programming in general, we decided against using C++/CLI, which also ruled out the use of .NET for C++.

Because the customer preferred C++ over C#, the choice fell on Qt. Qt is a tried and true framework, and has a wealth of information and tools online, which should make working with it relatively pain-free.

2.3.4 Integrated development environment (IDE)

The development environment is an essential tool when developing applications. It provides a suite of tools designed to maximise productivity and help the developers design, implement, test and deploy the code. It is often used in conjunction with other tools that aid other aspects of the project, and may interface directly with these.

2.3.4.1 Candidates

Microsoft Visual Studio [18] is an IDE made by Microsoft. It can be used to develop applications from a range of application types in both native code and managed code for platforms including .NET Framework.

The software includes a code editor, an integrated debugger, tools to build graphical user interfaces, design classes and more. It also has very good plug-in support, which can be used to extend or improve its functionality in all areas. Examples include integration with SVN and tools to keep track of the software development process.

Any language is supported in Visual Studio, provided that a service for that language is available. Such a service can be authored by anyone. Languages supported natively include C/C++, VB.NET and C#. For these languages, more limited versions of the software are available, including Microsoft Visual C++, Microsoft Visual C#, Microsoft Visual Basic and Microsoft Visual J#. Most of these versions are also available in Express versions that are free. Professional editions of Microsoft Visual Studio 2005 and 2008 are also available for free to students through Microsoft's

DreamSpark website.

Visual Studio is, since it's a Microsoft application, only compatible with Windows. Any use on other platforms has to be conducted through a Virtual Machine.

Eclipse [11] is a free and open source development environment with extensive plug-in support. It is written in Java, and is by default set up to develop Java applications. Plug-ins enable support for other languages such as C/C++, Python, PHP and others, and can extend and add functionality on every level, such as SVN support, support for editing typesetting languages like Latex and creating UML diagrams. Since it's written in Java, it's available on most modern platforms, including Windows, Linux and OS X.

The editor supports syntax highlighting, refactoring, easy navigation of code through links and other tools.

Eclipse is released under the Eclipse Public License, which is a free software license not compatible with the GNU General Public License.

NetBeans IDE [21] is a free open source developer environment written in Java using the NetBeans platform. It supports the development of applications with Java, C/C++, PHP and other dynamic languages. Its multi-language debugger has many tools to let the user supervise and dictate the execution of code, such as taking snapshots of the application state during execution. Like Eclipse it's avilable on all most platforms.

For Java applications the IDE provides a GUI builder to easily design graphical user interfaces visually.

2.3.4.2 Evaluation and Conclusion

All the IDE candidates proposed are high quality software packages that share common and important features. The basic code editor with syntax highlighting, smart indentation functionality, auto-completion of code, easy browsing of existing code and a range of hotkeys to execute common tasks is provided by all alternatives. Small differences separate them, but different languages also handle differently in any one of them. These differences have almost no impact on the productivity of the team, as it will take very little time to adjust to whatever IDE and language that is used.

Integrated in all the candidates is also a debugger. Again, most of the basic and most used tools are available in them all; realtime check for syntax errors, easy insertion of breakpoints in the code, the ability to run through the code step by step etc.

The major decider for IDE is which language is chosen. For C++/C# Visual Studio is the only IDE that comes with a native compiler. For the other IDEs a compiler will have to be added seperately, and will likely incur extra work related to installing a GCC compiler environment or similar. On the other hand, Visual Studio only runs on Windows, and since a majority of the group runs OS X, this will also cost some time, not to mention that it will limit us to building Windows binaries. If the choice falls on Java, however, the decision is far more open, since all three IDEs use the same compiler.

In the end the choice fell on Visual Studio. The customer expressed a wish for using C++, and did not have any particular reservations about cross-platform compatability as Windows is their primary platform. The relatively hassle-free setup of Visual Studio was a big point for the group. Even despite the need for Virtual Machines, building for a single platform will likely save a lot of time since nobody has much experience developing for OS X. Visual Studio is also the standard internally at Sintef.

2.3.5 Serialisation Technologies

Serialisation is a necessity in this project. The environments and setups created using the product will have to support saving to a persistent medium. This means saving the state of all the data objects in memory in such a way that they may be loaded back up later. Support for serialisation varies with the languages; some of the older languages lack native support, while the newer ones have several options when it comes to serialisation.

2.3.5.1 Candidates

JAVA supports serialisation natively. Any object which implements the Serializable interface can be saved to a specified file or database. JAVA handles all the serialisation of the objects. If certain methods are implemented by the user, the serialisation process can be modified somewhat. To get even more control over the serialisation process, the Externalizable interface can be implemented instead.

The default encoding is a translation of the fields into a byte stream. There are rules for how objects referenced by the object to be serialised have to be either serialised themselves or marked for other uses. If these rules are not obeyed, the serialisation will fail.

.NET languages also supports serialisation natively. The class needs the Serializable attribute, and members need to be tagged with the OptionalField attribute. Members can also be tagged with the NonSerialised attribute to indicate that it should not be serialised. Further modification of the serialisation process can be made if a class implements the IDeserializationCallback interface and define the IDeseriabilitizaionCallback.OnDeserialization method.

Objects can be encoded using XML via the XmlSerializer object, or binary format for use in other .NET applications.

C++ does not natively support serialisation. Objects may be written to file in a binary format.

Boost libraries [14] is a set of peer-reviewed open source libraries for C++. Among them is a library for serialisation f data structures. It allows for entire objects to be serialised, including recursive serialisation of member objects and pointers. The target of the serialisation is an instance of the Archive class, whose implementation dictates how the data is rendered and stored. Having the two processes separated like this allows for several types of archives, like XML, plaintext and binary, to be used independently of how the classes are serialised.

Qt offers serialisation of Qt variables through QDataStream. QDataStream can output binary data to any QIODevice, which includes QFile, and takes input from most primitives like int/float/char*, and also native Qt objects like QColor, QPixMap, and QDateTime. It does not, however, support STL containers like Vector and Map.

QDataStream does not offer serialization to any other formats than a binary stream, without additional library support.

2.3.5.2 Evaluation and Conclusion

Between Java, Boost and .NET, the capabilities are quite similar. Java and .NET are native, which means a slightly lower implementation cost, but largely the choice depends on what programming language we end up using as they are all more than up for the job.

Qt differs from the other options in that it's quite simple, and requires a lot of maintenance to work. There is no automatic serialisation of objects, nor any kind of recursive support, meaning all variables have to be serialised manually. It also only supports a limited amount of data types, which will likely mean conversion will be necessary. Despite having chosen Qt as our GUI library, in light of these things, Qt is not a good fit.

In the end we've decided to use Boost, primarily as a result of choosing C++ as our programming language.

2.4 Proposed Solutions

Both suggestions to solutions were based on the team using the OGRE 3D engine to do the core graphical work.

2.4.1 C#, .NET and MOGRE (Managed OGRE)

One of the solutions presented to the customer was the combination of C#, .NET and MOGRE. MOGRE is a .NET wrapper of OGRE, and shares the same functionality as the core OGRE engine. C# would let the team develop using a high-level language similar to Java, which all the team members are familiar with. .NET provides good tools to build GUI elements, and integrating MOGRE did not take much effort. .NET also provides a namespace to build plugins with a defined model for interaction between the host application and the plugin.

2.4.2 C++, Qt and OGRE

The other solution presented was the combination of C++, Qt and OGRE. This solution makes the application more platform independent, avoiding using .NET's

Windows specific GUI components and C#. C++ might complicate the development a bit for the team, as we are not as competent using this language. The language is, however, very well documented and widely used, so solutions to problems are easy to find.

2.4.3 Conclusion

When presented with the two solutions, the customer favored the second one. The team sees the two solutions as equally suited to develop this application, and was prepared to use either one. This made it an easy decision to go along with the customer's wishes and use the combination of C++, Qt and OGRE.

2.5 Chapter summary

By the end of the preliminary studies we had examined candidates for development process, technology we could use to produce the application, and the customer resources available to us. The team concluded that Scrum would be our development process. The development of the kind of software we were tasked to do was unknown to both us and the customer, so the frequent feedback and agile structure of Scrum would be essential to be able to deliever the best possible solution to the customer.

While the customer was not familiar with this kind of development, they are experts in their respective fields. In addition to providing the team with sonar data from real locations and 3D models of aquacultural equipment, they were aware of what they wanted out of the product, and thus giving the team useful and precise feedback.

The team explored a range of different technologies that could be used or incorporated in the final product. We had to map out the combinations of the different technologies, and find out which would work well together. In the end we had two candidate combinations for the customer, and used the one they requested to be used.

Chapter 3

Project Plan

This chapter covers decisions related to project management, including team organisation, schedule and work plan, risk management and some management rules.

The team organisation is described first in this chapter. The team members are fitted into a flat team organisation structure, where everyone has equal authority but different roles with particular responsibilities.

Based on the pre-study, Scrum is chosen as the development model. The team organisation and the schedule are customised to fit into the Scrum framework. An overall work plan is made where phases, milestones and time allocation is defined. Scrum, as an agile method does not emphasise all documentation and planning done before the development starts [27, 67], therefore a detailed work plan is not necessary. Sprint backlogs will serve as the concrete plan of development which will be made before each sprint and planning will be done by the beginning of each sprint and on the Scrum meeting each day. The second section of this chapter contains the schedule. This part will be updated along with the Scrum process and the chosen backlog items.

The predictable risks were identified in the planning phase. The third section of this chapter will cover a group of the risks with highest importance and/or probability to occur. The corresponding risk control plan is also presented.

Finally, some management rules, like templates and standards are documented, which will be a guideline for the team during development work and the project management.

3.1 Team Organisation

We believe in a relatively flat structure where all team members are forced to take responsibility for the whole project to succeed. Each team member is given certain roles with certain responsibilities to ensure all aspects of the project is maintained.

The group was free to choose between agile and waterfall life cycle model for the project. Our group chose Scrum (agile) as the framework for this project. Section 2.1 reasons for our choice.

3.1.1 Organisation chart

The team is organised in a relatively flat structure, where everyone has certain responsibilities. Figure 3.1 shows our organisation chart.

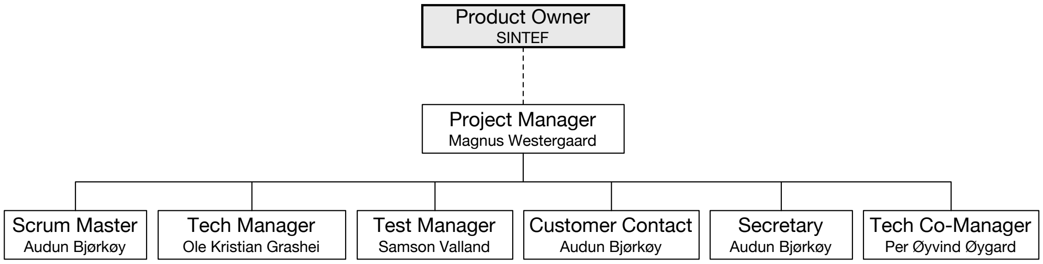


Figure 3.1: Team Organisation chart

3.1.2 Product Owner

Our main contact at SINTEF is Finn Olav Bjørnson. He is responsible for prioritising the tasks the team is working on. The product owner has an important role when deciding what items to be placed in the backlog and the importance of them [13].

3.1.3 The Team

Each team member is responsible for the work he has approved on doing. As mentioned, the team itself has a relatively flat structure, but each member is given certain roles on which he is responsible for doing. The team members have agreed on being dynamic and flexible when working on the project.

Scrum Master The Scrum master is not the team leader, but is supposed to make sure that the team is using the Scrum framework in the most effective way. He is responsible for identifying the initial backlog, leading the Scrum meetings and follow the progress of each increment and each team member [26]. A good Scrum master is capable of arranging short and focused Scrum meetings. **Audun Bjørkøy** is the Scrum master of our team.

Project Manager is responsible for the over all progress of the work, and ensures that the group maintains the goals and spirit they have agreed upon. The project manager of the team is: **Magnus Westergaard**.

Customer contact will be the link between SINTEF and the team. He shall ensure that the customer always feel taken care of and always be available to the customer if there should arise any questions. **Audun Bjørkøy** is the customer contact of the team.

Technical Manager has major technical experience in working with system development and architecture. He is responsible for making decisions on architecture, 3D engine and coordinating the use of different systems and tools to be used throughout the project. The system architect has an especially important role in the implementation phase. The technical manager in the team is: **Ole Kristian Grashei**.

Assistant Technical Manager will co-operate with the technical manager in important design matters and technical decisions. The assistant technical manager has wide experience in C++ and programming. The technical manager on our team is **Per Øyvind Øygard**.

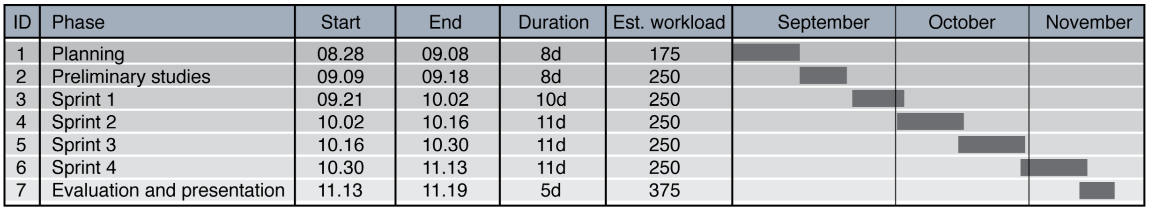
Secretary is responsible for taking minutes from each meeting within the group, as well as each meeting with the supervisor and the customer. He will ensure that all minutes are sent to the involved parties within the deadline defined in QA. The secretary is also given the main responsibility for the reports and other documentation to be developed during the project. For this job, the team has chosen **Audun Bjørkøy**.

Test Manager is responsible for developing and executing tests on the system as well as documenting the results. He will develop a test plan designed to verify the requirements in the requirements specification. **Samson Valland** is responsible for all the testing in our group.

3.2 Schedule

A typical agile Scrum project starts off with planning and preliminary studies. When all pre-planning is finished, the implementation is done in iterations called sprints. The team first agreed on using three sprints of 3+3+2 weeks, but quickly changed this to four sprints of two weeks after consulting the customer and the supervisors. By having four sprints instead of three, the group was able to have closer contact with the customer and prevent misunderstandings earlier.

There are seven stages of this project; Planning, preliminary studies, sprint 1-4 and evaluation and presentation. Figure 3.2 shows the gantt diagram.



ID	Phase	Start	End	Duration	Est. workload	September	October	November
1	Planning	08.28	09.08	8d	175			
2	Preliminary studies	09.09	09.18	8d	250			
3	Sprint 1	09.21	10.02	10d	250			
4	Sprint 2	10.02	10.16	11d	250			
5	Sprint 3	10.16	10.30	11d	250			
6	Sprint 4	10.30	11.13	11d	250			
7	Evaluation and presentation	11.13	11.19	5d	375			

Figure 3.2: Gantt diagram

Planning: During this phase the team will get a good idea of what the customer and the product is all about. After attending the group dynamics seminar, roles will be divided among the team members and responsibilities set. The structure of the report will be set up.

Preliminary Studies: The goal of this phase is a good understanding of the customer, the product and its requirements. The team must explore the problem and solution space, and come to conclusions on how they are going to define and solve the problem.

Sprint 1-4: These phases are the execution of the tasks settled in the previous phase. The customer prioritises what tasks are to be done, and the team goes through design, implementation and testing in each of the sprints. Each sprint has a goal which is evaluated at the end of each one.

Evaluation and Presentation: During this phase all the functionality exists, and the only thing done to the code is bug removal and testing. The customer, the team, the process and the course are all evaluated. The instructions for the end user are produced, and the report is finalised. The presentation is prepared. The final presentation of the product will be held at the University in front of sensors, supervisors, students and other interested people that wants to see the presentation.

With the phases of the project sorted out, a schedule for the project was extracted. The schedule contains important dates and a brief description of what to be done and when it should be finished. As we chose to use an agile framework like Scrum in our project, the project plan and product backlog was evaluated for each sprint and new items where chosen. The schedule was continuously updated during the project to reflect the selected work for each sprint in the project.

3.2.1 Planning and preliminary studies

Scheduled for 21.09.2009

Requirement specification ready From the initial project description, found in appendix A, the group formed a requirement specification. The complete requirement specification can be found in section 4.3. The requirement specification was created to help the team create a product backlog for the project.

Product backlog ready for sprint planning meeting The product backlog was extracted from the requirement specification. The requirement specification and product backlog was reviewed at the first customer meeting with SINTEF at Brattøra on September 10th 2009. The initial product backlog can be found in Appendix D.1.

Basic software architecture designed The complexity of the software to be implemented is high and a lot of features are requested. A good software architecture is needed to easily expand the software as the project develops. It is also preferable to the customer as it makes it easy to develop new features after the project is finished. Chapter 5 is dedicated to explaining the software architecture.

First chapters ready for evaluation Before starting on the sprints, most of the first chapter of the report had to written. The introduction, project directive and pre-study was the chapters we focused on.

3.2.2 Initial delivery of report

Scheduled for 28.09.2009

Edit and finish up the structure and early chapters According to feedback from the supervisor, the first chapters where edited and restructured. Some new chapters had to be added before we delivered an early report for review by the sensor.

Invite external sensor Together with the early report, an invitation to the final presentation was delivered to the sensor.

3.2.3 Sprint 1

Scheduled for 05.10.2009

Basic GUI application running Get a small core application running on which the team can continue to build. The GUI does not need to be complete, but the software architecture must be implemented and OGRE must run inside a Qt-window.

Map plugin support Implement plugin support to be able to add compatibility with different map formats in the future.

OLEX map plugin Create a plugin to load OLEX data, a specific map data format [34], into the application and construct a surface.

Basic navigation in the 3D space Implement a navigation scheme similar to that of major CAD programs use. This means being able to pan, rotate and zoom the camera.

3.2.4 Sprint 2

Scheduled for 19.10.2009

GUI design prototype finished Make sketches of the overall design of the user interface. What tools to include, where to place them and how to group them. Make a prototype, with all the GUI elements the system should have, regardless of missing implementation of functionality.

Place objects in 3D space Let the user place an object in the 3D scene. Emphasis on the ability to see the exact coordinates for the object to get the correct placement.

Select, move and rotate objects in the 3D space Let the user manipulate the rotation and position of already placed objects in the 3D scene.

Possibility to save project implemented Let the user save the state of the project to disk for later use

3.2.5 Sprint 3

Scheduled for 30.10.2009

Possibility to save project continued As this feature was not finished in the second sprint, the group continued to work on this in the third sprint.

Collision detection/place object implemented Let the user place an object in preferred surface according to object metadata. For example, objects that has a specification in the metadata to be located at the seabed, should stick to the seabed when placing and moving them.

Reference compass implemented A compass rose telling the orientation of the camera to help navigate in the 3D scene.

Support for metadata Each object should have metadata related to it. The user must be able to edit and add user defined metadata both to the object blueprints and each single object.

Simple cost estimation Based on metadata, the software should be able to calculate the total cost of the equipment placed in the 3D scene.

3.2.6 Sprint 4

Scheduled for 13.11.2009

Import object via menu Let the user import custom 3D objects into the application, like an anchor.

Connecting different objects Ability to connect objects with chains. For example connecting an anchor on the seabed to a fish net on the surface with an anchor line.

Basic physics to anchor lines Make the anchor line deform according to physical laws to get more accurate length estimates.

Map colour plugin A plugin to let the user choose what colours the seabed should have. The user must have the options to chose gradient colour or steps (e.g. different colour for each 10 height meter).

3.2.7 Application functionality done

Scheduled for 10.11.2009

Application ready for final testing and polish The application functionality must be finished and the team should focus on finalising the application for delivery.

3.2.8 Report done

Scheduled for 18.11.2009

Project evaluated The project must be evaluated and all experiences should be documented in the report.

User instructions document complete The user documentation gives details on installation and starting the AC3DS software. In addition it will give basic instructions on how to use and navigate in the program. The user documentation is located in chapter 12.

Report ready for delivery The report will be delivered on the presentation November 19th. It should be completed in good time before this, so the team can focus on the presentation.

3.2.9 Presentation

Scheduled for 19.11.2009

Presentation prepared The presentation must be finished and all team members has to know what their role during the presentation is. A well prepared Power Point presentation must be ready together with a nice working demo of the final product.

Product ready The final product must be ready to work in the demonstration, as well as copies of the software on CDs to hand over to the customer.

3.3 Milestones

A milestone is used to mark or define critical points in completion of the scheduled work for the project [28]. Milestones are efficient when it comes to verifying that the project is on track and on schedule.

We have chosen four milestones for our project. Each are listen in its own table below.

Table 3.1: Milestones

Milestone 1 - First Sprint Demo	
Goal:	The goal of the first sprint is to have an application running, able to load map data and do basic navigation in the map. Most of this functionality should be finished during the first sprint. If this milestone is completed in time, we consider the project to be on time.
Quality measures:	The code finished for this milestone is allowed to have some bugs in the code, and the GUI does not have to look as nice as the final presentation of the product. The important thing is to have a working software with correct software architecture that easily can be expanded further during the project.
Target date:	The sprint demo of the first sprint is held at october 6th. This is also the target date of this milestone.

Milestone 2 - Final Delivery of Product	
Goal:	By the second milestone, the software should be considered finished by the group. The software should be able to load a map, place elements in the water, on the seabed or on the shore. The user must be able to rotate and move these elements, and navigate in the 3D space by turning and panning. The software should also have a nice looking GUI, making navigation and placing of elements easy. Saving configurations for later use, is also expected.
Quality measures:	Given the short implementation time, the software is not expected to be 100% free of bugs, but the software must be able to make basic tasks without crashing. To confirm this, the group has developed a test-plan to ensure the functionality working as it should.
Target date:	The second milestone is scheduled to be finished on the 15th of november.

Milestone 3 - Final Delivery of Report	
Goal:	The report must be finished including all important parts. Appendix must also be complete with all necessary parts.
Quality measures:	Spelling and grammar of the report must be flawless and the design and structure of the document has to be nice and consistent.
Target date:	The third milestone is scheduled to be finished on the 18th of november.

Milestone 4 - Final Presentation	
Goal:	The final milestone marks the end of our project. The final presentation includes a nice looking Power Point presentation and a demonstration of the software. All team members will talk about their responsibilities and the results.
Quality measures:	The whole presentation must be held without technical problems. The demo must work without bugs, and all team members have to be well prepared.
Target date:	The final presentation is held at November 19th.

3.4 Risk Management

In all projects, there are a lot of potential risks. This section will highlight some of the risks that we consider most probably to occur during the project. We have divided the consequence and probability of each risk into three groups - Low, Medium and High[3]. The properties are pretty self-explanatory, the risks with high consequences are most critically for the project if they occur, and the ones with high probability is most likely to occur. A high consequence/high probability-risk is most unwanted. The whole risk table is placed in appendix C.

3.4.1 #1 - Inadequate skills in the selected tools (H/M)

Description: Among the largest risks of this project is the teams lack of experience with the selected tools and programming language. The outmost consequence of this might be software not working at all or software containing a lot of bugs and bad structured code.

Solution: To reduce this problem, the group will focus on sitting together when coding, so that the more experienced members will be able to help the other team members along the way.

3.4.2 #3 - Punctuality of team members (L/H)

Description: The team has agreed on meeting 9 in the morning every day. Making team members show up in time, is something that proves to be difficult.

Solution: The team has agreed upon a punishment system to reduce this. We keep track of when people are running late, and the person late are given alcohol point according to how late he is. One alcohol point equals one beer. 6 points can be exchanged into one bottle of wine. The late team member gets one alcohol point for every 5 minutes he is late for the first 30 minutes. Then one alcohol point for every 15 minutes after that.

3.4.3 #4 - UKA (M/H)

Description: A risk that is highly possible to affect the work in some way is the upcoming social event, UKA. Samson is working on the graphical design for UKA, and has several deadlines during the project. In addition, we have to take into account that all team members will attend at a few concerts and happenings during the three weeks UKA is in progress.

Solution: The best way to prevent this for affecting our project noticeably is to make good agreements on when we are going to work.

3.4.4 #8 - Estimation of work hours (H/M)

Description: There is a relatively big chance of underestimation of the time it will take to implement our solution. The result of this might be that the product is not finished on schedule and a lot of the last work will be done in a hurry.

Solution: To prevent this from happening, the group will break each task down to as small elements as possible. This will make time estimation more accurate.

3.5 Management Rules

This section gives a set of rules and standards that the group has agreed upon. Following these rules will help all team members knowing where to find information.

3.5.1 Templates

All documents should be written in latex using predefined templates. This is done to ensure a consistent look, and to ease document creation.

Weekly report This template describes the setup of the status report that the group delievers to the supervisors every week. It contains a summary of what has happened the last week, a more detailed description of the tasks planned that were completed, and a plan for tasks to be done the next week. See appendix section B.5.

Supervisor meeting agenda This template describes the setup of the agenda for the weekly supervisor meetings. See appendix section B.3.

Supervisor meeting minutes This template describes the setup of the meeting minutes that are produced after the weekly supervisor meetings. See appendix section B.4.

Customer meeting agenda This template describes the setup of the agenda for the biweekly customer meetings. See appendix section B.1.

Customer meeting minutes This template describes the setup of the meeting minutes that are produced after the biweekly customer meetings. See appendix section B.2.

3.5.2 Standards

- File Organisation All documents and code produced should be stored in the project's SVN repository. This lets several users edit documents simultaneously and ensures a minimum loss of data if anything should happen to any of the individual computers used.

  The repository has two main folders. One contains all the documents and the report. The report consists of one master document which includes all the documents that together make up the entire report. Having one or more documents per chapter keeps them at a managable size. The other main folder contains all the source code for the actual application that is produced over the course of the project.

- File naming

  All files should be lowercase and dash-separated. All time-specific documents should include dates, and in case of agendas and minutes, who the meeting was with.

  - YYMMDD-customer-meeting
  - YYMMDD-customer-meeting-agenda
  - YYMMDD-supervisor-meeting
  - YYMMDD-supervisor-meeting-agenda
  - YYMMDD-weekly-status-report
- Coding style Coding style and standards are defined by the customer. It is based on Brett Slocum's C++ Coding Standard [31].

3.5.3 Version Control Procedures

All the files, including notes, report documents and source code, generated during the course of this project will be placed in a shared repository using SVN. This will facilitate version control, easy sharing and automatic backup of the everything made by the group members. The repository will be structured in a logical way, separating finished documents from work in progress and static assets from dynamic files such as source code. This ensures that files are easy to find and categorised with related items.

When using the repository every member is responsible for:

- always updating before editing
- avoid locking down a file for excessive amounts of time without committing changes at decent intervals
- making informative comments on what an update consists of with every committed update

3.5.4 Meetings

Internal group meetings: The group will have at least one internal meeting every week, usually following the supervisor or customer meeting. Therefore the time of these meetings will vary. These meetings will be used to clarify and delegate work to adjust to the feedback received from the customer and the supervisor. Meeting minutes form these meetings will only be taken when necessary.

Supervisor meetings: The weekly meeting with the supervisor and the assistant supervisor is scheduled to 2 PM on Tuesday. Before noon on Monday a status report and an agenda must be delivered. The status report should state what the group has been doing the previous week, what the main problems have been and what the group plans to do the following week.

Customer meetings: Meetings with the customer are not scheduled to a specific time for every week. A few days after a customer meeting the group decides when a new meeting is needed and requests a time that suits the customer. All meetings take place at SINTEF Aquaculture and fisheries.

Scrum meeting: Each day the group is supposed to work together, the day starts with a Scrum meeting. This object of this meeting is described in detail in subsection 2.1.2. This is a meeting for the team members to give a short status on what they have been doing and what they will do for the next hours.

3.5.5 Meeting minutes

Notes are taken by a designated secretary during every meeting. For the supervisor and customer meetings minutes are made within one working day, which are then sent to either the supervisor, the customer, or both.

3.6 Chapter summary

The team quickly delegated roles and responsibilities. Making these known to all the team members made it easy to assign tasks and follow up on their progress. It also helped the communcation within the group, as everyone knew to whom any inquiry should be directed.

Once the assignment had been clarified and discussed with the customer, we set up a schedule for the entire timeline of the project. This gave us an idea of how much time we had to complete the different phases of the project, and settled important dates at which we knew we would have to deliever or present our work. The milestones gave us specific goals to work towards and served as both a measure for how far along the project should be at points in time as well as motivating the team.

A risk analysis highlighted the most relevant risks that might affect the progress and completion of the project. In addition to raising the team's awareness of these risks, it also produced countermeasures to minimise their impact.

Finally, the management rules were set. They dictate how communications with the customer and supervisor should happen, and templates for the most common documents produced before and after meetings were made.

Chapter 4

Requirement Engineering

This chapter of the report contains the requirements gathered for the AC3DS project. A lot of the requirements will be updated or modified during the project, as the team has chosen to use the Scrum framework [13]. According to changes made, this part will be updated during the project to contain the correct specifications for the final product.

The purpose of the requirements engineering part is to give a short description of the requirement process, and define the final requirements given by the customer, SINTEF Fishery and Aquaculture, in collaboration with the team. The product backlog is used to clarify the most important parts of the product, and will help the team work on the most important stories in each sprint.

The first section in this chapter will cover how a general requirement process is done and how we have chosen to do our requirement process. In short, the group has focused on good communication with the customer because the project has a tight time schedule that does not allow a lot of analysis and numerous meetings.

Secondly, the textual use cases developed in the requirements engineering process is introduced. These use cases are not only made to get a better understanding of the final product. They also make testing of functionality easier later on in the project.

Thirdly, the initial requirements specification developed together with the customer before we in the last chapter introduce the final product backlog.

4.1 Requirement Process

4.1.1 General Process

The requirement process is a process used to discover, analyse and validate system requirements for the whole system. The processes used for requirement engineering can vary widely depending on the people involved, the organisation developing the requirements and what the application domain is[1]. However, there are some activities that are common to most processes:

- Requirements elicitation
- Requirements analysis
- Requirements validation
- Requirements management

These activities involve the developing team working with the customer to define the application domain, what services the system should provide and the operational constraints of the system. The activities may as well involve stakeholders, such as end-users, domain experts, engineers involved in maintenance etc.

The team developing the requirements has to be aware of the many problems that may arise, such as conflicting requirements from different stakeholders and stakeholders stating their requirements in their own terms.

4.1.2 Requirement Process in AC3DS

The requirement process is mainly based on the project description described in appendix A and meetings with the customer. In addition, some uncleared cases was solved over the phone or by e-mail. Table 4.1 shows the steps our team has taken to gather the system requirements.

The first customer meeting found place the same day as the project started. The whole group was given a short introduction by the product owner, Finn Olav Bjørnson at SINTEF. He was very open for suggestions on the software, but meant the software was to be used as demo purposes in their lab, and therefore should look nice

Table 4.1: Our requirements engineering process

Domain understanding	How the organisation operates and what the software is supposed to do/-solve. This was done through study of the original project description, other written material provided by SINTEF and through meetings with the customer.
Requirements collection	Collection of all possible requirements for the project. Mainly done by studying the project description and through meetings with the customer.
Conflict Resolution	Finding requirements that would not go nicely together and find a solution to them. This was done by internal meetings in the team.
Prioritisation	Find the most important requirements in the project, helping the team to focus on the right things. This was done together with the customer, first for the whole project, then for each sprint.
Requirements checking	Checking the completeness and consistency of the requirements. The group did this in internal meetings.

instead of being rich on features. From this description, the team had some thoughts on how the final product should look like.

When visiting SINTEF on our second customer meeting, it soon came clear that the customer wanted something else than first stated. This meant that the team had to reconsider our previous thoughts about the program. A lot of wanted functionality were given, and the process of defining the functionality and requirements continued before a third customer meeting.

Final agreement on the requirements and prioritising of the stories in the backlog was done on the fourth customer meeting, which also was a sprint-meeting and the start of our first sprint. The final requirements specifications can be found in chapter 4.3

Some scenarios represented by textual use cases where developed in order to communicate better with the customer and give the team better understanding of how the software was supposed to perform. The textual use cases can be found in chapter 4.2.

4.2 Textual Use Cases

The group developed textual use cases for the most important user scenarios. These use cases are easy to make tests for, and they put the user interactions in more conrete terms. The textual use cases can be found in Appendix F.

Table 4.2: Use case 1: Starting a new project

Use case name	**Start a new project**
Backlog item #	1, 11, 18
Actors	User, system
Flow of events	**1** The user launches the application **2** The application shows options for loading an earlier project or starting a new project. **3** The user chooses to start a new project. **4** The application shows available map areas. **5** The users chooses map area. **6** The application loads the map data into the 3D-window and the tools.
Extensions	**1.1** The user launches the application **3.1** The application shows options for loading an earlier project or starting a new project.
Entry condition	The application is running
Exit condition	The user closes the application, or the project is initiated with or without map data

Table 4.3: Use case 4: Navigating the 3D scene

Use case name	**Navigate the 3D scene**
Backlog item #	2
Actors	User, system
Flow of events	**1** The user moves the mouse cursor into the 3D scene **2** The user holds down one of the ciew manipulation buttons **3** The user moves the cursor **4** The view of the scene changes
Extensions	**2.1** The user holds down ctrl **2.2** The user holds down alt **2.3** The user holds down shift **4.1** The view is zoomed in **4.2** The view is rotated **4.3** The view is translated sideways in relation to the current viewing direction
Entry condition	The application is running and a project with map data is open
Exit condition	N/A

Table 4.4: Use case 2: Placing 3D object in scene

Use case name	**Place imported 3D object in scene**
Backlog item #	7
Actors	User, system
Flow of events	**1** The user chooses a 3D object from the 3D-object list. **2** When the user moves the cursor into the 3D-window the object follows it and snaps to the reasonable grid for the object to be placed in. **3** The user clicks the mouse button to place the object in the preferred place. **4** The object stays there, and will stop following the cursor.
Extensions	**3.1** The user gets rid of the object at the cursor by hitting esc
Entry condition	The application is running and a project with map data is open
Exit condition	The user hits esc or closes the application

4.3 Requirements Specification

4.3.1 Functional requirements

A functional requirement is a definition of a specific function required in the product. This function is a certain element in the product, a component, which can perform something. This can be to take in some input, process it, and output a result. It is *what* the product can accomplish, regardless on *how*.

4.3.1.1 Core Requirements

Core requirements are the most important features to make an acceptable product. One can say they are the minimum of what is required. Many of the points are also necessary to be able to implement other(non-core) requirements. Our product does not have a long list of core requiremets, especially compared to most of the other products being made in this course(non-3D applications). It shall be an application for showing a 3D-enviroment, navigate around in it and add and manipulate objects in it. You can say this is the core of the program, and every other requirements come under that statement in this project. But we have specified the core requirements a bit more, and taken things easy to oversee into account, so the list is quite longer.

The core functional requirements are listed in Table 4.5

Table 4.5: Core Functional Requirements

ID	**Description**	**Priority**
CFR01	Show a 3D model in a window	High
CFR02	A menu next to the 3D window which gives access to different objects and information	High
CFR03	A toolbar at the top where the user can choose different tools and commands	High
CFR04	Generate 1:1 model of the seabed, surface and coastline from map data	High
CFR05	Ability to choose an object and place it in the 3D-environment	High
CFR06	Objects already in the 3D environment can be selected, moved, rotated and deleted	High
CFR07	Objects can be imported from a datafile which describes the model	High
CFR08	Ability to save the current environment with the state/setup for the moment	High
CFR09	Cost estimation of the current setup	Medium
CFR10	Object can join in a logical manner, anchor lines can be snapped between reasonable points on other objects	Medium

4.3.1.2 Desirable Functional Requirements

Desirable functional requirements are the requirements that would be nice to have implemented, but probably does not. It is the requirements that get low priority, and if there is time left for implementation, we will implement these. The desirable requirements are listed in Table 4.6.

Table 4.6: Desirable Functional Requirements

ID	**Description**	**Priority**
DFR01	Multiple viewports for more detailed placing of objects	Medium
DFR02	Support for modules for different simulations of fish cages, oxygen level, etc.	Low
DFR03	Transformation between different coordination types	Medium
DFR04	Support for modules for streaming of live sensor data	Low
DFR05	Simulation of tidewater	Low
DFR06	Support for multiple model- and map data	Medium

4.3.2 Non-functional requirements

Non-functional requirements are the requirements that don't goes directly on the product's functionality, but more on *how* the functionality is implemented. It can be everything from response time to arrangement of buttons. The non-functional requirements are listed in Table 4.7.

Table 4.7: Non-Functional Requirements

ID	Description	Priority
NFR01	Language used: C++	Medium
NFR02	Good response time	High
NFR03	Acceptable framerate	Medium
NFR04	Short start-up time	High
NFR05	Handling of large map data and large project	High
NFR06	Easy to navigate in the 3D-environment, in the same manner as commercial 3D-programs	High

4.4 Product Backlog

The stories in the product backlog corresponds to the items defined in chapter 4.3 Requirements. It is important to notice that the customer wanted a program with a lot of functionality. The backlog contains what we believed the team was able to do as a absolute maximum during the project. Most of the desirable requirements stated above are therefore left out of the backlog. Even though these requirements are not part of the backlog, we have designed our architecture to be very modifiable through the use of what we call plugins. This will make it possible for the customer to easily continue expanding the functionality of the software after the project is done.

To find out how much time we would use on each item in the backlog, we gathered the whole team, and each team member individually gave their estimation of each item. After some discussion and explanation of the estimations, an average was calculated and used as time estimation of each backlog item.

Estimation of the time for each backlog item is represented as story points. One story point roughly equals one person hour. It is important to notice that after each sprint we evaluate the time estimation, to find out how good our estimation was. The value of story points could then be adjusted to give an better estimation of the time. The important thing is not to get the absolute estimation correct (i.e. that a 2-point story should take 2 hours), but to get the estimates correct (i.e. that a 2-point story should require about half as much work as a 4-point story) [13].

The complete initial and final product backlog is located in Appendix D. Table 4.8 contains descriptions of all the items in the inital backlog.

4.5 Chapter summary

The process of gathering requirements for the product started with the group having several meetings with the customer over the first two weeks of the project. The assignment was clarified, and the team got a better understanding of what the customer wanted. This included gathering the non-functional requirements that the customer had. The next step was producing some use cases for the core mechanics of the final application. From these a rough requirement specification was made and discussed with the customer. After evaluating it, we made the initial product backlog that would be used and evolve over the course of the project.

Table 4.8: Initial product backlog

ID	Name	Description
1	Basic GUI app running	Have a running prototype. Should include a 3D viewport integrated with other GUI elements (toolbar, basic menu).
2	Navigate 3D space	Let the user navigate the 3D space in the viewport of the application. Should use controls similar to those of existing 3D applications (either game-like controls or 3ds Max-like controls)
3	Import object	Let the user import models of equipment and use them in the application.
4	Map plugin support	Have the application support extensions that load map data from different formats and transforms them into something the 3D engine can display.
5	OLEX map plugin	A plugin specific to the OLEX data format. Allows the user to generate map data from a file of this format.
6	Map colour plugin	A plugin that allows the user to change the colour of the already imported map. Different colour modes should be available, for instance a gradient from one colour to another.
7	Place object in 3D space	Let the user select an object and place it in the viewport by using the mouse.
8	Select and move/rotate objects in 3D space	Let the user manipulate the objects already placed in the viewport. Translation and rotation of objects through intuitive tools.
9	Cost estimation	Give the user information on the current setup of equipment. Total cost, quantity of different materials etc.
10	Connecting different objects	Let the user connect equipment. Connections can be logical (a child object will follow any change in position in its parent object) or actual connections made with for example anchorlines.
11	Coordinate transforms	Let the user decide if she wants to work in local a local coordinate system or the global one. XYZ values vs latitude/longitude/depth.
12	Save objects	Let the user save an object and its configuration to a file so that others might import it and use it.
13	Save project	Let the user store the entire setup of equipment to a file, so that it may be loaded back up and worked on at a later time.
14	Simulation plugin support	Implement support for plugins that will simulate the physical environment and affect the objects (weather, current etc.)
15	Streaming data plugin	Implement support for plugins that stream data about the environment to the application and manipulates objects accordingly.
16	Edit object data	Let the user add, edit and remove metadata for the objects in the application.
17	Measure distances	Let the user define two points and return the distance between them.
18	Visually select the map area	Show a 2D map in which the user can select and area. Data on the selected area is retrieved and 3D geometry generated from it.
19	Terrain above surface	Also generate the terrain that is above the surface in the current area.
20	Basic physics applied to anchorlines	The anchorlines that connect equipment on the surface with anchors at the seabed are deformed by forces, which have to be taken into account when their length is determined. Some simple model should simulate this to allow for better cost estimation.

Chapter 5

Software Architecture

In this chapter we will give an overview of our pre sprint 1 design decisions. Concrete design and implementation of features will be done in each sprint. The design decisions described in this chapter are overall system design guidelines on how modules should be organised to create a feasible system. Selected technologies and their position and interaction with other modules will be defined.

5.1 Core architecture

A well defined core architecture is an important part of any software development project. The architecture will need to suit the requirements the customer has for the product, while at the same time making it easy for the team to develop the application. The key is to avoid reinventing the wheel, but instead adapt well known and widely used design patterns that fit the project.

5.1.1 Essential features

The customer has brought forth a number of features that the architecture should facilitate. This section identifies them and how they affect the basic architectural design.

5.1.1.1 Plugin support

Support for plugin libraries and programs are wanted to allow the functionality of the application to be extended easily in the future. This will require well defined interfaces and permissions to let the plugins co-operate with the the host application, while at the same time keeping the host safe from both malicious code and faulty plugin programs that might crash. Some sort of manager must take care of the administrative tasks such as loading the plugins dynamically and giving them access to whatever they have permission to. This manager will use something similiar to a client-server pattern, so that the plugins (the clients) have to register with the plugin manager (the server), and operate independent of each other.

5.1.1.2 Database and file support

One of the features requested by the customer is the ability to save application data both to file and database. This means the application data at runtime must be organised in a logical manner, keeping parameters and other object data for some object represented in the application tied to that object's data object. This will allow the application to, for instance, get all the data necessary to save an object's state easily. Having the data organised like this is the whole foundation of object-oriented programming, and is implicitly done throughout the development.

5.1.1.3 An interchangable GUI

The customer might want to change the GUI part of the application, or have the application interface with some other application through the same interface as the GUI uses. By separating the logic of the program and the GUI and only keeping a loose connection between the two, the GUI can be swapped out for something else that conforms to the interface of the logic. The GUI will be event-driven, so that anything listening to the events fired when something is changed by the logic can present the change in whatever way it wants.

5.1.2 High Level Design

In this section a basic overall design philosophy will be presented. The group has decided to use the three layer Model-View-Controller (MVC) [25] architecture, separating the data from the logic and the logic from the presentation. This will make sure that any input will have to go through the logic layer before any data is actually changed. The aforementioned plug-in manager sits in the logic layer with a complete overview of the application's objects and logic. The data layer contains data objects that are manipulated by the logic layer, and the database and file storage functionality also lies here. At the top is the presentation layer which handles the input from the user and the graphical representation of the application state.

Figure 5.1 shows a conceptual draft of our software architecture design. We can see that our program has two main packages, AC3DS and the kernel. The kernel is the core of our application representing the model in the MVC pattern. This kernel is totally independant from any other package beeing developed in our solution. AC3DS is the main program and GUI package. Controller objects connecting the GUI to the model is found here together with the GUI itself. Qt and OGRE3D are external packages beeing used by the AC3DS package.

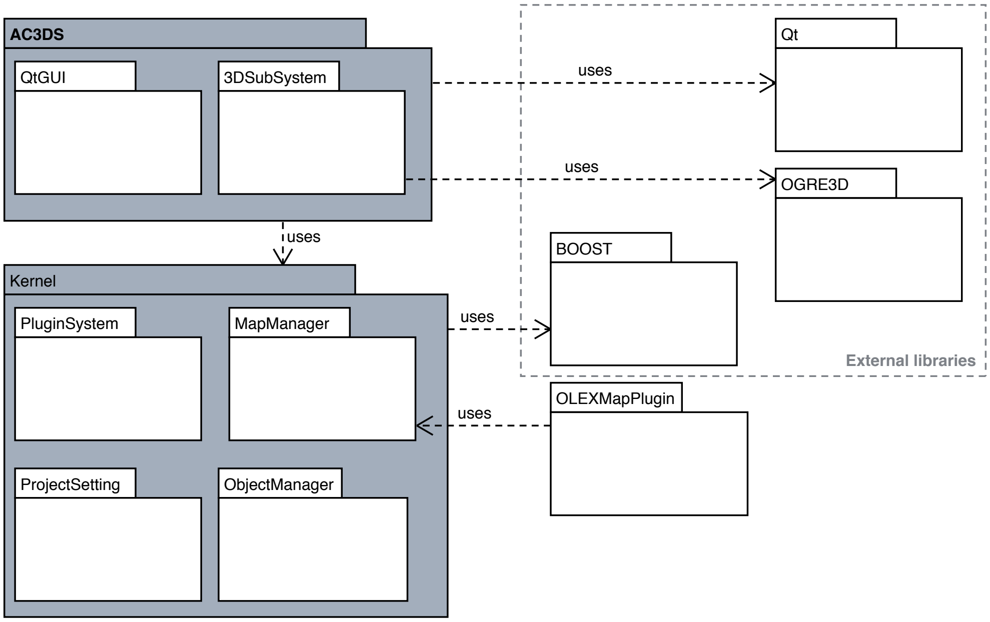


Figure 5.1: Conceptual Software Architecture Design

5.1.2.1 Model-View-Controller and our system

The model will be represented by classes that hold the data in our world and modifies it as requested by the controller. The model will use BOOST for serialisation of objects. The MVC architecture enforces a strict policy that the model should have no direct knowledge of the controller or the view. A weak connection to the controller exists via listeners. When a change occurs in an object at the model level the object will send events to all listeners registered with that object. The listeners will then act accordingly. These listeners exist within the controller layer. We have decided any object in the model layer should not know of OGRE or Qt at all to fit with the MVC paradigm. This decision will make it easier to use the model with another view if OGRE or Qt are changed with other technology. Not having access to OGRE in the model will, however, have some drawbacks. OGRE has a lot of functionality desirable in the model such as 3D-object representation and modification. Since we don't want to reinvent code that already exists, we solve this by allowing some business logic to reside in the controller layer.

The view will be our graphical user interface made with Qt and OGRE. The view will be listened to by controller objects. If an object is edited in the view, controller objects will update the model with the new data.

The controller is the layer between the view and the model. The controller will have strong knowledge of both the view and the model. Controller code will be responsible for keeping the view and model in sync. Controller objects are mainly listeners that will act on user input and they aremostly found in the AC3DS package of our design (see 5.1).

5.2 Chapter summary

The architecture was designed to meet the requirements of the customer. Most notably, it allows for the functionality to be extended through plugins. The core architecture is designed after the MVC pattern to separate logic from data and visualising. This well known design pattern has been implemented and put to the test by many others, allowing us to build on their experience. Using it facilitates the implementation of features like the save/load functionality and the interchangable user interface component.

Chapter 6

Quality Assurance

This chapter contains a set of rules and routines to make sure the quality of the process and the final product is as expected. All involved stakeholders has to agree on the QA requirements, and this chapter can be looked at as a loose contract between the team and the different involved stakeholders.

In addition, this chapter also contains the test-plan for the whole project.

6.1 QA Routines

6.1.1 Group routines

Standards: Coding standards are mentioned in subsection 3.5.2 Standards. The guidelines for naming classes and member variables have been followed by the team. The guidelines for comments have been followed to a certain degree, mostly where the name of a class or function is not enough to explain its functionality. In some special cases the coding standards have not been followed at all. This includes the code written by people outside the team and the code which is auto-generated by the editor used to design the user interface.

Review: All checked in code should be reviewed by at least one other group member.

Committing code: Before committing code, the written code should first run locally without any errors. It should also be documented as mentioned in subsection 3.5.2 Standards.

Code Conventions: All written code should preferably follow the code conventions, also mentioned in subsection 3.5.2 Standards.

6.1.2 Customer routines and response times

Customer Meetings: Meetings will be scheduled with the customer by phone, and a formal invitation with agenda sent by e-mail at least two working days in advance.

Customer Meeting Minutes: Minutes from the meeting should be mailed within a working day of the meeting.

Feedback on Meeting Minutes: Feedback on meeting minutes should be sent within two working days after the minutes are received.

6.1.3 Supervisor routines and response times

Supervisor Meetings: Meetings will usually be held at 14:00 every tuesday, with any changes in time or place to be announced within 24h before the meeting.

Supervisor Meeting Agenda: Status report, meeting minutes and meeting agenda are to be delivered within 12:00 the day before the meeting.

Feedback on Meeting Minutes: Feedback on meeting minutes are given on the first supervisor meeting after the minutes are distributed.

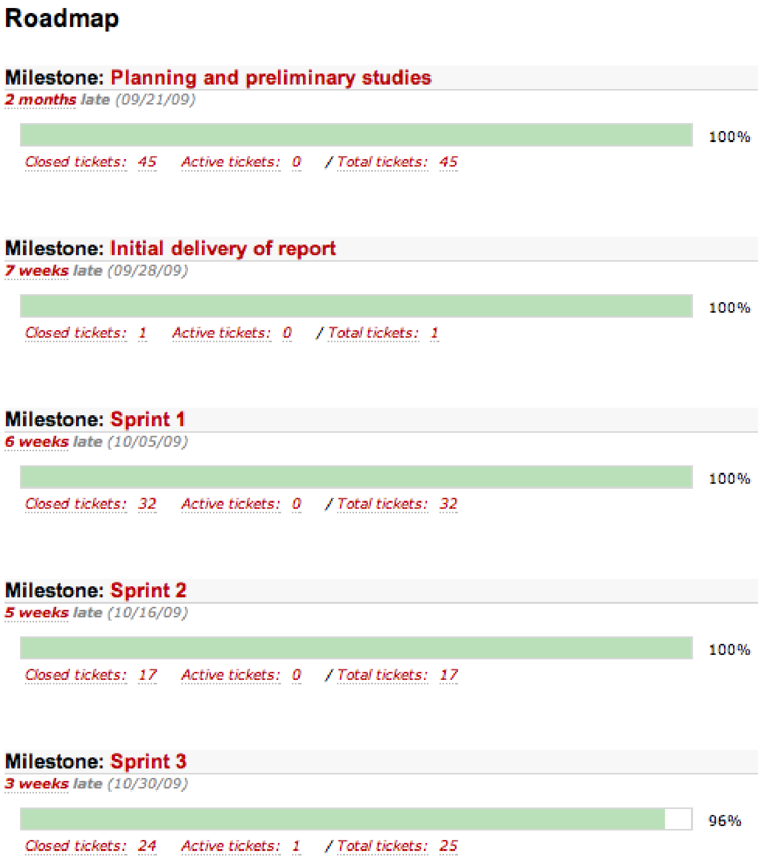
6.2 Templates for agenda and minutes

Standard templates for agendas and meeting minutes with supervisor and the customer are developed. These can be found in Appendix B.

6.3 Documentation of project progress

To keep track of tasks to be done in the project, we use an open-source system called Trac. Trac is an enhanced wiki and issue tracking system for software development projects. It uses a minimalistic approach to web-based software project management [32].

In the Trac system we have defined milestones and components of the project. The system handles all kind of tasks, from small enhancements to defects and new tasks. The system also has a roadmap-view that allows the team to have full control of the schedule of the project, telling how much work has been done and what is left. Figure 6.1 shows a screenshot of the roadmap view in the Trac.



Roadmap

Milestone: Planning and preliminary studies
2 months late (09/21/09)
100%
Closed tickets: 45 Active tickets: 0 / Total tickets: 45

Milestone: Initial delivery of report
7 weeks late (09/28/09)
100%
Closed tickets: 1 Active tickets: 0 / Total tickets: 1

Milestone: Sprint 1
6 weeks late (10/05/09)
100%
Closed tickets: 32 Active tickets: 0 / Total tickets: 32

Milestone: Sprint 2
5 weeks late (10/16/09)
100%
Closed tickets: 17 Active tickets: 0 / Total tickets: 17

Milestone: Sprint 3
3 weeks late (10/30/09)
96%
Closed tickets: 24 Active tickets: 1 / Total tickets: 25

Figure 6.1: Roadmap-view in Trac

In each milestone on the roadmap, we put all related tasks and enhancements. Each task is given a Ticket number, description, owner, type, priority and a component of the project that it relates to. We are also able to comment on each task when we have useful information for later. When developing sprint backlogs, we actively used the track system, and each sprint backlog item is identified by a trac id.

6.4 Test Plan

Testing can be one of the hardest parts in a software development project. Altogether with the debugging and re-releasing testing takes normally more time than planned. Since it also is the last thing you do in a project phase/sprint, it is very common to prioritise to do other stuff considered more important than the testing. This easily leads to buggy product delivered, and a lot of extra work on bug fixing in the following sprints. But testing is still something needed, it is impossible to not at least have an acceptance test before releasing a new product. But there is some things one can do to make this phase as short as possible: [13, 74]:

- Maximising the quality of the code delivered from the Scrum team
- Maximising the efficiency of the manual test work (i.e. find the best testers, give them the best tools, make sure they report time-wasting tasks that could be automated)

And to maximise the quality of the code, we can:

- Put testers in the Scrum team
- Do less per sprint

One team member has the role as test manager, and will have the main responsibility for making the tests and test plans. The test manager is also the one who has “the last word” before the delivery. Nothing is ready for delivery before this person says so.

6.4.1 Overall test plan

Test	Importance	Phase	Description
System Test	High	Each sprint	Overall test that the product delivered is acceptable according to the goals or backlog items chosen for this sprint.
Unit Test	Low	Each sprint	The unit test has only been done informally by each programmer. To save our limited time we have concluded that this is the best way to do it.
Integration Test	Low	Each sprint	The integration test is done under way in the sprint, to clarify that each part different people have been working on functions togheter. This test has also been carried out informally, due to save time and be able to focus on other things.

The system test for each sprint is located in section 6 in each respective sprint chapter.

6.5 Chapter summary

The team established conventions to ease the production and maintenance of the report, the weekly documents and the code. Routines regarding external communications with either the supervisors or the customer clarified when and how information should be exchanged. To keep track of the progress made throughout the project, we decided to use the trac system. After setting it up with the tasks to be performed in a phase, it generates statistics and visuals that can tell the team how they are performing in relation to the plan.

Finally, the test plan ensures the quality of the product we present to the customer. Any major flaws or bugs will be found and corrected, thus preventing them from being carried over to the next iterations of code.

Chapter 7

Sprint 1

The sprint started with the first sprint planning meeting for the project at SINTEF Brattøra on Monday, September 22nd. It was obvious for both the team and the customer that the first sprint would be focused on getting a basic application with a good software architecture up and running. It was also decided that a basic OGRE window with a map generated from OLEX data, and basic 3D navigation should be implemented.

After the meeting, the group produced a more detailed sprint backlog in an Excel spreadsheet to keep track of time. All sprint backlogs with burndown charts can be found in appendix E.

The sprint started quite well, but the groups lacking experience with C++, OGRE and Qt, soon showed us that much time was necessary for research and learning, giving less time to implement and develop the application. This left us behind schedule. The last few days of the sprint were used for intensive coding, getting a demo product finished to show the customer on the sprint demo. Figure 7.6 is a screenshot of the demo used for the presentation.

7.1 Sprint goal

The goal of the first sprint is to be able to demonstrate a plugin that generates map geometry from OLEX data, and show it rendered in an OGRE viewport which is embedded in a Qt window. This should be demonstrated for the customer in two weeks.

7.2 Sprint Planning

The first sprint meeting was conducted at SINTEF Brattøra on Monday, September 22nd. After going over some minor issues, the group presented the product backlog for the customer. Over the course of an hour, the backlog items were discussed and prioritised by the customer, with some input from the team where requested. This initial backlog can be seen in Appendix D.

The team then went on to pick out backlog items for the first sprint and discussed them further with the customer. A goal was set for the demo meeting at the end of the sprint.

The items picked out were the four top prioritised items from the product backlog:

Basic GUI app running

This item is just getting a small core running on which the team can continue to build. It includes designing the core classes, getting a 3D viewport from OGRE running inside a Qt window, adding a basic menu system using Qt GUI elements and having some basic interaction, i.e. loading a map using a GUI menu item.

Map plugin support

To visualise the seabed model in the OGRE viewport, map data has to be processed and converted into a 3D mesh which OGRE can display to the user. This is done outside the core of the application, in a map format-specific plugin module. To have the core application and these types of modules interact, some interface must be defined between them. This is the main work done in this backlog item. A mockup plugin should also be produced to do unit testing.

OLEX map plugin

Olex is a specific map data format that the customer uses. This backlog item includes making a plugin to load data in the OLEX format and make a mesh for the core application to use. In more detail, the point data in the OLEX file has to be transformed from the global lat/long coordinate system used into a local coordinate system. From these points, a surface has to be reconstructed and visualised.

Navigate 3D space

It is useful to be able to navigate the 3D scene in the application. This backlog item includes implementing a navigation scheme similar to that of major CAD programs used. This means being able to pan, rotate and zoom the camera.

7.3 Actual Process

The items mentioned in the subsections above were broken up into smaller tasks for the team members to do. See Appendix E.

The estimates shown in the sprint backlog are relative, and should not be interpreted as work hours. When evaluating the sprint, the team will have a good understanding of what one of these units is equivalent to in work hours. This will be used when picking backlog items for the next sprints.

The first few days of this sprint consisted of setting up the project's code base. This meant setting up the IDE, compiling Qt and OGRE, and making a working combination of the three available to the entire team via the shared subversion repository. Once this was done, the parts concerning plugins and navigation in 3D space could be merged with the working foundation and finished.

Basic GUI app running

The task of creating a Qt window which incorporated an OGRE viewport, was mainly done by the technical co-manager. After Qt and OGRE had been successfully compiled, a Visual Studio project was set up, integrating the OGRE window inside a Qt GUI. The GUI was given some basic menu widgets to illustrate the possibilities.

Setting up a working project with relative file paths, proved to be the hardest part, though tutorials and example applications made it easier. By the end of the sprint the basic framework for the application was done, and we were able to successfully build the demo on all team-members' computers.

Map plugin support

This part was implemented quickly and without any major problems. After researching how to compile source code into dynamically loaded libraries (DLLs) and how to have them loaded and used at runtime, we implemented the core classes and a mockup plugin quickly. An interface was then created to allow the map plugins to be integrated with the core.

OLEX map plugin

Loading and transforming the OLEX data was done by the project manager. Working on the data without having a way of visualising it was hard. There was no way to know how the coordinate points contained in the data were ordered. This meant there was no way of knowing if the points' organisation could ease the problem of recreating a surface from them. When the initial setup of the code base was done and the coordinate points could be viewed in 3D space, it became clear how hard it would be. After spending a couple of days researching existing solutions to similar problems, we came up with our own solution: A heightmap was generated from the point data. Generating the actual geometry from the heightmap was done by the technical manager.

Navigate 3D space

How the user would navigate in 3D space was discussed internally in the team. The first alternative was game-like controls, using a combination of the keyboard and the mouse to maneuver. This is common in games using a first-person view. The other alternative was one more similar to the controls used in existing 3D software, where everything is done using the mouse. Both alternatives were roughly implemented and tested, and in the end we decided to go with the second alternative.

Tuning the way navigation behaved was done by the test manager.

7.4 Software Design

In the first week of the sprint, design work was done to simplify the implementation phase of the sprint. Design of the program core with plugin support was focused upon. A plugin system is achieved by every plugin registering itself in the correct subsystem of the core. E.g. a map loader plugin will register itself as a map loader with the CMapManager class. See 7.1 for a class diagram of the core application, incorporating its plugin capabilities. A chain of responsibility pattern [8] is used by the kernel to load a map. The kernel will ask each map loader plugin if it can load the given file. If no map loader plugin can load it, an exception will be thrown. An OLEXMapPlugin is designed to allow loading of OLEX map data, see fig 7.2. This diagram shows composition and generalisation between classes in different packages.

In addition, we produced a class diagram for the first map plugin that supports the OLEX map data format. See figure 7.3. This plugin consists of a few classes that converts an OLEX file into our own CMapModel class.

7.5 Implementation

Qt and OGRE working together

Setting up the source code base and getting OGRE and Qt working together was time consuming. This was, however, necessary to the further development of the application.

The main problem at this point was the team's lack of experience with the development environment Microsoft Visual Studio. Using instructions found on the engine's website [4], we set up OGRE so that we could examine and run the example applications that came with the engine. These showcased different technical aspects of the engine such as basic simulation of physics and collision detection, rendering a realistic water surface and animating geometry.

Installing Qt was harder. It had to be compiled from source code, and could not be installed as easily as OGRE had been. This required an understanding of the parametres that need to be specified when compiling Qt. They determine things like whether it should support OpenGL, DirectX or both. The actual compilation took

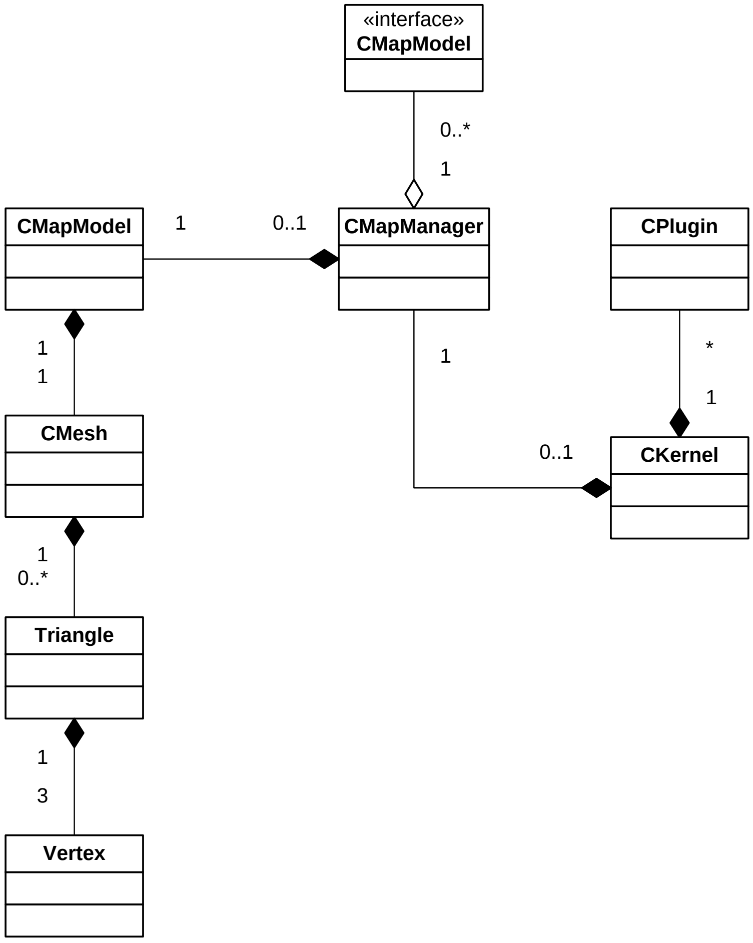


Figure 7.1: Uml class diagram for the core package in sprint 1.

somewhere between 4 and 6 hours.

When both OGRE and Qt were successfully installed, it was a matter of making them work together. A graphical user interface made in Qt consists of widgets. They can be menus, toolbars or other GUI widgets. The challenge was making OGRE render to a viewport which could be used as a widget by Qt.

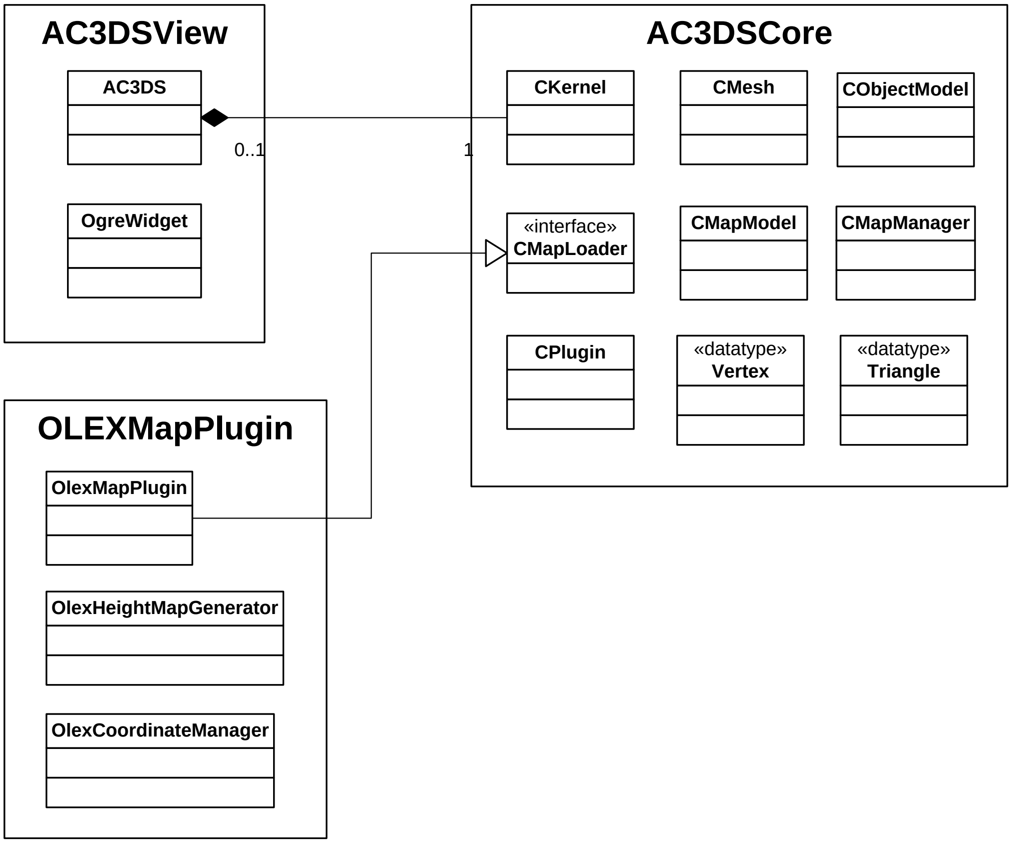


Figure 7.2: Diagram showing class composition and generalisation between packages in sprint 1.

The task was solved by implementing the OGRE rendering view as a subclass of QGLWidget, which is a Qt widget for rendering OpenGL graphics. By overriding the native functions for rendering we could easily control the content as we wanted, and still retain compatability with Qt to ease UI development.

Map plugin support

Map plugins are, like all plugins, loaded through a registerPlugin function, which bootstraps the core functionality of the plugin. In the case of the map plugins,

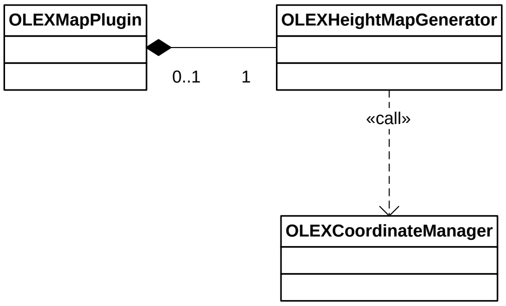


Figure 7.3: Diagram showing class breakdown of the OLEX map plugin.

they register themself through the MapManager class. All map plugins inherit from the CMapLoader interface, which is relatively simple. It consists of canLoad, which informs whether it can open the supplied file, and loadMap which takes a file path and returns a CMapModel pointer. This made it easy to implement the OLEX plugin, and any other future map plugins.

The OLEX map plugin

The OLEX map plugin was one of the more challenging tasks. After reading the test data file, which contained nearly 800 000 coordinates, and then visualising the coordinates as points in 3D space using OGRE, it became apparent that the points were unordered and unevenly distributed. This was a major concern, as reconstructing a surface from arbitrary points is a hard problem to solve. There is no single solution, and since the points are unordered, advanced algorithms have to be used. The team looked at the Ball Pivoting Algorithm [22], which, from a brief pre-study, seemed to solve the problem. However, the algorithm is patented in the US, and being unsure about the legal implications and where the application was going to be used, the team decided to abandon it.

We ended up using a solution that was not based on anything found in the pre-study of the problem. A grid made up of uniform squares is superposed onto the coordinate data. The resolution of this grid determines the resolution of the geometry that is generated in the end. All the points in the data are located within some square in this grid. By averaging the depth of all the coordinates contained within one such square, we get a good estimate of the real depth at that particular square's location. After iterating through all coordinate data points and having them affect the depth value of their square in the grid, there is no guarantee that all the squares in the grid have a value. Since the coordinate points are so unevenly distributed, there are "holes" where the distance between points is so large that they "skip" a square. To fill in the squares with no value, a simple convolution using something similar to a gaussian kernel is used. This means that any empty square gets its value from a weighted sum of its neighbours' values. See figure 7.4 for an example of this method using a 3x3 sized kernel. The result is smooth transitions where there was no depth value previously.

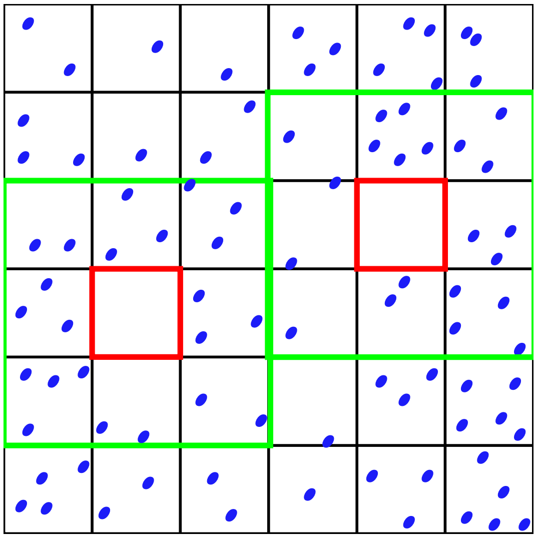


Figure 7.4: Superposed grid on coordinate data. The red squares will get their value from their neighbours in the green areas.

Based on this grid, the "heightmap", the actual geometry is generated. It now visualises the raw map data loaded from the file.

The vertices that made up the triangles in the geometry were coloured according to the depth they were at, making the shallow parts of the terrain pink and the deeper areas green. This made it easier to see contours in the terrain. See figure 7.6 in section 7.8.

7.6 Testing

System test

Even if the product is up and running, it is still in an early prototyping phase. Therefore, a thoroughly and detailed test is not considered necessary to spend time on now. We have made an overall system test based on the sprint backlog items, and checked how well they are implemented. We have also narrowed it further down to a goal statement for this sprint, which we have had in the back of our head when the testing have been done:

“The system should be able to show an OGRE 3D-window which you can navigate in and another window or palette showing some simple GUI-elements, both by using the Qt UI-framework. It shall also manage to load and generate a seabed 3D-model in the OGRE window from a file containing OLEX map-data.”

This will be the factor that determines if the sprint was a success or not. You also find the test plan based on the backlog below, with the results and comments:

Test ID - Name	T1 - Get basic GUI application up and running.
Description	This is a loose definition of a test, but its underlying structure is quite important, especially for further development of the product.
Criteria	1. The OGRE 3D window is running in a GUI window using the Qt GUI-framework. 2. The application is able to show 3D models in the 3D window.
Estimated hours	4
Actual	4
Result	Passed
Comments	As this was an early version of the product, there were very few GUI-elements, even for a prototype to be. But the underlying functionality for making it easy to add more(and design it further) is in place. The same is the OGRE 3D window, inside the Qt framework. And it works well.

Test ID - Name	T4 – Plugin support
Description	Validate correct loading of a test-plugin, and that the plugin functions as it should according to the system architecture.
Criteria	Ability to load a plugin from the running application, and afterwards be able to use the plugin's functionality.
Estimated hours	4
Actual	4
Result	Passed
Comments	None

Test ID - Name	T5 – OLEX map plugin
Description	Test if the plugin for load map data of the OLEX form works correctly, and generates a seabed with corrects metrics according to the project settings.
Criteria	1. The system loads a file with OLEX map data correctly. 2. Correct transformation from the coordinate system of the OLEX map to the systems coordinate system and metrics. 3. Visualisation of the seabed in the 3D-window.
Estimated hours	4
Actual	4
Result	Passed
Comments	The loading takes long time, some code optimization and maybe threading of the process would be nice.

Test ID - Name	T2 – Navigation in the 3D space
Description	Validate that the input from mouse and keyboard is working as wanted, and that there is no problem to navigate around the 3D space by modifying the position and orientation of the camera viewport.
Criteria	Ability to move around in the 3D space by tilting, rotating, and moving the camera viewpoint.
Estimated hours	4
Actual	4
Result	Passed
Comments	It works, but the usability can get better. For instance, it can be hard to navigate with only a trackpad, which is most common on laptops.

7.7 Results

All the four sprint backlog items were completed according to the team's understanding of them. The application now opens a window that displays the terrain geometry generated from the provided sonar data. The reading and transformation of the data is done using a plugin for that particular sonar data format, outside the core of the application. The user is able to navigate in 3D space using controls similar to that of many major 3D applications.

Having spent a lot of time setting up our code base, we now have a platform on which we can build further and extend the application's functionality. As a result, work in the coming sprints will be able to focus more on functionality, and avoid the initial overhead of having to set up an environment to work in. The team members now each other and their skills better, and should be able to delegate tasks better and work more efficiently overall.

7.8 Evaluation

The team had a quite good start according to the burndown chart in Figure 7.5. There was minor problems due to some report work being done. This shows in the burndown chart as few tasks getting done in the first half, and is somewhat misleading as the tasks were well underway.

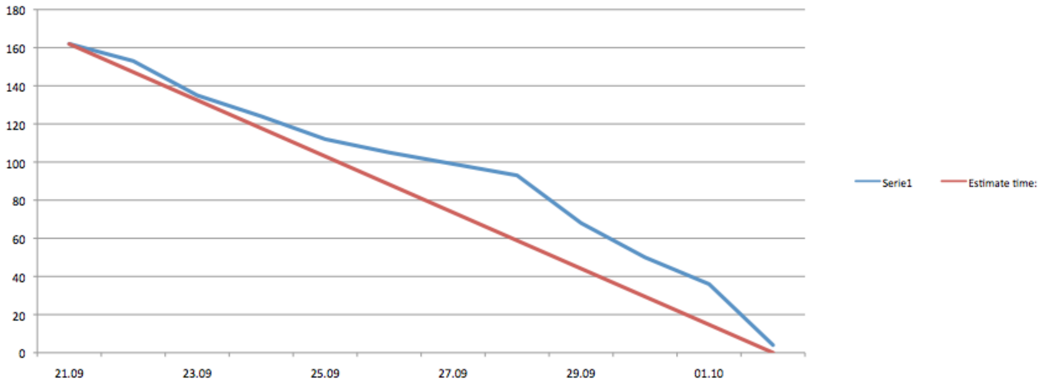


Figure 7.5: Burndown chart for first sprint

The description of the backlog items was not good enough. This was made apparent at the demo meeting, when the customer expected more from the user interface of the application than the team did. The customer and the team had different assumptions of what a basic graphical user interface meant, and the short description included did make it clear enough what the team thought this backlog item contained. As a result, a new item was added to the product backlog (to be picked in the second sprint). This new item calls for prototyping of the complete (at least what is foreseeable) user interface of the application.

The features described in the goal of the sprint were implemented. We set up an environment where OGRE and Qt work together, and a plugin lets the user load geometry from a file obtained from OLEX. The customer was content after the demonstration, and the team is happy with the results as well. Figure 7.6 shows a screenshot from our demo.

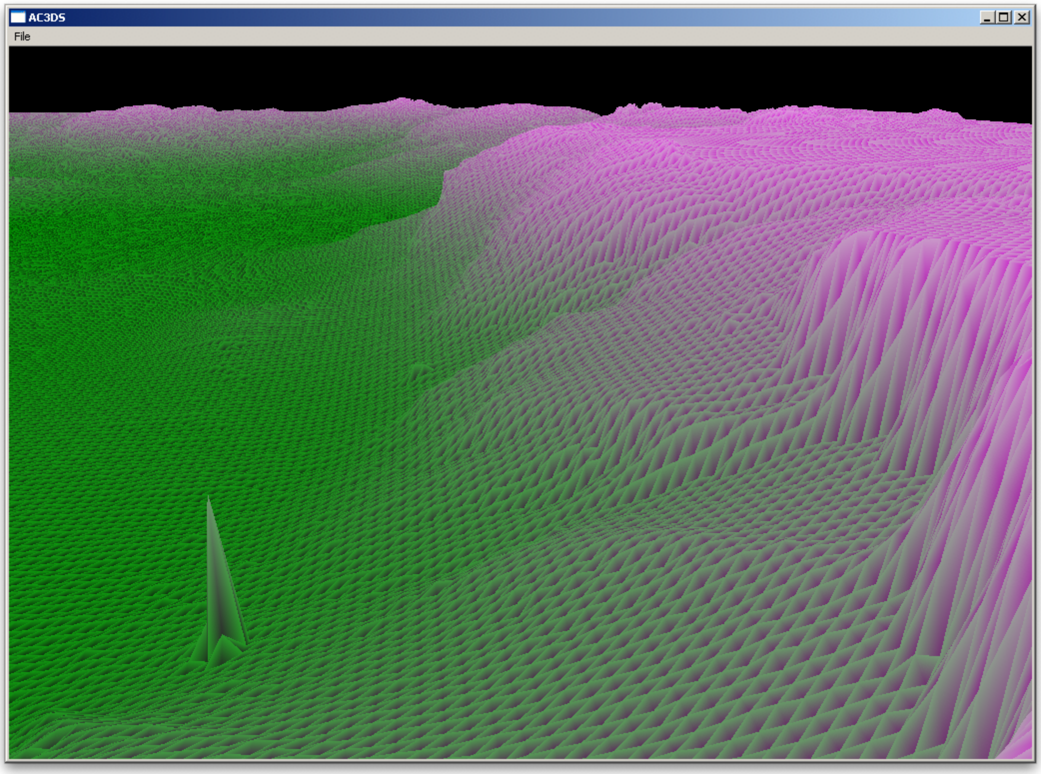


Figure 7.6: Screenshot from first demo meeting

Work on the report was not satisfactory during the last part of the sprint. The team was one man short, and most of the ones remaining worked on the implementation of the application. This resulted in little work being done on the report.

We achieved our goal, but some expectations were not met. The report suffered, giving the team extra work in the next sprint. However, the basic environment with OGRE and Qt is set up. Using this platform, we can now focus on implementing the more practical functionality that will be visible to the customer.

The lack of communication, which resulted in different expectations concerning the user interface, has been noted. We will describe the backlog items in more detail from now on, and also be sure to clarify them with the customer at the Scrum planning meeting. This will better our understanding of what the customer wants, and give us the same expectations for the backlog items as the customer.

Chapter 8

Sprint 2

This chapter describes the work done in sprint 2. Building on the work done in the first sprint, this sprint focused more on adding features and extending the functionality of the application. Instead of only being able to navigate the static 3D scene, the user would now be able to add and manipulate objects dynamically in the scene. The look and feel of the application would be reworked to reflect the added functionality, making it easy and intuitive for the user to perform the available operations.

The addition of a save feature meant incorporating a way to serialise the data structures in the application. The open source Boost libraries were added to the code base, eventually allowing the state of the program to be written to an XML file, which could then be loaded back up at a later time.

The student festival UKA started during this sprint. The team adapted by allowing flexible work hours.

8.1 Sprint goal

The goal of the second sprint is to demonstrate a nice looking GUI and the ability to place, select, move and rotate objects in the 3D environment. On the second sprint meeting it should also be possible to save the project configuration and load this configuration back into the program.

8.2 Sprint Planning

The second sprint planning meeting was held at SINTEF Brattøra on Monday, October 6th. All team members where present, besides Audun, who was on vacation. The product owner, Finn Olav Bjørnson, and his two colleagues Carl Fredrik Sørensen and Gunnar Senneset also participated.

Due to some misunderstandings on the completeness of the GUI after the first sprint, it was decided to add a new item to the backlog; GUI design/prototype to make sure that the finished product will have the look and feel the customer wants. In addition to this item, three other items where picked from the product backlog in Appendix D. New goals for the demo of sprint 2 was also defined.

The new items chosen from the product backlog was:

- Gui design/prototype
- Place objects in 3D space
- Select and move/rotate object in the 3D space
- Save project

GUI design/prototype

The customer had hoped to get a better presentation of the user interface at the first sprint demo. When the user interface presented was too simple, it was decided to add this item to the backlog and prioritise it for sprint 2. The prototype must present how the final user interface will look like with buttons, information about objects and object browser. Having a good user interface prototype would also help the team get a better understanding of how the customer wants the software to behave and what kind of functionality is considered to be the most important.

Place objects in 3D space

Placing objects in the 3D space in an intuitive and easy way is one of the most important features of the software. This will make the user able to quickly visualise and test layouts of objects in the virtual world before anything is done physically.

Select and move/rotate object in 3D space

The user must be able to select objects to be able to manipulate their data, position, orientation and existence. Being able to move objects that have already been placed in the virtual world is also one of the most important features of this software. It allows the user to correct errors made when initially placing objects and generally move things if needed.

Save project

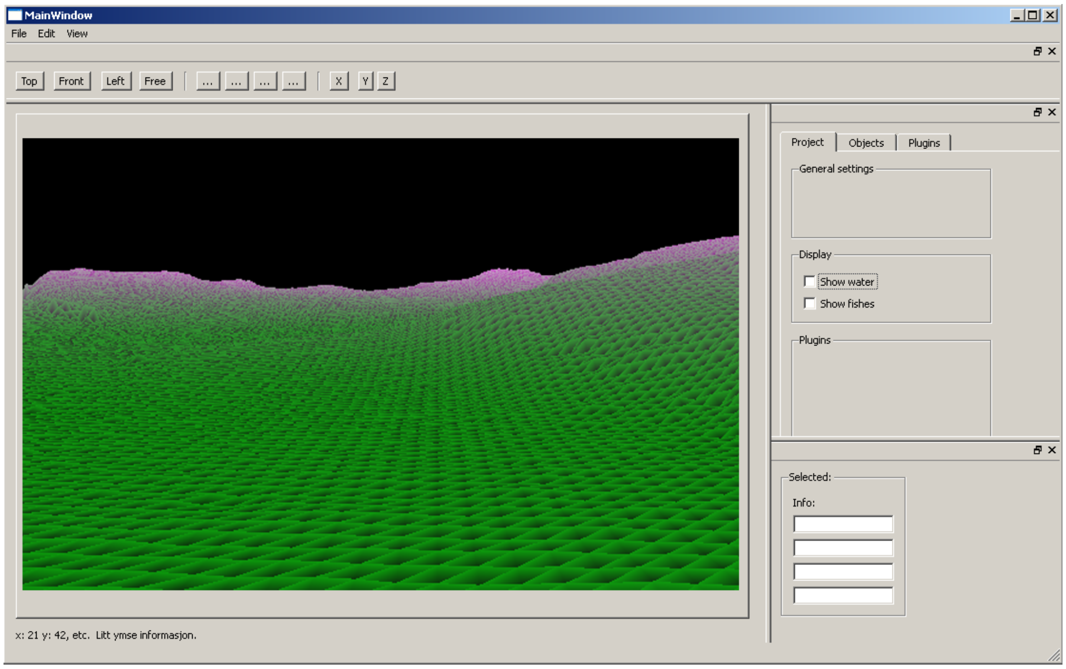
Saving and restoring the state of the map and objects placed allows the user to continue working on a project later, and also share the project with others. The use of XML also opens up for parsing and analysis by other software.

8.3 Actual process

GUI design/prototype

At the customer sprint planning meeting for this sprint, we agreed with the customer to make a GUI-prototype and send it to them for evaluation within the first week of the sprint. We discussed within the group how we should make the overall layout as good as possible considered the already implemented functionality and also considered extensions of functions. We also did research on how the user interface for other commercial application, that one could compare to ours, was designed. Especially other 3D/CAD applications, like Autodesk 3ds Max and SolidWorks were interesting to look at. Afterwards we drew some simple sketches. When we was satisfied with the layout in the skethes, we designed it with the QT Creator, a nice GUI-editor for QT user interfaces. On top of this text you can see one of our early prototypes.

We sent the prototype to the customer, and they were satisfied with the look. So we just started to implement the prototype into the application. Since we used the QT Creator for prototyping, most of the work was already done, and coupling the prototype with the actual program was quite easy. The prototype is shown in Figure 8.3



□ Show water
□ Show fishes

x: 21 y: 42, etc. Litt ymse informasjon.

Figure 8.1: An early prototype of the GUI

Place objects in 3D space

This task was divided into two parts. The first was retrieving the position at which the cursor was pointing and make a dummy object follow it. The second part was making copies of the object at that position if the user clicked the left mouse button. Both were completed without any major problems.

Select and move/rotate object in 3D space

OGRE has native functionality for 3D picking and object transformation. This already existing functionality made this process alot easier than what it could have been. We experienced some problems with making the objects move naturaly in sync with the mouse cursor. After alot of testing and going through the logic we got a natural feeling for moving objects in the end. Rotating objects was done quite easily by using native OGRE functionality.

Save project

Saving a project largely revolved around interfacing the project with the Boost serialisation library which is responsible for saving and restoring the state of the program. Due to the complex nature of the library, a larger part of the sprint than expected was spent getting to know it, which caused problems later on. In addition there were unforeseen technical challenges due to the way our program architecture was structured, which caused additional work. In the end, while basic saving was achieved, restoration of state was not finished in time for the sprint end.

8.4 Software Design

The design was updated to support the new features of this sprint. Mainly support for objects has been added. The kernel now includes a CObjectManager that is the class responsible for storing and managing objects in our world. To load objects in the first place, classes realising CObjectLoader interface will be used. Like map loaders, object loaders also use the chain of responsibility pattern. In the AC3DSView package a CameraManager class was added to move some controller code out of the main program file. CObjectQListWidgetItem and COgreObjectLink are controller classes that bind model items with view items. The class CExampleObjectLoaderPlugin will provide our program with some dummy objects, see Figure 8.3.

8.5 Implementation

Place objects in 3D space

By selecting an object from the list of available objects and clicking a button, a copy of the object follows the cursor when it is inside the OGRE viewport. It is constrained to the horizontal plane, meaning that the user will not be able to place an object anywhere but on the water's surface initially. When the user clicks the left mouse button, an object is placed at the exact location of the copy following the cursor. The copy still follows the cursor, so the user may place objects elsewhere without having to go via the menu every time.

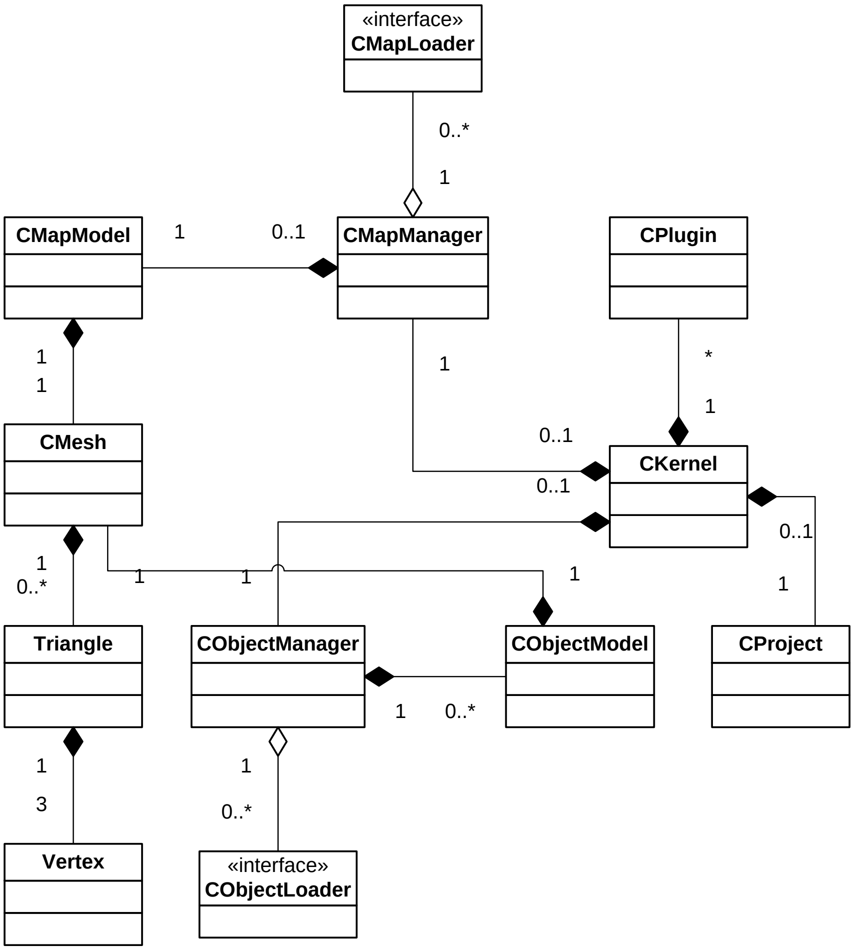


Figure 8.2: Uml class diagram for the core package in sprint 2.

Select and move/rotate object in 3D space

We could use built-in functionality in the 3D engine for this part. Selecting is done by casting a ray (an infinite line) from the camera's position and through the cursor position, and then checking which objects it hits. The first object it hits is highlighted, and if the left mouse button is clicked, also selected.

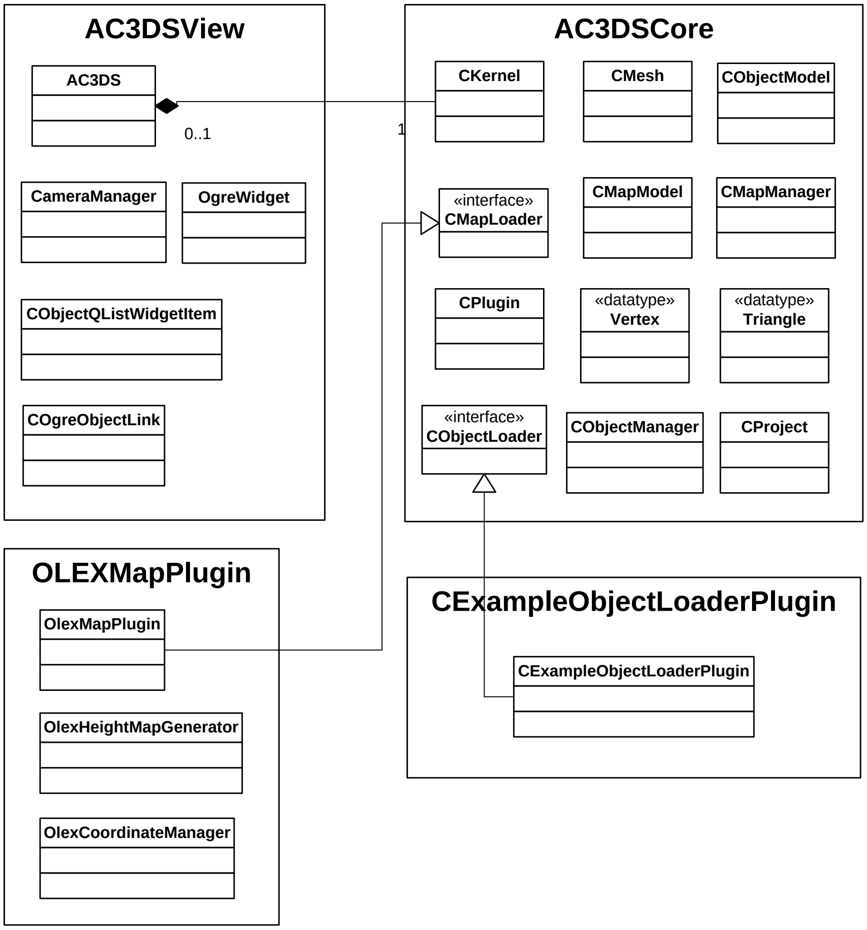


Figure 8.3: Diagram showing class composition and generalisation between packages in sprint 2.

When the move tool is selected, the user may choose one or two of the three available axes to be active. By clicking on the object and keeping the left mouse button pressed, the user can manipulate the selected objects position in either a plane (if two axes are selected) or just along the one selected axis. Only one axis may be selected when using the rotation tool. This defines which axis the object is to be rotated around. The mechanic is the same as with the move tool. By holding down the left mouse button and dragging, the selected object rotates around the selected axis.

Save project

Serialisation with Boost is relatively straightforward, depending on the complexity of your architecture. It works in a strictly hierarchical manner, recursively drilling down from the object you pick for serialisation, and includes any member objects and variables that are defined for serialisation. Which variables should be serialised is specified in the header files for each class and struct. In our case we based the serialisation around CObjectManager which contains references to all available objects and state of objects inserted into the world. The serialised classes were then fed into an XML Archiver, and outputted to disk.

While saving was completed, restoration proved more complicated due to problems initializing objects in OGRE. The architecture at the time was very tightly coupled and did not easily allow for external manipulation of objects.

8.6 Testing

System test

In this sprint the functionality have been strongly extended, and more testing than in the first sprint is needed. The approach of making a system test based on the sprint backlog items worked so well in the first sprint that we have chosen to use this for the rest of the sprints as well. Doing so, we save a lot of time on writing completely new test cases, and still the overall acceptance of the product gets maintained since the items are dependent on each other.

Test ID - Name	T7 – Place object in 3D space
Description	Test if the user can choose a 3D object from a list and place it in the 3D view.
Criteria	• The application shows all available 3D objects in a list • The user can choose one of the objects from the list • The application places the selected object in the 3D window. • Ease of placing the object in a given position in the 3D space.
Estimated hours	4
Actual	4
Result	Passed
Comments	The usability is quite good, you select the preferred object, and drag it around before you choose to place it with clicking the mouse button again.

Test ID - Name	T8 – Select and move/rotate object in 3D space
Description	Test the ability to select a specific 3D object in the application, and modify its placement and orientation.
Criteria	• Select an object by clicking on it • The application shows a bounding box around the selected object • The user can choose different tools for moving or rotating • According to selected tool and axis/axes, the user can change the position or orientation of the object.
Estimated hours	4
Actual	4
Result	Passed
Comments	It is easy to select desirable objects and move them around. It is a bit difficult to rotate it as desired, but not impossible.

Test ID - Name	T13 – Save project
Description	Test if the user can save a project and reload it in the same state as it had before saved.
Criteria	• The user can save a project to a given location on disk • The user can load a project, and every objects and project states gets restored.
Estimated hours	4
Actual	1
Result	Not passed
Comments	It is not possible to save a project at the moment.

8.7 Results

All but one sprint backlog item was completed this sprint. The test manager worked a lot on the user interface and made prototypes which the customer reviewed. The user interface is also connected to the new functionality of the application. Informative buttons and labels let the user choose an object and place it in 3D space while being able to observe its position numerically. Objects that are already placed can be selected and manipulated intuitively using the move and rotate tool.

The sprint backlog item describing the ability to save the state of a project in the application was not finished. A lot of progress was made on the task, but it was not ready in time for the demo meeting at the end of the sprint.

8.8 Evaluation

Several improvements were made on the work process from the first sprint. Learning from the relationship between estimated points and actual work hours, the team now had a better idea of how big a workload it realistically could take on in one sprint. Using this knowledge, we picked backlog items worth just a little more than last time. The decision to take on a bigger workload was done because the team in the first sprint spent extra time getting familiar with the programming environment and the 3D engine.

We were careful to clarify and agree with the customer what the backlog items picked for the sprint meant. This was done to avoid the situation we had at the end of the first sprint, where the customer and the team had a different understanding of what one of the backlog items entailed.

By having one person dedicated to work on the report, we made more progress in that area than we did the previous sprint. In addition to finishing the parts that were missing from the first sprint, most of the parts concerning this sprint was also done. The design section was still lacking. This was not because no designing had been done, but it had yet to be properly written for the report.

The sprint goal was not achieved. The backlog item describing how a user would be able to store the application's state was not completed in time. This item was dependent on the use of an external code library, which was not introduced until this sprint. Getting familiar enough with this library to implement the feature took more time than expected. This resulted in the work on the feature being nearly finished, but not complete or presentable. The customer was satisfied with the other features that were added.

This was also the sprint when the student festival UKA started. We had predicted that this would cause a shift in the daily work hours to accomodate late night activities. However, because the team consists of five individuals with different schedules, we ended up with some days where a few members would come in early to work and some members coming in later. This made it hard to have the regular meetings at the beginning of each day. On the days in question, we ended up having meetings whenever our work hours overlapped, usually some time after noon. Maintaining good communication within the team became harder with the team split like this. We had to rely on instant messaging and e-mail more than before. It did not, however, become a major issue.

Chapter 9

Sprint 3

This chapter describes the work done in sprint 3. Having implemented the basic interactivity regarding placing and moving objects in the 3D world in the last sprint, this sprint would add the ability to give the objects properties, and functionality aiding the user in placing objects on the seabed. The work on saving the project state would also be continued from the last sprint and finished.

9.1 Sprint goal

The goal of the third sprint is to demonstrate the new object property system, being able to place and move objects on the seabed and storing the state of a project made in the application.

9.2 Sprint Planning

The planning meeting for sprint 3 was conducted at SINTEF Brattøra. Carl-Fredrik Sørensen from SINTEF did not attend. During the course of the meeting, two backlog items were added. These were originally contained in other items, but were extracted to simplify the items. A goal for the sprint was also defined.

The items selected from the backlog were:

- Continue working on save project
- Collision detection/place object
- Reference to coordinate system - compass
- Object metadata
- Edit object metadata in menu
- Cost estimation based on object metadata
- User defined metadata

Continue working on save project

Finalise the work started in the previous sprint. The state of the application with all its objects and all their positions and properties can be stored and loaded back up at a later time.

Collision detection/place object

Being able to place objects directly on the seabed without having to first placing them at the surface and then moving them down. Also includes being able to move objects around on the seabed with them automatically sticking to it when moved in slopes.

Reference to coordinate system - compass

To make it easy for the user to know in which direction she is moving an object, or even in which direction she is currently looking, some sort of compass on screen should indicate which direction the different axes represent. This should be in the form of a cross similar to the ones used in major 3D applications already, with a labeled arrow indicating the X, Y and Z axes.

Object metadata

The data structures representing the objects placed in the 3D world should incorporate a list of metadata. This list will take the form of a dictionary, where pairs of properties and their values can be added.

Edit object metadata in menu

The objects having metadata attached to them would be useless if there was no way for the user to edit them. This item includes making user interface elements that display the current metadata contained in objects placed in the 3D world, and a way for the user to edit existing entries in the metadata table.

Cost estimation based on object metadata

By looking at the metadata contained in the objects currently placed in the 3D world, the application should be able to display the total cost of all the objects. This functionality allows the user to estimate the cost of different setups.

User defined metadata

In addition the possibility of editing existing metadata, the user should also be able to add and remove metadata from objects. This item adds the functionality and user interface elements needed to do this.

9.3 Actual process

Continue working on save project

Work continued in this sprint to achieve full saving and loading of program state. A major refactoring was done at the start of the sprint to the view->model interaction, which had been a stumbling block in the previous sprint. The refactoring made it possible to fully implement saving/loading.

9.4 Software Design

In this sprint we changed some of the design to make CModelObject and CModelManager classes observable according to the observer pattern [9] See 9.1 to see how the CObservable class is extended. The CObjectModelObserver is an interface used by objects that will observe the CModel class. A lot of controller code was moved out of the AC3DS class into separate classes, see 9.2 for updated design.

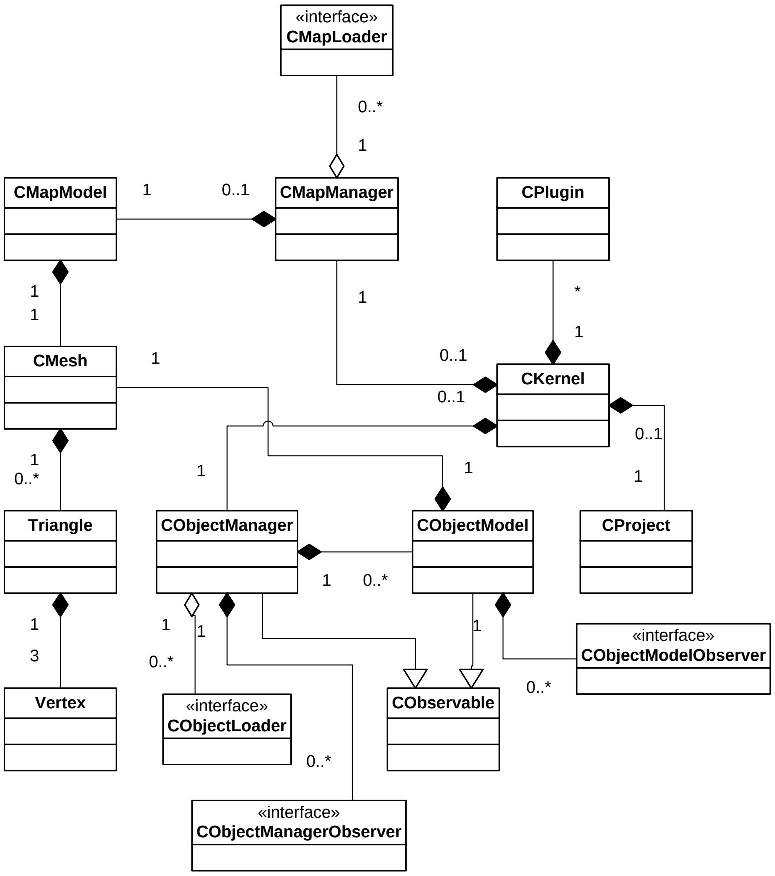


Figure 9.1: Uml class diagram for the core package in sprint 3.

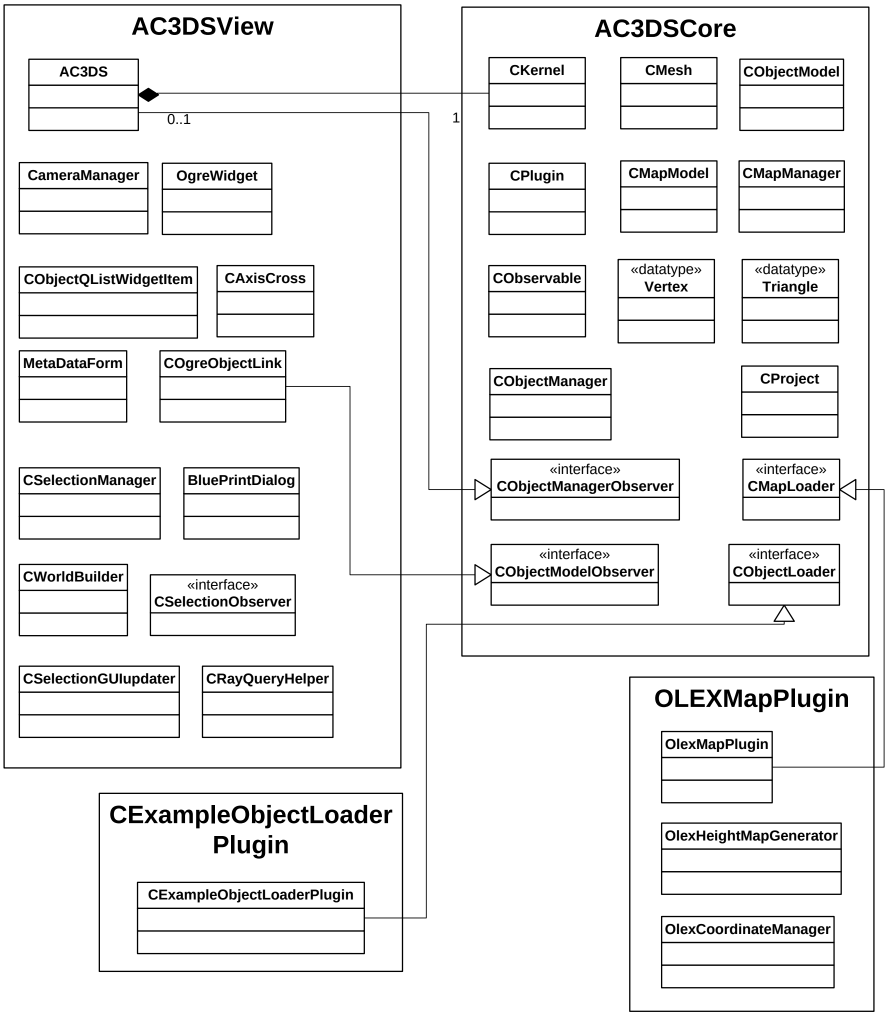


Figure 9.2: Diagram showing class composition and generalisation between packages in sprint 3.

9.5 Implementation

Continue working on save project

Functionality for saving was largely completed in the previous sprint, though additional project information was added which led to some minor changes. While loading data into the model did work, it was not possible to reload the information back into the view. A major refactoring was done on the 3D view to ease the tight coupling. Use of the observer pattern on the model allowed for a much cleaner and modular approach. Since the data was loading correctly into the model, it was at this point a relatively simple task to notify the view, as an observer, of the change, which then loaded the objects into the view.

Snap functionality extended to seabed

In the second sprint the snap functionality that allows objects to be locked at the level of the sea surface while being placed was added. This was to be extended in this sprint to also let objects automatically stick to the seabed as they are placed or moved.

Retrieving the coordinates that correspond to where the cursor is currently pointing on the seabed is not trivial. The 3D engine the application is built on allows for retrieval of the position at which the cursor is pointing on an object's bounding box. The bounding box is an imaginary box that is just big enough to contain all the geometry of an object, for example its height is decided by the object's lowest and highest point. This built-in functionality could not be used in the solution to our problem.

Instead of trying to use the geometry of the object in any way, we settled on a solution that is based on the underlying heightmap from which the map geometry is generated. A ray is cast from the camera's position through the position of the cursor in the viewport and into the scene. Along this ray a binary search is performed to find at which point along it it intersects with the geometry. In short, at points along the ray the difference between a given point's height and the height value obtained from the heightmap is calculated. If the point is higher, the search continues down along the ray. If lower, the search backs up along the ray. Whenever the search changes direction, the distance travelled in the new direction is half of the last distance travelled. This lets the search converge to where the ray is intersecting the geometry. When the difference calculated is below a given threshold or the search has gone through a given number of iterations, it is stopped.

Under normal camera conditions, the search terminates in under 20 iterations. The lookup to find the value from the heightmap is cheap and the computations to find the next point along the ray simple. This makes the solution fast enough for its purpose. There is no noticable slowdown of the application when moving or placing objects on the seabed, even though a new search is performed every time the cursor is moved.

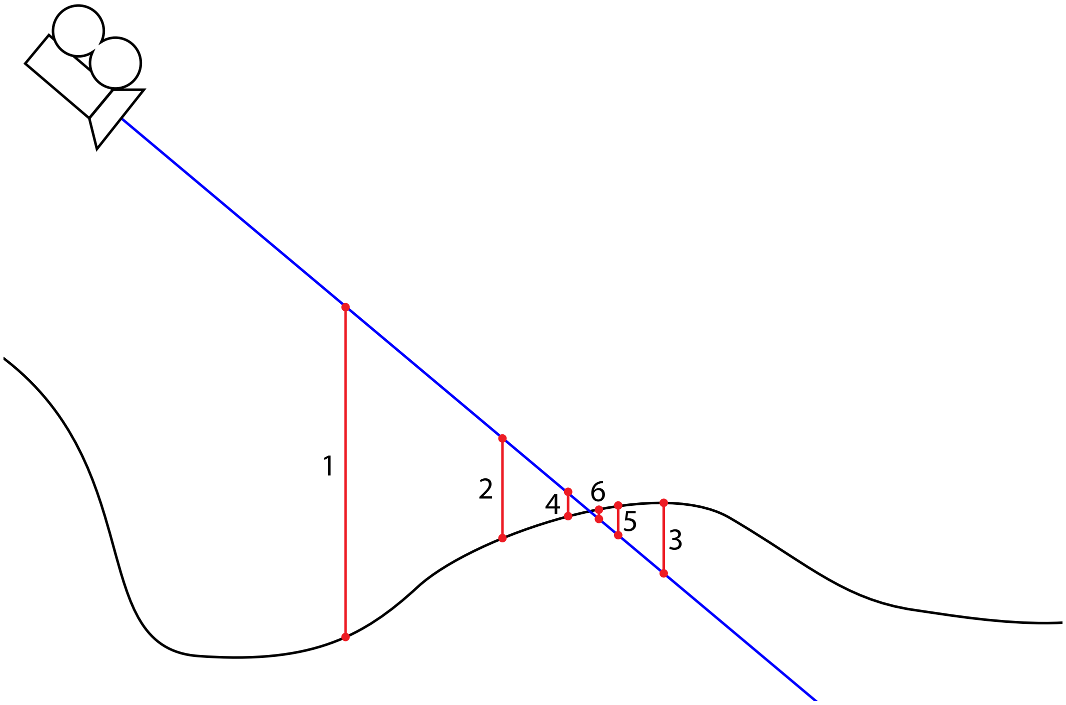


Figure 9.3: The first six iterations of a search to find the intersection where a ray hits the irregular seabed. 1: The first point is above the seabed, so the search continues in the same direction. 2: The second point is also found to be above the seabed. 3: The point is below the seabed, which means the search now backtracks along the ray in its next iteration. 4: The fourth point is above the seabed. 5: The fifth point is below the seabed. 6: The sixth point is also below the seabed, and the search will continue in the same direction in its next iteration.

9.6 Testing

System test

Test ID - Name	T13 – Save project (continued from sprint 2)
Description	Test if the user can save a project and reload it in the same state as it had before saved.
Criteria	• The user can start the program, place objects and load a map. And when he wants he, should be able to select “save project” in the file menu and get up a new dialog box for choosing name and location for the project. • The user can start the program, select “load project” from the file menu and get up a dialog box for choosing which project to load. Then every object and the map in the chosen project should get loaded in with the correct position and other parameters/states they had from when they got saved.
Estimated hours	4
Actual	4
Result	Passed
Comments	The feature function just as wanted. There is also possible to choose “save as..” if you want to save it as another project.

Test ID - Name	T24 – Test collision detection
Description	Test if the collision detection functions as wanted. That is, object that should be placed to the sea bed snaps to it, and object that should be place on the surface snaps to that.
Criteria	• When you choose an object to place, it should automatically follow the preferred surface given the object's metadata.
Estimated hours	4
Actual	4
Result	Passed
Comments	It actually collides only with the surface, and given the coordinate for the collision point at the surface, it gets the depth. This may not be the best solution for placing object, especially when you will place underwater objects, and you don't see the surface.

Test ID - Name	T23 – Test reference to coordinate system
Description	Test if the ability to get an object coordinates in the local coordinate system and also in correct latitudes and longitudes in a global coordinate system.
Criteria	• When you select a placed object, you should be able to see its coordinates in both coordinate systems. • You should be able to place the object numerically changing the coordinates in both coordinate systems.
Estimated hours	4
Actual	4
Result	Passed
Comments	The program shows correct coordinates for a given point on the map if the button for showing latitudes and longitudes is toggled.

Test ID - Name	T9 – Test object metadata
Description	Test the ability to change, add and/or delete object metadata to a specific object, or an object's blueprint.
Criteria	• For a selected object, you should change values for already added properties. • You shall be able to add new properties and values for them. • You should select a blueprint - the main object the instance have been created from - and choose to override the local metadata for the instances created from that blueprint. Like synchronization of every instance.
Estimated hours	4
Actual	4
Result	Passed
Comments	It works OK, but is very primitive. There are three predefined metadata properties for each object - material,snap and cost. • **Material:** A string saying what kind of material it is made of, ex. steel, plastic, etc.. • **Cost:** An integer representing the cost of the one object instance. • **Snap:** Specifies if the object is supposed to snap to the bottom or the surface when placed. You can also add new properties, but I could only add one. I can delete all of the properties, but only create one new also then. This is not the preferred functionality.

Test ID - Name	T21 – Cost estimation based on metadata
Description	One should be able to make cost estimations based on different metadata form objects in the project.
Criteria	
Estimated hours	4
Actual	2
Result	Passed
Comments	You get the overall cost of the project based on the cost and amount of object in the project.

9.7 Results

All the backlog items selected were completed this sprint. The one concerning saving the state of the project was continued from the last sprint and finished. In addition, the user is now able easily place and move objects along the seabed because of the automatic snap functionality. This means that objects defined to belong on the seabed will have their depth value updated to reflect the depth at which the seabed is at that particular object's location.

The other major feature that was added this sprint was the ability for objects to store metadata. This can be cost, material or whatever else the user sees fit. One example of how this new metadata is used is deciding what an object should snap to when being placed or moved. By setting a certain property in an object's metadata, the user can change this from being for example the water's surface, which is default, to being the seabed. The object will update and stick to the seabed the next time it is being moved. Another use for the metadata is the simple cost estimation, which sums up the cost of all the objects currently placed in the world. This cost can be manipulated as any other metadata property.

9.8 Evaluation

The backlog items picked for this sprint summed up to a little less than those of the last sprint. We had not been able to complete all the work, and decided the lighter workload and the work still remaining from the last sprint would be enough. Making this adjustment was a factor that made us achieve our goal for this sprint.

The still ongoing UKA festival had the same impact on the first half of this sprint as it had last sprint. Despite the odd work hours, the team worked efficiently and was on schedule the first half was over. The team had become increasingly comfortable with the programming environment and the system design over the last few weeks. In addition, no new code libraries were introduced, so we were working with code we had written ourselves. This made the implementation process more predictable.

All the backlog items, including the unfinished one from last sprint, were completed and the goal was achieved. While the last sprint had added some interactivity to the application, this sprint added the ability for the user to add their own properties to objects. Being able to save the state of the application is also an important feature that was completed. The customer was very satisfied with the demonstration, and particularly impressed with the way an object could be moved around while sticking to the seabed.

The work on the report was less than satisfactory. The chapter concerning the second sprint was nearly completed, but not a lot of work was done on the one for the third sprint beyond the planning and design sections. The group had this in mind when moving on to the next sprint, as even more work would have to be done in the report.

Chapter 10

Sprint 4

This chapter describes the work done in sprint 4. The main tasks done in this sprint would be the functionality to connect objects in the 3D world, allowing the user to define a colour scheme to assign to the map geometry and letting the user import or make objects to use in the project.

The group wanted to reduce the amount of implementation work for this last sprint, to be able to work on the report and final presentation. The customer was aware of this, but had several requests regarding functionality. We settled on picking out four heavy items from the product backlog. The customer understood that the group might not accomplish all tasks, but wanted all four tasks to be selected for the sprint to ensure that implementation of the most important tasks was done if there were time left.

The sprint would end in a demo meeting where the customer invited several people from SINTEF that might be interested in the application. This meeting would mark the end of the programming part of the project.

10.1 Sprint goal

The goal of the fourth sprint is to demonstrate the ability to connect objects, generate realistic anchor lines, change the colour and colour scheme of the map geometry and import user made objects from external files.

10.2 Sprint Planning

The planning meeting for the fourth sprint was held October 30th at SINTEF Brattøra. Everyone but Samson Valland attended the meeting. The team discussed the priority of the remaining backlog items with the customer. They were items that initially had been given low priority, because the core functionality had been implemented first with little regard to personal opinions. This time the discussion was more about what each of the customer's representatives personally thought would be most useful in a prototype.

The items selected from the backlog were:

- Import objects (3D models) made by the user
- Connect different objects
- Basic physics applied to the anchor lines
- Map colour plugin

Import objects (3D models) made by the user

Expanding the library of objects available to the user in the application is important. This allows the user to use objects that did not exist at the time when the application was made, and removes the responsibility of foresight from the developers.

Connect different objects

Being able to connect objects is important to have the 3D world model the real world. This means connecting objects into logical groups, so that moving the group makes all the objects within it follow. The other type of connection models the fact that objects on the surface are connected to anchors, positioned on the seabed or elsewhere, by anchor lines. When such a connection is set up, the anchor line should update its shape if either of the objects it is connected to is moved.

Basic physics applied to anchor lines

In the real world, all of the objects placed on the surface or below are affected by forces such as wind, current and gravity. An anchor line's length must take these forces into consideration. Its length affects its cost, so applying a physical model to the anchor lines in the 3D world is important to get good cost estimates.

Map colour plugin

Being able to colour the map geometry allows the user to present the shape of the terrain in both a functional and aesthetic manner. A functional example is to set colour interval which colours the geometry according to which height it is at. The length of the interval determines which depth interval one colour should represent. For presentation purposes, the user may want to use realistic or bright colours to emphasise the contours in the terrain.

10.3 Actual Process

Import objects (3D models) made by the user

Importing a 3D object meant reading a file describing a mesh at runtime. The 3D engine supports reading files in its own file format, but this is not a format is special to the engine. To be able to use 3D models made in other applications, a converting process had to be integrated to convert the mesh files into the file format which the engine could read. We had, together with the customer, decided to focus on the 3ds file format, which is a file format produced by several major 3D applications, most notably 3ds Max. Code snippets to perform this conversion were freely available through the engine's website, so most of the work went into understanding and integrating this existing code into the project. No real issues were encountered, and this feature was completed within the first week of the sprint as planned.

Connect different objects

The process started with a brain storming session, we had alot of ideas on how to realise this requirement. It was quite a big task to create support for connected objects, several substasks existed within this larger task. Creating hierarchy support for objects had to be done first, then connection point objects could be implemented. At last the object simulating a chain could be created. Connection points and chain objects where done via polymorphy and that created some troubles because we had not created support for this from the start. After some work to support polymorphy in our system everything worked well.

Basic physics applied to anchor lines

We had not received any information as to which physical model the anchor lines would follow, so a day was spent researching that. No simple solution was available, and to avoid spending more time than estimated but still have a nice visual result, the team decided to go with a model that would not model the deforming of the anchor lines realistically. Instead, a simple setup using bezier curves was implemented. The result is a deformation that looks OK for visualising purposes, but is not accurate enough to base cost estimation off of. This backlog item was implemented during the first week of the sprint.

Map colour plugin

Implementing the manager whose task it is to generate the colour scale and act as a container for it was done in a couple of hours. The challenge was integrating it with the engine and Qt to allow a user to visually pick colours and have them applied to the terrain geometry. We had to alter the way in which we build the geometry to able to apply the colours at runtime, as discussed in subsection **??**. Building the user interface elements needed to pick colours from a palette required a bit of research. Qt had a built-in colour picking dialog which we could use for this purpose. This backlog item was implemented during the first week of the sprint.

Software Design

Two new packages was added in this sprint, the SnapPlugin and the MeshLoaderPlugin. The SnapPlugin is a plugin that will observe objects and always enforce their snap property, therefor this plugin extends the CObjectModelObserver. See Figure 10.1 for updated package overview. CMapColorManager and MapColorWidget have been added to the AC3DSView to implement the map colour support. CObjectModelConnector and CConnectionPoint has been added to the core, these classes extends the CObjectModel to provide new functionality required by the new object types in this sprint. See fig 10.2 for an updated view of the core.

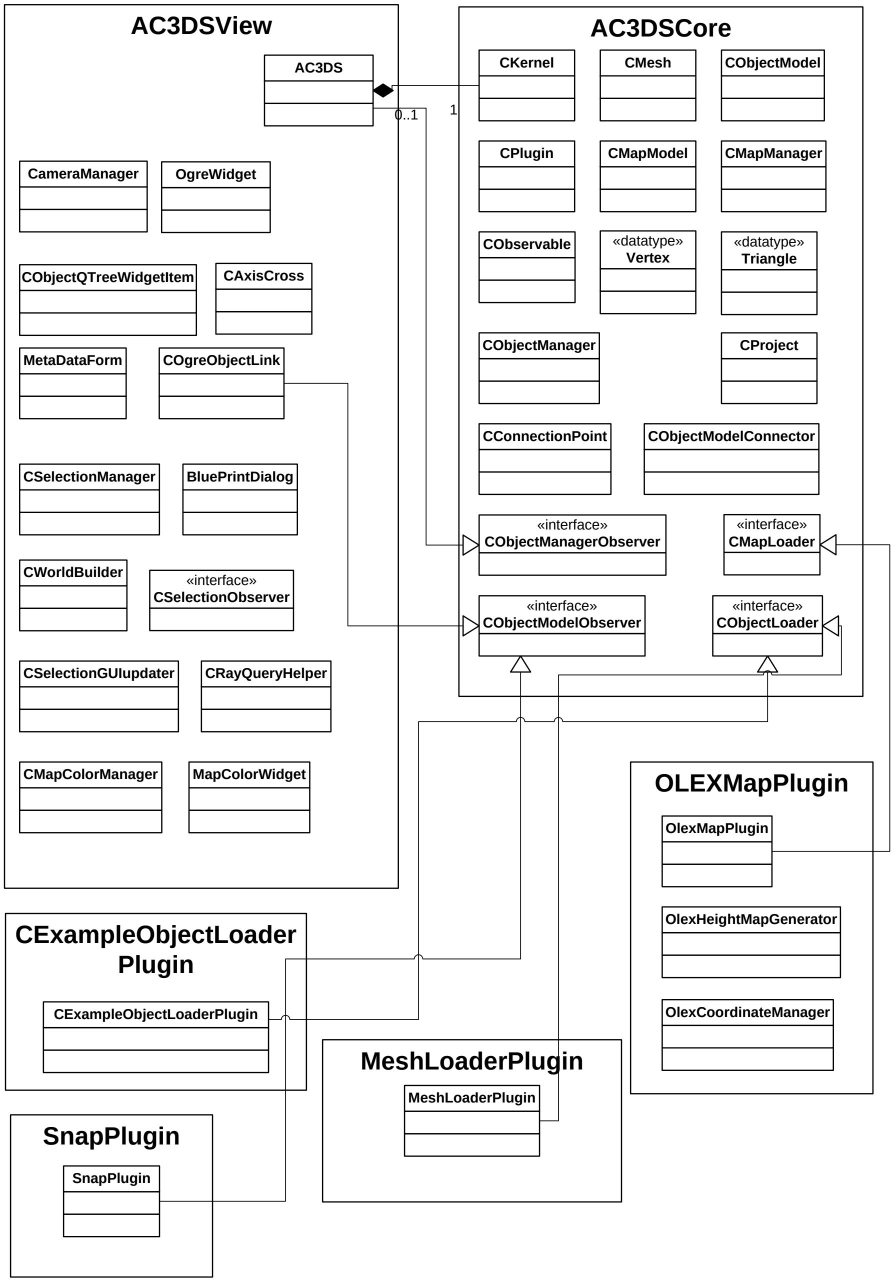


Figure 10.1: Diagram showing class composition and generalisation between packages in sprint 4.

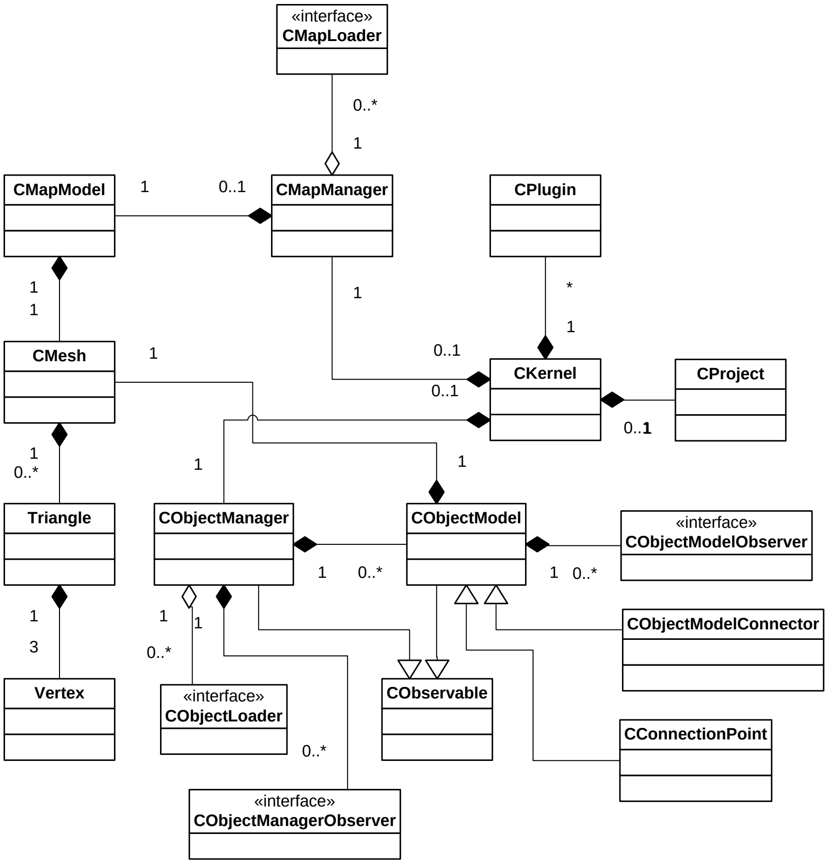


Figure 10.2: Uml class diagram for the core package in sprint 4.

10.4 Implementation

Connect different objects

Support for connecting objects was done via three sub features. First objects would have to support a form for hierarchy to allow connection points be connected to them. This was done via adding a children list to all objects. Objects could now be creating in a hierarchy, by having an object selected when creating a new object the already selected object becomes the parent of the newly created one. The view would also have to be upgraded to support viewing this hierarchy as previously it had only shown a flat structure.

The next features to implement were connection points and the connectors that connect them. This was done via subclassing the CObjectModel extending it with the new required functionality. Another subclass of CObjectModel called CObjectModelConnector was also created, objects that will connect to a connection point will be of this type. It has the special functionality that it can be connected to two connection points. These two new types of objects extends their baseclass via Polymorphy, this caused some trouble in our code since we were using the copy constructor in creation objects, and the copy constructor cannot be virtual. The CObjectModel therefor had to define a virtual clone function to allow subclasses to be created correctly. For the connector type the 3D view object placement would have to be customised since it is not a regular object. Instead of placing a copy like regular objects, the connector will be placed between two connection points once they have been clicked.

Basic physics applied to anchor lines

Implementing the bezier curves that would imitate the deformed anchor lines, and making them look good, was done in a couple of hours. The 3D engine had built-in classes to facilitate the creation and visualisation of them. This backlog item was completed during the first week of the sprint.

Map colour plugin

Up until this point the colour of the terrain geometry had been set to a green gradient that made the higher areas of the map light green and the lower areas a darker green. These colours were set as the mesh geometry was assembled. After making all the triangles that the terrain was made up of, the geometry was made static to optimise performance. This meant that changing the geometry itself or its colour scheme was not possible, as the underlying structure of vertices, triangles and colours had been "sealed" to gain performance.

To gain access to the colour setting of the terrain mesh, the way in which it was built had to be changed. After testing that the difference was insignificant, we made the decision to skip the optimisation of the geometry. This allowed us to easily access and update the colour settings of it. Making the plugin that generates the colour scale for the geometry was simple. A manager object was introduced. Its tasks were to build a scale from a given pair of colours, and to return the colour value for any depth. The scale could either be a gradient that provided a smooth transition from the first colour to the second colour, or a scale where the change in colour was at larger intervals. Generating the scale from the two input colours was a case of implementing linear interpolation between two points.

10.5 Testing

System test

Test ID - Name	T4 – Import objects from SolidWorks via menu
Description	Test the ability to import 3D-objects from SolidWorks.
Criteria	• The user can select “import object” from the file menu, get up a file dialog and select which file(of the correct format) to import. • The object gets loaded into the “object blueprints”-list among the other objects in the project. • The imported has correct scale according to the project’s scale and the object’s own specified size.
Estimated hours	4
Actual	4
Result	Not passed
Comments	It is impossible to import SolidWorks-objects directly, but it is possible to get these kind of objects in if you convert them to proper 3D-meshes first. This is achieved with a simple application made for this task.

Test ID - Name	T10 – Connect different objects
Description	Test the ability to join different objects.
Criteria	• The user can select an object and attach connection points to it. The connection points follows the selected object's position when it gets moved later on. • The user can connection lines(anchor lines e.g.) between different connection points regardless of their parent object or if they have any at all. • The user can also create children object that is any kind of object, not just connection points.
Estimated hours	4
Actual	4
Result	Passed
Comments	It works as wanted

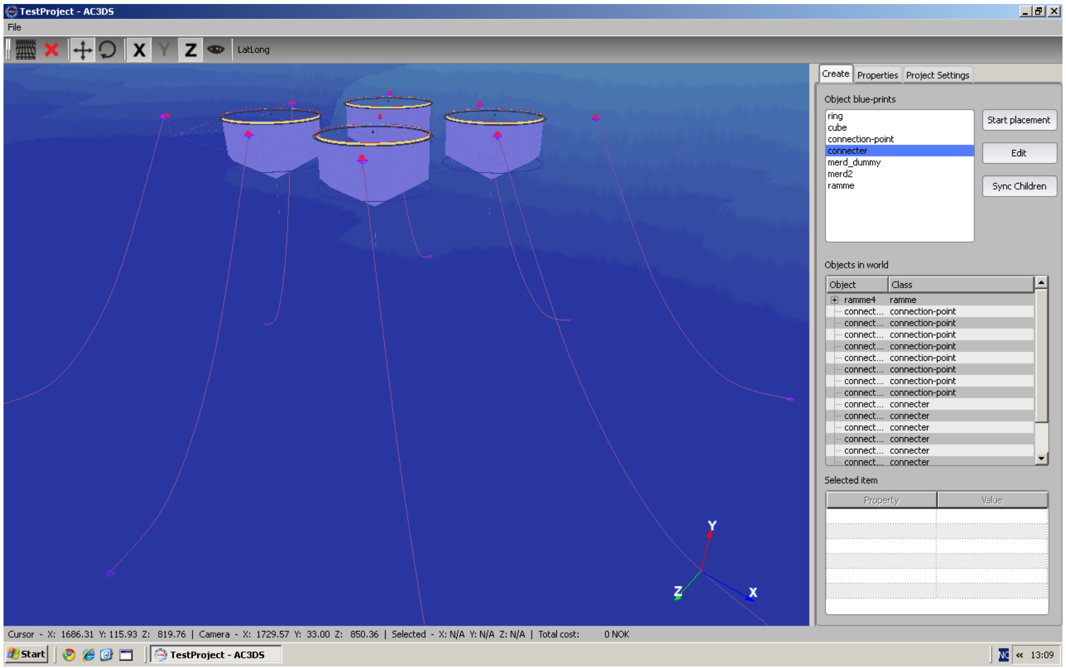
Test ID - Name	T20 – Basic physics applied to anchor lines
Description	Anchor lines between different objects should bend like real-world objects
Criteria	There is no important criteria here, it should just look nice/real.
Estimated hours	2
Actual	2
Result	Passed
Comments	

Test ID - Name	T6 – Map color plugin
Description	One should be able to change the coloring of the map, set different colors for specific depths and gradients on certain intervals.
Criteria	• Via the palette for map properties, there is an interface to set and change different map color settings. • The map coloring updates when you have chosen your preferred settings
Estimated hours	2
Actual	2
Result	Passed
Comments	

10.6 Results

All the selected backlog items for this sprint were completed. The user can now create objects from files describing a 3D mesh, connect objects with anchorlines that deform according to a very basic model of gravity and change the colour scheme of the terrain geometry.

The feedback from the customer after the sprint demo was positive. They were impressed with the way connecting objects worked, and how it was possible to import 3D models and use them in the application. The extra people that had been invited were also content with what they saw, coming from a different background than the three who had followed our progress throughout the project. The demo was used as a source for further discussions. Possible extensions to allow realtime streaming of data and simulation of the environment were highlighted as important possibilities that this software could facilitate. Within this rapidly evolving industry, this kind of software is something they think will be able to give actors in the market an advantage. Simulating the environment allows owners to quickly evaluate locations and setups for their fish farms, thus allowing them to optimise to increase the production



Object	Class
ramme4	ramme
connect...	connection-point
connect...	connection-point
connect...	connection-point
connect...	connection-point
connect...	connection-point
connect...	connection-point
connect...	connection-point
connect...	connection-point
connect...	connecter
connect...	connecter
connect...	connecter
connect...	connecter
connect...	connecter
connect...	connecter

Property	Value

Cursor - X: 1686.31 Y: 115.93 Z: 819.76 | Camera - X: 1729.57 Y: 33.00 Z: 850.36 | Selected - X: N/A Y: N/A Z: N/A | Total cost: 0 NOK

Figure 10.3: Screenshot from the final version

efficiency.

10.7 Evaluation

The workload this sprint was estimated to be a little less than that of the previous sprint. This was to allow the team spend more time on the report, and do the finishing touches on the product before the last sprint demo. The sprint goal was reached, and the demo meeting a success.

Initially, the plan had been to get most of the implementation work done in the first week of the sprint so that the second week could be spent working on testing and the report. Our technical co-manager was, however, ill that first week. As the task he was supposed to finish was well underway and well known only to him, the rest of the team decided against trying to complete that part of the implementation without him. When all the other implementation work was complete, they began working on testing the application and completing the report. This put a lot of pressure on the technical co-manager when he returned, as he didn't have the buffer that the second week was supposed to provide. The technical manager, being free of any major implementation work, assisted the technical co-manager.

The work effort on the report was divided between the parts remaining in the previous and current sprint chapters, and also the software architecture chapter. We were able to spend less time and write more coherently by focusing only on the report for the last half of the sprint.

Chapter 11

Evaluation and Conclusion

This chapter contains the evaluation of the whole project. All relevant aspects will be discussed, including the internal process, our results, the finished product,suggestions on future work and improvements, the customer, supervisors and the course as a whole.

The first section will deal with the task given in the project. We will share our thoughts of the initial assignment and comment on the specification as the project evolved.

Secondly, we will give our feedback on the customer, discuss the way we communicated with them and give a resume of all the resources the customer provided us with.

The whole process will then be evaluated in the third section. The team dynamics will be discussed and some conflicts evaluated. The choice of Scrum as a project framework and relevant technology used for project management will be evaluated before we summarise the positive and negative sides of the whole process. At the end we will state what we have learned from the whole project.

In the fourth section, we give an overview of the results from the project. Both the product and the report will be evaluated.

The fifth and sixth section will be dedicated to the evaluation of the supervisors and the course. The supervisor resources, quality and communications will be discussed. Regarding the course, we want to evaluate the seminars, the compendium and the course as a whole together with suggestions on improvements for later.

11.1 Project Task

Our group was, in our opinion, given one of the more complex assignments out of all the ones presented in the Customer Driven Project 2009. The original project description can be found in appendix A. When first presented with the task, it was obvious that the team members lacked experience in 3D programming. The task seemed like it was comprehensive and would require a lot of effort.

Shortly after being assigned to the task, the group had a short meeting with the product owner and our supervisors. The group perceived the original project description as quite vague, and the situation had not changed after the first meeting. Later the same week, we was invited to SINTEF's facilities at Brattøra in Trondheim. There we met the product owner again together with two colleagues. Before the meeting, the group was certain that we had to clarify the task and the customer's wishes to be able to make something useful. Based on this meeting, a requirement specification was created and a clearer picture of the final product was formed.

As the team chose to use an agile development framework, the requirements specification (the backlog) was updated several times during the project. New items were added and other items removed. Using Scrum was essential for getting the final product as close to the customer needs as possible, and resulted in a highly functional product regardless of the vague project description given in the beginning of the project.

11.2 Customer Relation

Our group had the pleasure of having SINTEF Fisheries and Aquaculture as customer. They have great experience with using students for projects like this and was very helpful in many ways.

Despite the fact that SINTEF is a large organisation, we got to know our customer as flexible and amenable and they had great understanding of our concerns regarding workload and report writing. During meetings, the communication with the customer was informal and at the same time effective. We managed to keep customer meetings short, while stille being able to discuss all the current matters thoroughly.

Having defined some quality attributes for communications with the customer in the beginning of the project turned out to work well, and we had no problems getting the information we needed in time. All quality attributes are stated in section 6.1.

Getting access to customer resources was crucial in this project, as the software we implemented was supposed to use specific map data and 3D objects. The customer provided us with this information in good time together with blueprints and illustrations on how fish farms are constructed and fastened to the seabed.

If we had more time on the project, we would definitely ask for some end users of the software to make better user testing of the user interface and functionality. That being mentioned, we would hardly have time for doing so during this short project, and the group decided not to prioritise user testing.

Among the things we requested, but did not receive from the customer, was map data from other locations, to perform better testing of the software's ability to read the OLEX map data.

11.3 Development Process

11.3.1 Team

Students are not allowed to choose their own groups in the Customer Driven Project. This means that you have to work with the students you are put together with. Our group ended up having five members and none of us knew any other team member from before. This turned out to work well for us and we quickly organised the group and started working together as a team.

Our group also turned out to be one of the few without international students present in the group. Even though international students would not be a big obstacle for team dynamics, we had the advantage of communicating internally in Norwegian. Being able to communicate in our native language lead to an informal and good tone in the group, where jokes could be told and quick comments were made without misunderstandings.

On the other hand, as we all were unknown to each other, it soon came clear that we had different working norms and goals. Some of the team members preferred working in the evenings and nights, while others wanted to meet in the mornings and work at daytime. The group decided early to meet every morning at 9am to perform a daily Scrum meeting. Getting all team members to show up in time turned out to be a challenge, even though we decided upon a "punishment system", described in section 3.4. Despite many attempts to get all the team members to show up in time, we landed on a solution where all members were given tasks to do, but could choose where and when to execute them. We still had mandatory meetings, but at a different time.

Looking back at the whole development process, we are very satisfied with the good team spirit that we built throughout the project. The only thing worth analysing is the problems we had with team members showing up in time. Some more measures to prevent this happening could have been implemented in the beginning of the project. For example, we could have set the daily scrum meeting to noon each day instead of 9am.

11.3.2 Team Conflicts

The team had the privilege of not running into any major conflicts during the project. We managed to keep a democratic team where all members were allowed to give their opinion and all opinions were taken into consideration. None of the students in the team are experts in C++ programming or 3D software, which meant that no one was confident in having the only correct solution to a problem.

It is also worth mentioning that in many of the most complex discussions on finding a solution to a problem, we had to collaborate to find one good solution, because none of us knew how to solve the problem. Hence, many of our discussions have been creative and positive.

Early in the project, we gave each team member a certain role, and development tasks were given to different team members. The team member responsible for a task, was often the one to decide on the solution to use for their task.

11.3.3 Experiences with Scrum

Based on preliminary studies, personal preferences and customer wishes, the team settled on using Scrum early in the project. All team members, and especially the Scrum Master, spent time getting to know the Scrum process by reading books and web pages on the subject. In short, Scrum is an agile process framework especially used on software development [33]. Projects start off with a planning and pre-study phase and continue into a series of development iterations called sprints. After the final sprint, it is convenient to have a short evaluation phase to discuss the whole process and make notes of experiences worth bringing on to new Scrum projects. A more detailed description can be found in subsecion 2.1.2.

Initially, the team settled on using two and a half weeks on planning and preliminary studies, then go through three sprints of 3+3+2 weeks and end the whole project using two weeks on evaluation, conclusion and preparation for the final presentation. We soon realised that the time scheduled for planning and preliminary studies was too short. Our supervisors confirmed this on the first supervisor meeting. They also pointed out that our sprints should be of the same length, so that it would be easier for us to compare the workload from sprint to sprint. Four sprints would also give us an extra meeting with the customer making it possible to adjust the requirements even more. In addition, making the sprints shorter gave the team more deadlines ensuring the progress of the project was going on schedule.

Each sprint is supposed to start off with a sprint planning meeting and end in a sprint demo meeting. To save time for both the group and the customer, we chose to combine these two meetings into one. Every second week, we had a sprint demo meeting with the customer and continued directly with the sprint planning meeting. This worked out great, both for the group and the customer.

In hindsight of the project, we are confident that using Scrum was the right choice for our group. Having meetings with the customer every second week gave us great feedback on the work, and unique opportunities to change the product backlog in order to adapt to new demands. Already by the first sprint demo meeting we made major changes to the product backlog and the further development to ensure the customer was getting what they wanted.

Having a daily scrum meeting also proved to be beneficial to keep track of the progress of the project. On the scrum meeting each team member related to a task they where going to perform that day, and the next scrum meeting would confirm if the task was done or not. The daily scrum meeting is also good for the team because all team members have to share their work and experiences, leaving no one behind.

The social aspect is also important to mention. When using scrum we focused on sitting together as a team when working. This also contributed to the good team dynamics we achieved.

11.3.4 Documentation of Project Progress

In the beginning of the project, each group was given access to a server at IDI. All code and written documents were committed to this server using subversion. By using subversion we ensured that all team members had access to all files at all times. On the same server, we had access to a project management tool named Trac. Trac was connected to the subversion repository, and logged all committed files and changes. In addition, we kept track of all sprint backlog items in the system. Each phase of the project was divided into what the Trac system calls Milestones. All team members had access to the system and could log their progress on tasks as well as create new tickets for bugs and enhancements for our software. From the Roadmap view we had full control over the progress of each Milestone. The Trac system is described in section 6.3.

Having a system like this to track the progress of our work turned out to be very valuable. We experienced that the system only works if all team members use it diligently; otherwise the system is not up to date and team members can end up working on the same task.

Each week the group produced a weekly status report before the meeting with supervisors. The weekly status report contained the tasks planned and accomplished last week, as well as what was planned to be done the upcoming week. It also contained our time registration sheet and current sprint backlog with a burndown chart. We found the burndown chart to be a nice tool to track the progress of our work, and wished for this feature to be implemented in the Trac system. Our burndown chart was made in Microsoft Excel. Using Excel for such a task works fine, but one drawback is the missing ability for all team members to access the file at one time.

Working hours was registered in an online Google Spreadsheet. The big advantage of using this tool was that all team members could access and edit the document at the same time. Keeping track of the actual time used for each task turned out to be a challenge. To get the time as accurate as possible, all team members had their own spreadsheet which they updated as often as possible. We would also like to see better time tracking possibilities in systems like the Trac.

Overall, we think we succeeded in documenting our project progress in a good way. As the project went on, we got better and better routines for project progress documentation and in the end we were able to construct the weekly status report and have full overview of the project in very little time.

11.4 Results

The finished product delivered to the customer exceeds all expectations we had before we started the implementation, considering the little knowledge we had about C++ and 3D programming. We achieved this goal by systematically working on each task in the backlog and as the project went on, less time had to be used on research and learning and more time could be used on implementation.

If we were to start over again with the implementation, we would focus more on the software architecture and the documentation of the code. Even though we have tried our best to fulfil these goals, it is obvious that further development on the software by other parts can be a challenge and requires a lot of research on the existing code.

Advanced 3D tools for deciding on what equipment to use for fish production and where to place them are, according to our customer, definitely going to be used in the future. Our software is a "first step" in the work on showing how this can be done. Our application only generates a realistic 3D environment with possibilities to place aquacultural equipment and then make simple cost estimations on the models. In the future, we will se applications like this being used whenever someone is going to construct a fish farm or reorganise their existing one. Getting full feedback on oxygen level in the water, water currents, temperatures and so on are important for optimising such a facility.

To continue building on our application to make such advanced software is, in our opinion, not the best solution. Our software mainly showcases the possibilities in the technology, and is a tool for SINTEF Fisheries and Aquaculture to use for demonstration purposes.

The other big task in this project was the report itself. One person in the group was dedicated to work on the report the whole project. In addition, the other team members have written parts naturally for their role. We realise that more time should have been dedicated to the report from the beginning. A slow start on the report resulted in all team members having to make a big effort at the end of the project. Looking at it now, we think the report is good and well structured. It contains all our planning, preliminary studies, results and experiences from the project.

Proofreading and structuring the text good are what we think are the weakest points of the report. Unfortunately, we did not have more time to spend on this. We would also have prefered to have more time to analyse the whole project, define the positive parts and look into what we can do better in the future. That being said, we have learned a lot about project management and how the Scrum framework works. We think the good software we have produced gives reason to believe that the process itself went well, and we hope we have succeeded in making it clear in the report also.

11.5 Supervisors

The group was provided with two supervisors during the project. We had meetings every week where we evaluated the report and the weekly status report. Producing agendas, meeting minutes, weekly status reports and other documentation for the meetings took a lot of time. It took the group a surprisingly long time to find a way of making the document creation efficient. It was first during our last sprint we were able to produce all documents in under two person hours. The reason for this was constant improvements to the templates we used.

Providing the supervisors with all this documentation helped them understand how we were doing with the project. Having the documents available at all times also helped the group during the project. All reports, meeting minutes and agendas were stored in our repository so all team members had access to the latest versions.

Feedback from the supervisors has been important during the whole project. Good directions during the preliminary studies and on choices in the Scrum process was crucial for our project to succeed. The amount of feedback the last few weeks before delivery was very good, and we experienced the supervisors to always help us with our questions. During the last week we were given feedback from the supervisors several times. This resulted in improvements throughout the report, concerning everything from spelling to document structure.

The communication with the supervisors has mainly been on the weekly status meetings and through e-mails when simple questions needed answers. This worked out fine for us. We did not use the supervisors as much as we may could, but we got nice feedback on each meeting. In addition, our main supervisor had several extra meetings with representatives from the group the last weeks before delivery to go through the report.

We experienced the supervisors to have good knowledge in report writing and in the Scrum framework. The only minor obstacle was the differences in native language. We had no problems communicating in English, but the communication on the meetings might have been even better if we all were more proficient in English or spoke the same native language.

11.6 Customer Driven Project Course

The customer driven project is well known by all students on the computer science course at NTNU and all team members had high expectations from the course. The overall impression is that it is very useful and gives the students a way to get to know the working life. This way of working differs from most other courses at the university. It is hard to pinpoint exactly what you learn from a course like this, but it definitely gives the students more confidence on their upcoming careers as employees.

11.6.1 Assignments

Our impression is that the assignments of the course varied a lot. Some of the assignments were small web projects where the assigned group were able to make a product in short time and then have more time focusing on the process and the report. Other groups were assigned more complex tasks which required a lot of implementation time to even have something to show.

It is obvious that the complexity of tasks will vary from group to group and year to year. That is something the groups have to handle in the best possible way. We believe that it is important to emphasise this when it comes to the evaluation of the report and the whole project.

11.6.2 Workload

Because of the variation on complexity of tasks, the workload on groups will vary. We believe we had one of the more complex tasks in this years customer driven project, and it shows on our timesheet. Our group has used around 1700 person hours on the project. The expected workload for this course is 1440 hours, so we exceeded this by approximately 250. If we were to shrink the project to 1440 person hours, it would have affected our final product significantly in a negative way.

This course is, on paper, only half of a semester on computer science. We feel confident in saying that having two courses like this in one semester would be near impossible. The workload is tremendous in comparison to two randomly chosen subjects. In addition, we feel that having to think about other courses as we worked on the customer driven project affected the work significantly. If the course was the only thing the students had to focus on for one semester, even though it would require twice the workload, could really lead to a more realistic approach to the work life. In addition, agile development methods such as Scrum work best if you are working full time on a project with possibilities to meet every day and have the daily scrum meetings.

11.6.3 Seminars

In the following list, we will give a short evaluation of each of the seminars in the course.

Mini seminar in group dynamics: We felt that a lot of the things said during the two sessions in group dynamics were trivial. The focus was too general. Instead of talking about group dynamics in general, we would have liked to see a seminar focused more on the situation we were in. At the end of the second group dynamics seminar, Steinar Gynnild (the lecturer) touched this.

Project Management: Hearing how Bearingpont, as a major player in the consultancy market, managed their projects was very interesting. Maybe the things we heard about were hard to transfer to our concrete project, but it definitely gave us an impression on how project management works in “the real world”.

Scrum, agile development method: None of the team members attended this seminar.

IT-architecture: According to the topic of this seminar, this should be very relevant for our software which required a good software architecture. Disappointingly, the seminar was more about the history of computer than IT-architecture.

Seminar in technical writing: This seminar should in our opinion consist of concrete information on how to structure and write a technical report. Instead we found the seminar to contain only small concrete tips like where to place page numbers and how the headings should look like. We would like to see some more about what parts of a report is important and where to put different parts. In addition we were promised feedback on the pre-delivery of the report. When the only feedback we got was that it was positive that we used roman numerals on the first pages of the report, we found it pretty useless.

Use-case estimation Cancelled

Course in presentation technique Unfortunately, we had our last customer meeting earlier this day. Since several people from SINTEF were invited to this final presentation, a lot of questions and discussions came up. Therefore the group was not able to make it to the seminar in time.

11.6.4 Info booklet

The info booklet provided in the beginning of the course is very good at providing the groups with all the needed information during the course. We found the parts concerning the introduction, administrative information, development and Scrum to be useful several times during the project.

When it comes to the appendices, we followed appendix A and C to create the initial structure of our report. On the first supervisor meeting afterwards, they wanted us to change the whole structure of the document. We think that when information like this is given, it should make sense and be a good way to structure the report.

Appendix B contained templates on how to do risk management and make time registration. We found both templates to be inconvenient to use on a daily basis. The table for time registration is particularly poor as it is almost impossible to make this kind of template work with a spreadsheet.

Good templates on how to make agendas, meeting minutes and weekly status reports were also missing. We believe that providing the groups with Excel spreadsheets for time registration, as well as good templates for different documents will only improve the learning process for the students. In our case, we used a lot of time making all templates and documents look like the supervisors wanted, instead of focusing on implementation and the report.

11.6.5 Improvements

The overall impression of the course is good, and we hope the customer driven project course will continue in the future for other students.

Our group think the quality of the seminars should be improved. The seminars should also be introduced at the correct time. The way it is now, it seems random when the different seminars are held. A similar course at Industrial Economy at NTNU has gathered all seminars to an intensive week in the beginning of the project. This way, the students have all background information when they start on the project itself.

The facilities at P15 are also worth mentioning. First of all, it seems that there is not enough space when all the groups in the course are present. Also, in a course like this, students do a lot of work in the evenings. The ventilation in the evenings are turned off, which leads to poor air quality. Most of the computers on the top floor of P15 were not working properly. When computers are present in the rooms, and administrated by NTNU, they should work.

11.7 Conclusion

Looking back at the course and the project as a whole, we are very satisfied with most aspects. We were given a challenging task and it required a lot of time. The relationship with our customer was good and we gained a lot of experience and interest in the aquacultural field.

Concerning the development process, we developed a very good team spirit where all members where equally accepted. We had no international students on the group and even though none of the team members knew each other before the project, we made great connection. Our group also had the privilege of avoiding internal conflicts.

The experience with Scrum was very positive, and all team members saw the effect of agile development where you have frequent meetings with the customer to confirm the running prototypes and make adjustments. Our group completed four sprints of two weeks, and are satisfied with this choice. The effect of having a daily scrum meeting contributed to involving all team members in the project doing important work.

A system called Trac helped us keep track of the project progress, and we are all confident that such tools are important in project management. We would like to see even more features in the provided Trac system. The active use of burndown charts also contributed to keeping track of progress in each sprint.

Happily, the final product exceeded the groups expectations when it came to functionality and quality. The software still has minor bugs as it is delivered, but the biggest ones have been removed. Our customer has also stated that they are impressed by the results. They are certain that advanced simulation software will be the future for optimising fish farms and other aquacultural equipment.

The weakest part of our results from the project is probably report. By working intensively near the end of the project we managed to put together a good document reflecting our work. Based on guidelines from the supervisors, the overall structure of the report should be satisfying. The supervisors have given us valuable feedback on the weekly supervisor meetings during the whole course. Communication with them have worked well and they have been very helpful.

Regarding the course, it is clear to us that there are large variations in the complexity of the different assignments. Our task was, in our opinion, one of the more complex tasks, and we ended up using a lot of time on implementation. This affected our report more than it would have done if we had a simpler task. The course as a whole was very exciting, and the information booklet provided an overview of important dates.

Possible improvements for the course is the quality of seminars and the time at which they are held. In addition the facilities on P15 are deficient, and working conditions there are sometimes blameworthy.

Overall, we are satisfied with both the course and the results we produced. We have learned a lot that is not possible to learn from any other ordinary course in the Computer Science programme.

Chapter 12

User Documentation

12.1 Introduction

In this chapter we will describe how to use the application. Since it has been made in very short time and can be considered more or less like a prototype, the usability is not as good as other commercial applications of this kind. Therefore, it can be a good idea to take a look at this guide before rushing into using the application. Another reason for reading this is that the program is not particularly stable, and wrong can easily lead to crashes. So keep on reading!

12.2 System requirements

12.2.1 Operating System

The application have been tested for Windows XP, Vista and Windows 7. We had no problems running the application on any of the operating systems.

12.2.2 Software prerequisites

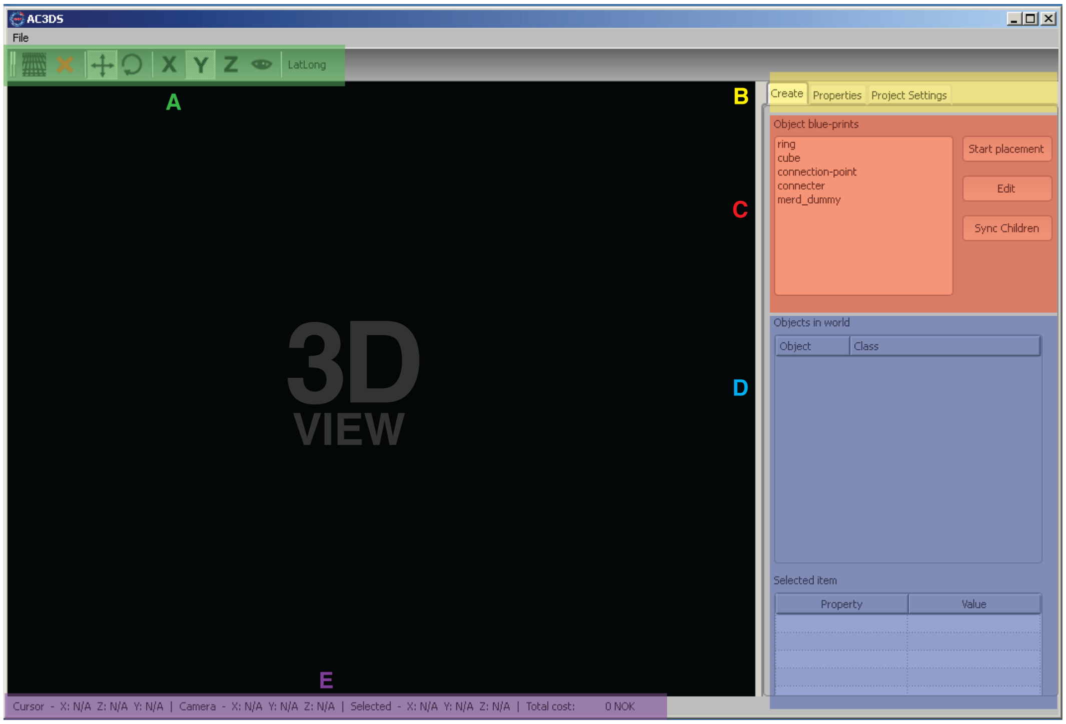
Our program is for the moment set up for using DirectX 9 or better for 3D rendering, but it is possible to customise it to use OpenGL instead. You will also need Microsoft's runtime environment for Visual C++, Visual C++ Redistributable Package 2008. You can download it from Microsoft's home pages:

`http://www.microsoft.com/downloads/details.aspx?FamilyID=9b2da534-3e03-4391-8a4d-074b9f2bc1bf`

12.3 User manual

12.3.1 The user interface

When you start the program, it shows an empty 3D window and the rest of the user interface around it, ready to use. We hope it is as intuitive and self-explanatory for every other user as it is for us. Unfortunately, that's probably not the case, so we have made this user manual to help you. Below you can see a screenshot of the start-up screen. We have labeled the different sections colours and legends.



ring
cube
connection-point
connecter
merd_dummy

Cursor - X: N/A Z: N/A Y: N/A | Camera - X: N/A Y: N/A Z: N/A | Selected - X: N/A Y: N/A Z: N/A | Total cost: 0 NOK

Figure 12.1: **A:** Main toolbar, **B:** Tab selector, **C:** Object Blue-prints, **D:** World objects and properties for selected item, **E:** Status bar

The main toolbar(**A**) consists of 9 different buttons. There is a short description of each button below.

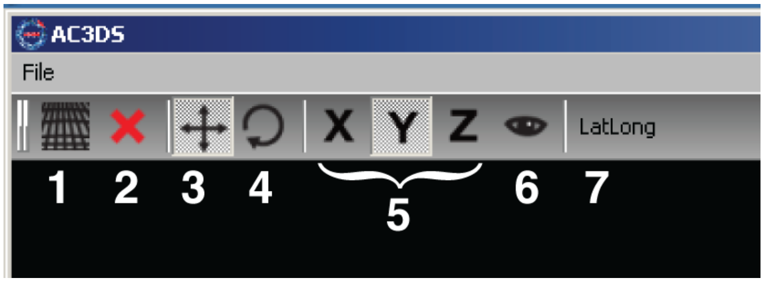


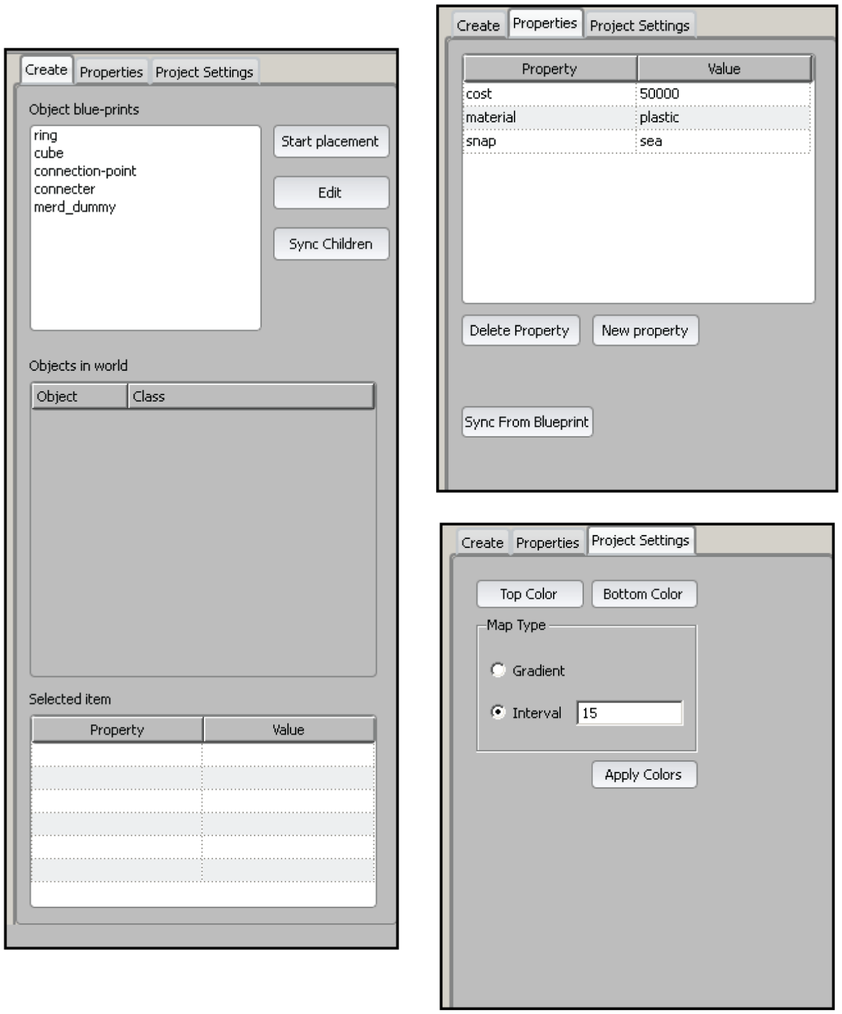
Figure 12.2: **1:** New project/Load map **2:** Delete selected object **3:** Move **4:** Rotate **5:** Which axes/axis to move or rotate in/about **6:** Look at selected object - Camera centers the selected object **7:** Show coordinates in Latitudes and Longitudes

The tabs

Create-tab As you can see in the tab selector(**B**), there is three different tabs: “Create”, “Properties” and “Project Settings”. The create-tab is the one that is shown by default. In this tab, you can see what objects you have(**C**), what object you already have placed(**D**), and the specific placement and orientation information for a selected object.

Properties-tab Under the “Properties”-tab you find a list over all the metadata for a selected object. You can also change and add as much data you want here.

Project Settings-tab In this tab you can customise the colours for the map. You can set a top and bottom-colour and set if it shall be a smooth gradient, or blended with specified intervals.



Property	Value
cost	50000
material	plastic
snap	sea

○ Gradient

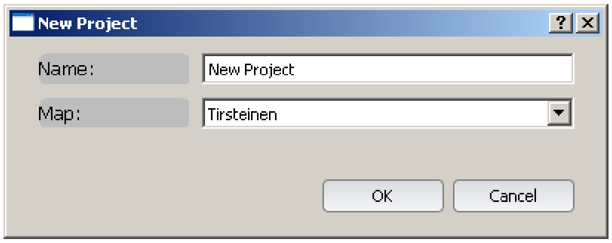
● Interval 15

Figure 12.3: The tabs

12.3.2 Tasks

In this section we will explain in details how to do some of the most common tasks in AC3Ds.

Start a new project



Name: New Project

Map: Tirsteinen

Figure 12.4: New Project

1. When you start the program, go to file in the top menu and choose “New...”.
2. Then you get up a dialog box, where you write in the preferred name of the project, and chooses what map/area you want to use.
3. When you press “OK”, the map data gets loaded and you can start placing objects in the environment.

Tip: You can also use the “create new project/load map”-button instead of using the file menu. It’s located left-most in the toolbar.

Navigation

It is quite intuitive to move around in the 3D-environment. To move the camera, you hold the left mouse button while you drag it over the map. If you want to rotate the view you hold in the right mouse button while dragging. If you just want to move the camera forward, you can scroll upward, and vice versa for zooming out. If you want to look at and rotate around a selected object in the view, you can press the look at-button illustrated as an eye in the toolbar. The the object you have selected gets centered in the view, and when you rotate, you rotate around it.

Insert object

1. Choose an object from the “object blue-prints”-box.
2. Press start placement

3. Move the cursor over to the 3D window, and you will see that the selected type of object follows the mouse. When you left click, you place one instance of the object. You can still hold the left mouse-button in without placing an object if you need to pan the camera. This is also the case for the right mouse button, if you hold this one in you rotates the view.

4. To end the placing of object, just right-click.

12.3.2.1 Create sub-objects

The program uses a hierarchy to connect objects so that one can move many objects at the same time. If you move an object, every sub object follows. This is very useful when it comes to making connection point for object, because you as often wants these points to follow the object they shall be connected to. To make a sub object, simply have the main object selected before you start placing a new object, and they gets automatically children of the selected one. You can always afterwards select the child and move this independently, but it will always follow when you move the parent.

12.3.2.2 Create anchor lines

Anchor lines are for the moment nothing more than just splines bended in a nice curve between two connection points. So to be able to create anchor lines, you need to have placed out at least two connection points. Regardless of their parent object(if any at all), you just have to click start and end connection point you want to use, and the program creates the anchor line for you. If you want to delete it, you can select it in the object-tree on the right.

Bibliography

[1] Andreas S. Andreou. Requirement engineering process. Internet, 2009.

[2] Autodesk. Autodesk 3ds max. http://usa.autodesk.com/adsk/servlet/pc/index?id=13567410&siteID=123112, 09 2009.

[3] E.J. Braude. *Software engineering: an object-oriented perspective*. John Wiley & Sons, Inc. New York, NY, USA, 2000.

[4] OGRE 3D Community. Settingupanapplication. http://www.ogre3d.org/wiki/index.php/SettingUpAnApplication, 10 2009.

[5] OSG Community. Openscenegraph. http://www.openscenegraph.org, 09 2009.

[6] cplusplus.com. C++. http://www.cpluscplus.com, 09 2009.

[7] Create. Create - aquaculture for the future. Annual Report 2008, Brattørkaia 17B, 2009.

[8] developer.com. Pattern summaries: Chain of responsibility. http://www.developer.com/java/other/article.php/631261/Pattern-Summaries-Chain-of-Responsibility.htm, 08 2003.

[9] Data Object Factory. Observer. REFERANSE:http://www.dofactory.com/Patterns/PatternObserver.aspx.

[10] SINTEF Fisheries and Aquaculture. Et hav av muligheter. Brattørkaia 17B, 2009.

[11] The Eclipse Foundation. Eclipse. http://www.eclipse.org/, 09 2009.

[12] Nikolaus Gebhardt. Irrlicht engine. http://irrlicht.sourceforge.net/, 09 2009.

[13] Henrik Kinberg. *Scrum and XP from the Trences - How we do scrum*. InfoQ, 2009.

[14] Boost Libraries. Boost. `http://www.boost.org/`, 09 2009.

[15] Torus Knot Software Ltd. Ogre. `http://www.ogre3d.org`, 09 2009.

[16] George M. Marakas. *System Analysis Design*. McGrawHill International Edition, 2006.

[17] Microsoft. C. `http://msdn.microsoft.com/en-us/vcsharp/default.aspx`, 09 2009.

[18] Microsoft. Microsoft visual studio. `http://msdn.microsoft.com/en-us/vstudio/default.aspx`, 09 2009.

[19] Microsoft. .net. `http://www.microsoft.com/NET/`, 09 2009.

[20] Sun Microsystems. Java. `http://www.java.com`, 09 2009.

[21] Sun Microsystems. Netbeans. `http://netbeans.org/`, 09 2009.

[22] F.B.J. Mittleman, H.R.C. Silva, and G. Taubin. The Ball-Pivoting Algorithm for Surface Reconstruction. `http://www.research.ibm.com/vistechnology/pdf/bpa_tvcg.pdf`, 10 2009.

[23] Nokia. Qt. `http://qt.nokia.com/`, 09 2009.

[24] PATS. Telenor thyolt weather station. `http://wiki.pats.no/index.php/Telenor_Tyholt_Weather_Station`, 10 2009.

[25] Trygve M. H. Reenskaug. Mvc xerox parc 1978-79. `http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html`.

[26] Linda Rising and Norman S. Janoff. The Scrum software development process for small teams. *IEEE Software*, page 8, July/August 2000.

[27] Joachim Rossberg. *Pro Visual Studio Team System Application Lifecycle Management*. Apress, 2008.

[28] MAF Information Services. Writing good milestones.

[29] SINTEF. Ace – viable technological solutions for tomorrow's aquaculture industry. `http://www.aceaquaculture.com/english/`, 10 2009.

[30] SINTEF. Fiskeri- og kystministeren åpnet sealab sso. `http://www.sintef.no/Marin/Fiskeri-og-havbruk-AS/Nyhetsarkiv/Nyhetsarkiv/Fiskeriministeren-apner-SeaLab-SSO/`, 10 2009.

[31] Brett Slocum. C++ coding standards. `http://www.weirdrealm.com/prog/cppstds.html`, 10 2009.

[32] Edgewall Software. The trac project. http://trac.edgewall.org/, 10 2009.

[33] Mountain Goat Software. Introduction to Scrum agile process. http://www.mountaingoatsoftware.com/scrum, 10 2009.

[34] Olex Software. Olex: Teknologi. http://olex.no/teknikk.html, 11 2009.

[35] Telenor. Pats - program for advanced telecom services. http://wiki.pats.no/index.php/Main_Page, 09 2009.

[36] Thushara Wijewardena. Comparison of Scrum vs pmbok. http://projectized.blogspot.com/2008/09/comparison-of-scrum-vs-pmbok_19.html, 10 2009.

[37] Wikipedia. Standard template library. http://en.wikipedia.org/wiki/Standard_Template_Library, 10 2009.

Appendices

Appendix A

Original Project Description

Title: Aquaculture 3D simulator SINTEF Fisheries and Aquaculture (SFH) have broad technological competence and domain knowledge in the field of utilisation of renewable marine resources. SFH also hosts CREATE, a center for research- driven innovation (SFI) in aquaculture.

One of our recent investments is the ICT laboratory SeaLab SSO (Surveillance, Simulation and Operation) in our office at Brattøra. This is a flexible infrastructure for development and testing and contains machines for heavy computations, servers, graphical workstations, HD projectors and a Smartboard. SeaLab SSO communicates with our large scale research aquaculture plant ACE (Aquaculture Engineering). Realtime measurement data and video are transferred from ACE to SeaLab SSO. Through cooperation with Telenor, SeaLab SSO is also connected to the PATS (Program for Advanced Telecom Services) laboratory at the Telenor office at Tyholt.

The project is to develop a graphical 3D demonstrator showing the potential of the new SSO laboratory. We imagine a prototype of a simulator/game that can visualise terrain from the area where ACE is situated, the user should then be able to place and manipulate aquaculture equipment in this terrain. Future expansions will be to connect the system to input from sensor data and decision support systems. Thus, a good overall architecture and interfaces for future expansions will be important. The task can be broken down to:

- Choosing a 3D engine for visualisation, we have good experience with an open source engine, but are willing to consider other solutions if they should prove more suitable.

- Design an architecture with interfaces that can import both measured and simulated data to visualise equipment behaviour, and export data to external systems for decision support.
- Develop a prototype. In this project we are mainly concerned with visualisation and the graphical elements. We can supply map data and 3D models of equipment.

SeaLab SSO is an important facility and frame for our activities. The laboratory and equipment will be available for the student group should they wish to use it.

Appendix B

Templates

Appendix B contains the following templates:

- B.1 - Customer meeting agenda
- B.2 - Customer meeting minutes
- B.3 - Supervisor meeting agenda
- B.4 - Supervisor meeting minutes
- B.5 - Weekly status report

B.1 Customer meeting agenda

Agenda - customer meeting, 12.11.2009

Project

Group 6, AC3DS - Sintef Fisheries and Aquaculture

Prepared by

John Doe

Invited

Customer: Finn Olav Bjørnson, Gunnar Senneset, Carl Fredrik Sørensen
Team: Audun Bjørkøy, Ole Kristian Grasheim, Samson Valland, Magnus Westergaard, Per Øyvind Øygard

Place and time

SINTEF Brattøra, 12.11.2009 at 14:00

Topics

1. Approval of agenda
2. Approval of meeting minutes from last customer meeting
3. Demo of results from sprint 4
4. Other issues

Comments

B.2 Customer meeting minutes

Meeting minutes - customer meeting, DD.MM.2009

Project

Group 6, AC3DS - Sintef Fisheries and Aquaculture

Prepared by

Audun Bjørkøy

Attending

Customer: Finn Olav Bjørnson, Gunnar Senneset, Carl Fredrik Sørensen
Team: Audun Bjørkøy, Ole Kristian Grasheim, Samson Valland, Magnus Westergaard, Per Øyvind Øygard

Place and time

SINTEF Brattøra, DD.MM.2009 at HH:MM

Notes

1. **Approval of agenda**
   -
2. **Comments on the minutes from last customer meeting or other meetings**
   -
3. **Approval of meeting minutes from last advisory meeting**
   -
4. **Topic 1**
5. **Topic 2**
6. **Topic 3**
7. **Other issues?**
   -

B.3 Supervisor meeting agenda

Agenda - supervisor meeting, DD.MM.2009

Project

Group 6, AC3DS - Sintef Fisheries and Aquaculture

Prepared by

Audun Bjørkøy

Invited

Supervisor: Meng Zhu
Assistant Supervisor: Bian Wu
Team: Audun Bjørkøy, Ole Kristian Grasheim, Samson Valland, Magnus Westergaard, Per Øyvind Øygard

Place and time

ITV 242, DD.MM.2009 at 14:00-15:00

Topics

1. Approval of agenda
2. Comments on the minutes from last supervisor meeting
3. Approval of meeting minutes from last supervisor meeting or other meeting
4. Approval of the status report
5. Review/approval of attached phase documents
    - Overall Comments
    - Comments to each chapter
6. Topic 1
7. Topic 2
8. Topic 3
9. Other issues

B.4 Supervisor meeting minutes

Meeting minutes - supervisor meeting, DD.MM.2009

Project

Group 6, AC3DS - Sintef Fisheries and Aquaculture

Prepared by

Audun Bjørkøy

Attending

Supervisor: Meng Zhu
Assistant Supervisor: Bian Wu
Team: Audun Bjørkøy, Ole Kristian Grasheim, Samson Valland, Magnus Westergaard, Per Øyvind Øygard

Place and time

ITV 242, DD.MM.2009 at 14:00-15:00

Notes

1. Approval of agenda

-

2. Comments on the minutes from last customer meeting or other meetings

-

3. Approval of meeting minutes from last advisory meeting

-

4. Approval of the status report

-

5. Review/approval of attached phase documents

Overall Comments:

-

B.5 Weekly status report

September 23, 2009

Weekly status report, group 6

Project

Group 6, AC3DS - Sintef Fisheries and Aquaculture

Prepared by

Audun Bjørkøy

Attending

Supervisor: Meng Zhu
Assistant Supervisor: Bian Wu
Team: Audun Bjørkøy, Ole Kristian Grasheim, Samson Valland, Magnus Westergaard, Per Øyvind Øygard

Summary

Activities planned for last week

Activites accomplished last week

Activities planned for next week

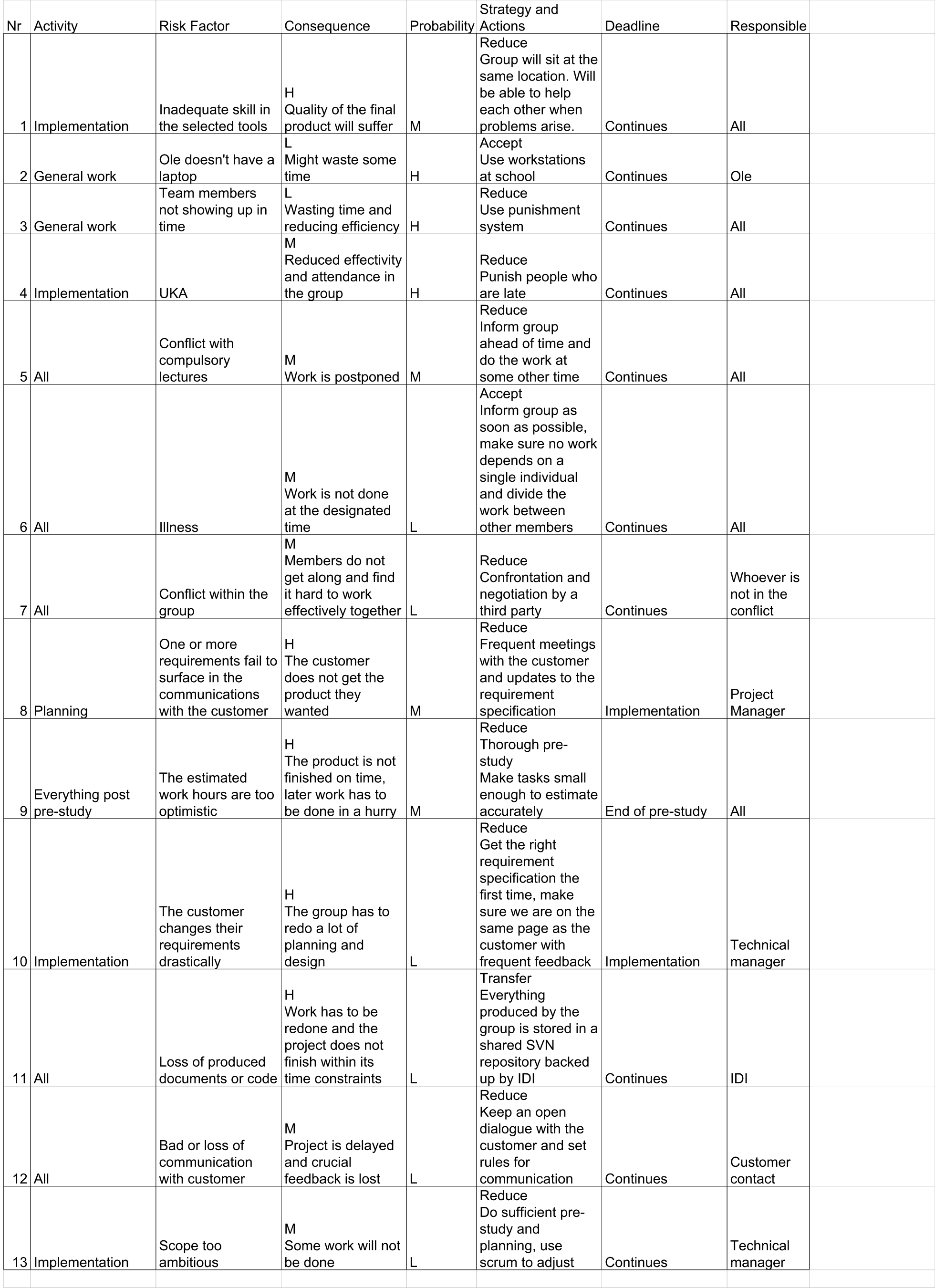
New issues

Old issues

Timesheet

Appendix C

Risks



Nr	Activity	Risk Factor	Consequence	Probability	Strategy and Actions	Deadline	Responsible
1	Implementation	Inadequate skill in the selected tools	H Quality of the final product will suffer	M	Reduce Group will sit at the same location. Will be able to help each other when problems arise.	Continues	All
2	General work	Ole doesn't have a laptop	L Might waste some time	H	Accept Use workstations at school	Continues	Ole
3	General work	Team members not showing up in time	L Wasting time and reducing efficiency	H	Reduce Use punishment system	Continues	All
4	Implementation	UKA	M Reduced effectivity and attendance in the group	H	Reduce Punish people who are late	Continues	All
5	All	Conflict with compulsory lectures	M Work is postponed	M	Reduce Inform group ahead of time and do the work at some other time	Continues	All
6	All	Illness	M Work is not done at the designated time	L	Accept Inform group as soon as possible, make sure no work depends on a single individual and divide the work between other members	Continues	All
7	All	Conflict within the group	M Members do not get along and find it hard to work effectively together	L	Reduce Confrontation and negotiation by a third party	Continues	Whoever is not in the conflict
8	Planning	One or more requirements fail to surface in the communications with the customer	H The customer does not get the product they wanted	M	Reduce Frequent meetings with the customer and updates to the requirement specification	Implementation	Project Manager
9	Everything post pre-study	The estimated work hours are too optimistic	H The product is not finished on time, later work has to be done in a hurry	M	Reduce Thorough pre-study Make tasks small enough to estimate accurately	End of pre-study	All
10	Implementation	The customer changes their requirements drastically	H The group has to redo a lot of planning and design	L	Reduce Get the right requirement specification the first time, make sure we are on the same page as the customer with frequent feedback	Implementation	Technical manager
11	All	Loss of produced documents or code	H Work has to be redone and the project does not finish within its time constraints	L	Transfer Everything produced by the group is stored in a shared SVN repository backed up by IDI	Continues	IDI
12	All	Bad or loss of communication with customer	M Project is delayed and crucial feedback is lost	L	Reduce Keep an open dialogue with the customer and set rules for communication	Continues	Customer contact
13	Implementation	Scope too ambitious	M Some work will not be done	L	Reduce Do sufficient pre-study and planning, use scrum to adjust	Continues	Technical manager

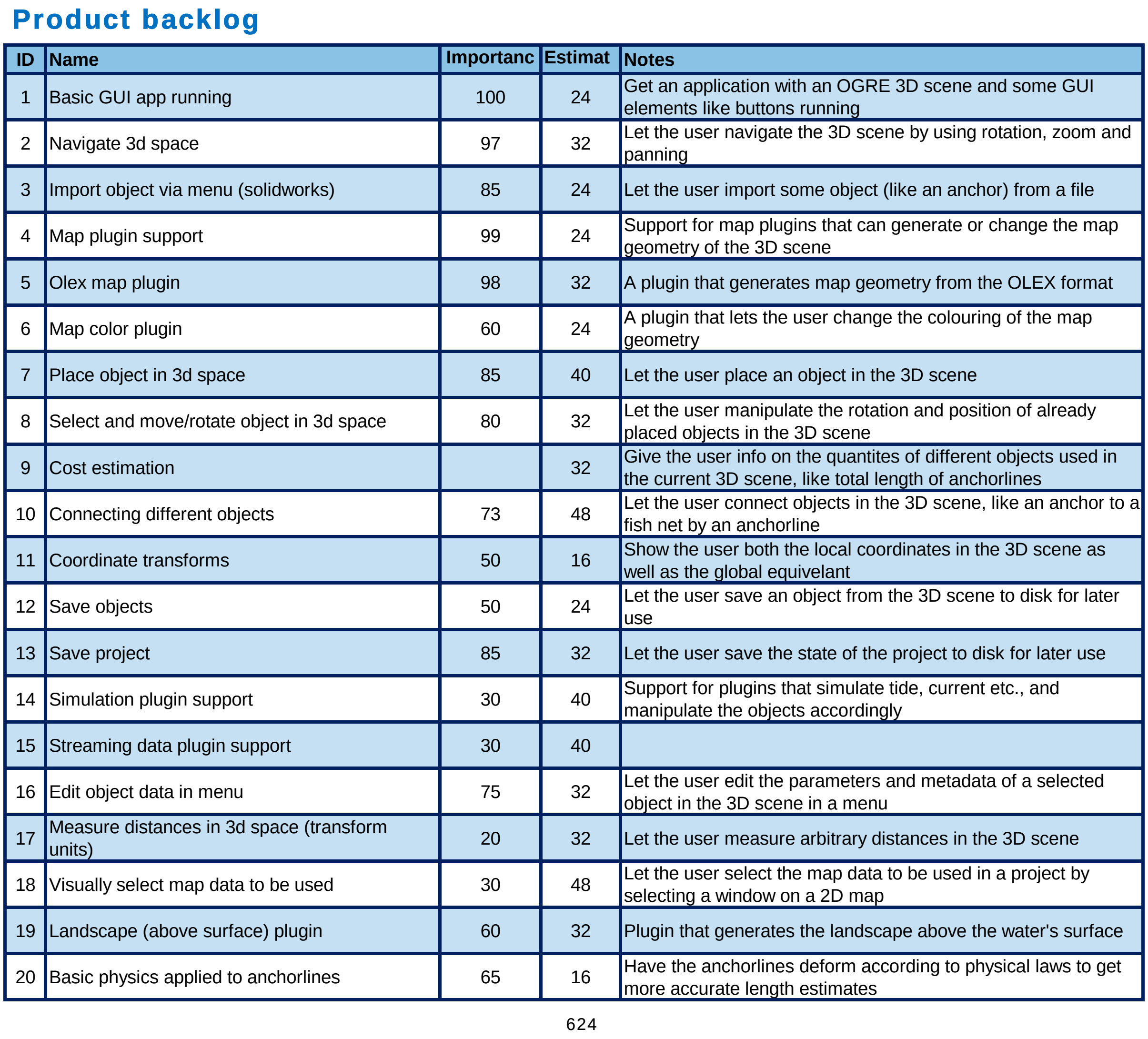
Figure C.1: Risk table

Appendix D

Product backlog

D.1 Initial product backlog

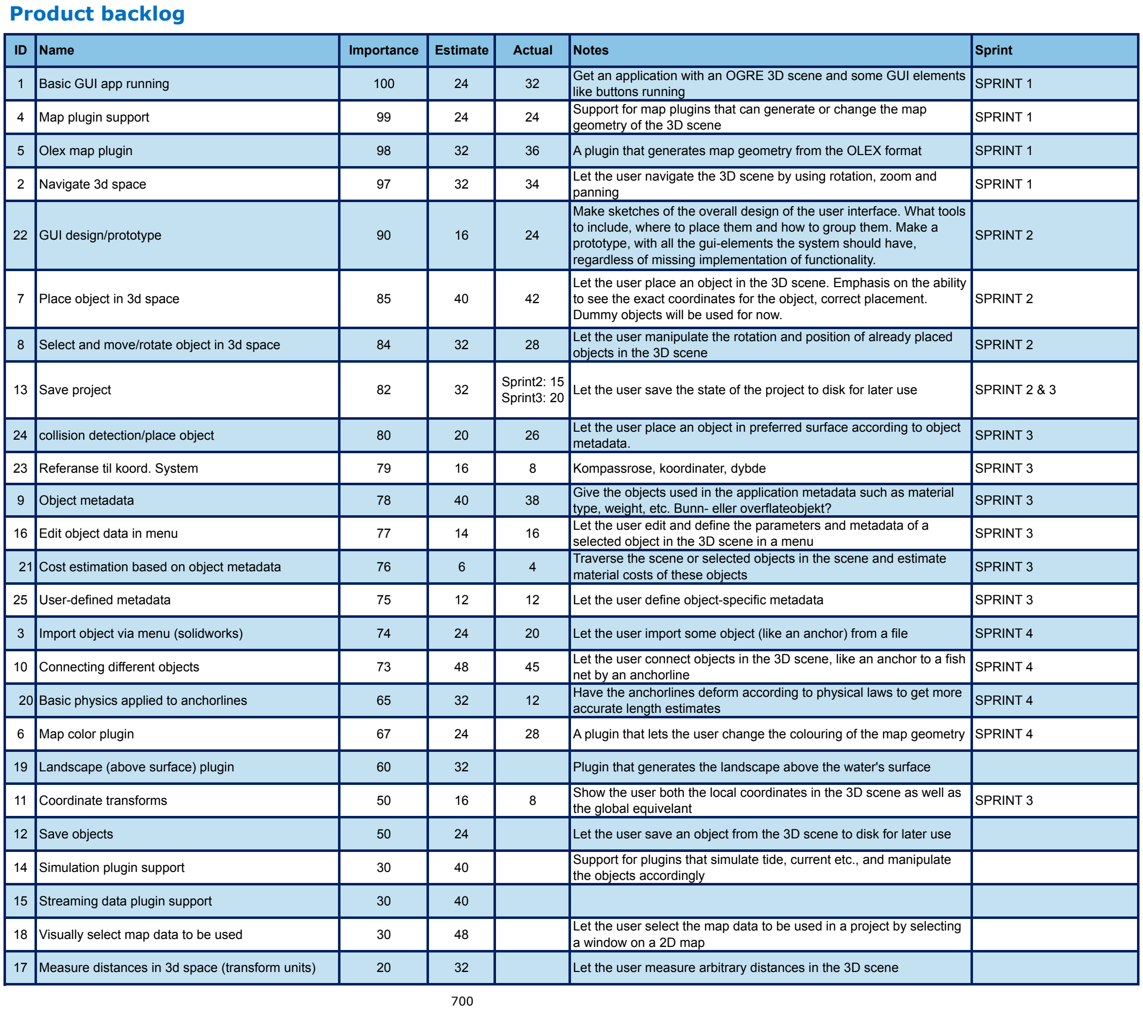
D.2 Complete product backlog



Product backlog

ID	Name	Importanc	Estimat	Notes
1	Basic GUI app running	100	24	Get an application with an OGRE 3D scene and some GUI elements like buttons running
2	Navigate 3d space	97	32	Let the user navigate the 3D scene by using rotation, zoom and panning
3	Import object via menu (solidworks)	85	24	Let the user import some object (like an anchor) from a file
4	Map plugin support	99	24	Support for map plugins that can generate or change the map geometry of the 3D scene
5	Olex map plugin	98	32	A plugin that generates map geometry from the OLEX format
6	Map color plugin	60	24	A plugin that lets the user change the colouring of the map geometry
7	Place object in 3d space	85	40	Let the user place an object in the 3D scene
8	Select and move/rotate object in 3d space	80	32	Let the user manipulate the rotation and position of already placed objects in the 3D scene
9	Cost estimation		32	Give the user info on the quantites of different objects used in the current 3D scene, like total length of anchorlines
10	Connecting different objects	73	48	Let the user connect objects in the 3D scene, like an anchor to a fish net by an anchorline
11	Coordinate transforms	50	16	Show the user both the local coordinates in the 3D scene as well as the global equivelant
12	Save objects	50	24	Let the user save an object from the 3D scene to disk for later use
13	Save project	85	32	Let the user save the state of the project to disk for later use
14	Simulation plugin support	30	40	Support for plugins that simulate tide, current etc., and manipulate the objects accordingly
15	Streaming data plugin support	30	40	
16	Edit object data in menu	75	32	Let the user edit the parameters and metadata of a selected object in the 3D scene in a menu
17	Measure distances in 3d space (transform units)	20	32	Let the user measure arbitrary distances in the 3D scene
18	Visually select map data to be used	30	48	Let the user select the map data to be used in a project by selecting a window on a 2D map
19	Landscape (above surface) plugin	60	32	Plugin that generates the landscape above the water's surface
20	Basic physics applied to anchorlines	65	16	Have the anchorlines deform according to physical laws to get more accurate length estimates

624



Product backlog

ID	Name	Importance	Estimate	Actual	Notes	Sprint
1	Basic GUI app running	100	24	32	Get an application with an OGRE 3D scene and some GUI elements like buttons running	SPRINT 1
4	Map plugin support	99	24	24	Support for map plugins that can generate or change the map geometry of the 3D scene	SPRINT 1
5	Olex map plugin	98	32	36	A plugin that generates map geometry from the OLEX format	SPRINT 1
2	Navigate 3d space	97	32	34	Let the user navigate the 3D scene by using rotation, zoom and panning	SPRINT 1
22	GUI design/prototype	90	16	24	Make sketches of the overall design of the user interface. What tools to include, where to place them and how to group them. Make a prototype, with all the gui-elements the system should have, regardless of missing implementation of functionality.	SPRINT 2
7	Place object in 3d space	85	40	42	Let the user place an object in the 3D scene. Emphasis on the ability to see the exact coordinates for the object, correct placement. Dummy objects will be used for now.	SPRINT 2
8	Select and move/rotate object in 3d space	84	32	28	Let the user manipulate the rotation and position of already placed objects in the 3D scene	SPRINT 2
13	Save project	82	32	Sprint2: 15 Sprint3: 20	Let the user save the state of the project to disk for later use	SPRINT 2 & 3
24	collision detection/place object	80	20	26	Let the user place an object in preferred surface according to object metadata.	SPRINT 3
23	Referanse til koord. System	79	16	8	Kompassrose, koordinater, dybde	SPRINT 3
9	Object metadata	78	40	38	Give the objects used in the application metadata such as material type, weight, etc. Bunn- eller overflateobjekt?	SPRINT 3
16	Edit object data in menu	77	14	16	Let the user edit and define the parameters and metadata of a selected object in the 3D scene in a menu	SPRINT 3
21	Cost estimation based on object metadata	76	6	4	Traverse the scene or selected objects in the scene and estimate material costs of these objects	SPRINT 3
25	User-defined metadata	75	12	12	Let the user define object-specific metadata	SPRINT 3
3	Import object via menu (solidworks)	74	24	20	Let the user import some object (like an anchor) from a file	SPRINT 4
10	Connecting different objects	73	48	45	Let the user connect objects in the 3D scene, like an anchor to a fish net by an anchorline	SPRINT 4
20	Basic physics applied to anchorlines	65	32	12	Have the anchorlines deform according to physical laws to get more accurate length estimates	SPRINT 4
6	Map color plugin	67	24	28	A plugin that lets the user change the colouring of the map geometry	SPRINT 4
19	Landscape (above surface) plugin	60	32		Plugin that generates the landscape above the water's surface	
11	Coordinate transforms	50	16	8	Show the user both the local coordinates in the 3D scene as well as the global equivelant	SPRINT 3
12	Save objects	50	24		Let the user save an object from the 3D scene to disk for later use	
14	Simulation plugin support	30	40		Support for plugins that simulate tide, current etc., and manipulate the objects accordingly	
15	Streaming data plugin support	30	40			
18	Visually select map data to be used	30	48		Let the user select the map data to be used in a project by selecting a window on a 2D map	
17	Measure distances in 3d space (transform units)	20	32		Let the user measure arbitrary distances in the 3D scene	

700

Appendix E

Sprint backlogs

E.1 Sprint 1 backlog

E.2 Sprint 2 backlog

E.3 Sprint 3 backlog

E.4 Sprint 4 backlog

Ticket in trac	Summary	Component	Importance		Days in											
				Date:	21.09	22.09	23.09	24.09	25.09	26.09	27.09	28.09	29.09	30.09	01.10	02.10
				Effort left:	162	153	135	124	112	105	99	93	68	50	36	4
Backlog item 1	Basic GUI app running			32												
#25	Design klassehierarki	Basic GUI app	100		10	10	10	10	5	5	5	3	0	0	0	0
#26	Få OGRE opp i Qt-vindu	Basic GUI app	95		10	7	4	0	0	0	0	0	0	0	0	0
#27	Bygg gui med qt	Basic GUI app	95		4	4	4	4	4	4	4	2	0	0	0	0
#28	Basic interaktivitet (knapper, slidere ...)	Basic GUI app	90		4	4	4	4	4	4	4	4	4	0	0	0
#30	Test	Basic GUI app	80		4	4	4	4	4	4	4	4	4	4	4	0
Backlog item 4	Map plugin support			24												
#31	Design pluginmodell	Map plugin support	90		16	16	13	12	8	4	0	0	0	0	0	0
#32	Mockup plugin	Map plugin support	85		4	2	0	0	0	0	0	0	0	0	0	0
#33	Test map plugin support	Map plugin support	80		4	4	4	4	4	4	4	4	4	4	4	0
Backlog item 5	OLEX map plugin			32												
#34	Få OLEX punktdata inn i OGRE	OLEX plugin	90		4	4	0	0	0	0	0	0	0	0	0	0
#35	Bygg flate fra punkter	OLEX plugin	85		20	18	16	12	12	12	12	12	12	4	0	0
#36	Lagre flate	OLEX plugin	70		4	4	4	4	4	4	4	4	4	4	4	0
#37	Visualiser flate	OLEX plugin	85		2	2	2	2	2	2	2	2	2	2	0	0
#38	Test OLEX plugin	OLEX plugin	50		2	2	2	2	2	2	2	2	2	2	2	0
Backlog item 2	Navigate 3D space			32												
#40	Få til input via Qt/OGRE	Navigate 3D space	60		6	6	6	6	6	6	4	4	0	0	0	0
#41	Implementer rotasjon	Navigate 3D space	50		8	8	8	8	8	8	8	8	4	4	0	0
#42	Implementer pan	Navigate 3D space	50		8	8	8	8	8	8	8	8	4	4	4	0
#43	Implementer zoom	Navigate 3D space	50		8	8	8	8	8	8	8	8	4	0	0	0
#44	Test navigasjon	Navigate 3D space	40		2	2	2	2	2	2	2	2	2	2	2	0
Not in BL	Report writing			42												
#71	Write sprint 1 introduction	Report	80		2	2	2	2	0	0	0	0	0	0	0	0
#72	Design sprint1 backlog	Report	100		4	2	0	0	0	0	0	0	0	0	0	0
#73	Write Sprint 1 - Design chapter	Report	90		8	8	6	4	3	0	0	0	0	0	0	0
#74	Write Sprint 1 - Implementation chapter	Report	80		12	12	12	12	12	12	12	12	12	12	8	0
#75	Write Sprint 1 - Testing chapter	Report	80		8	8	8	8	8	8	8	6	2	0	0	0
#76	Write Sprint 1 - Results chapter	Report	70		4	4	4	4	4	4	4	4	4	4	4	4
#77	Write Sprint 1- Evaluation chapter	Report	70		4	4	4	4	4	4	4	4	4	4	4	0

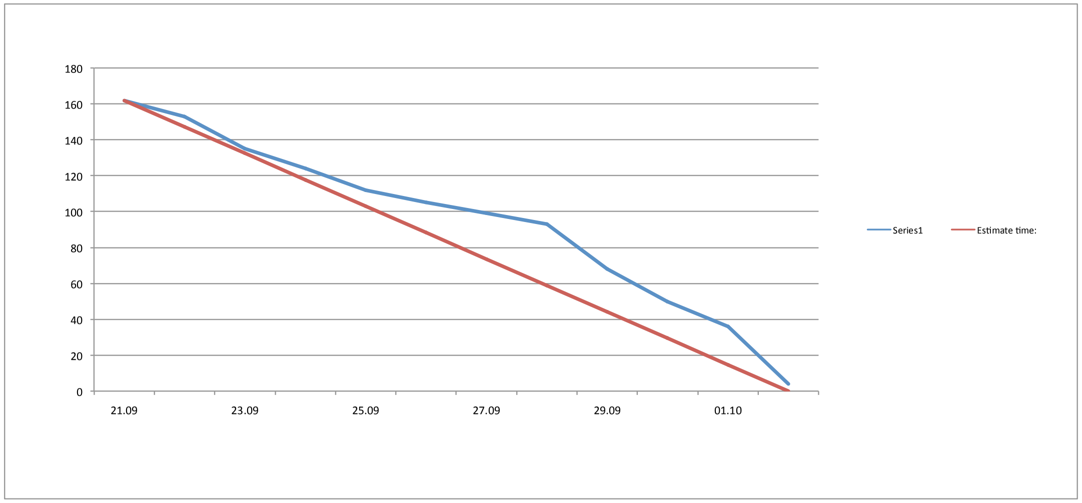
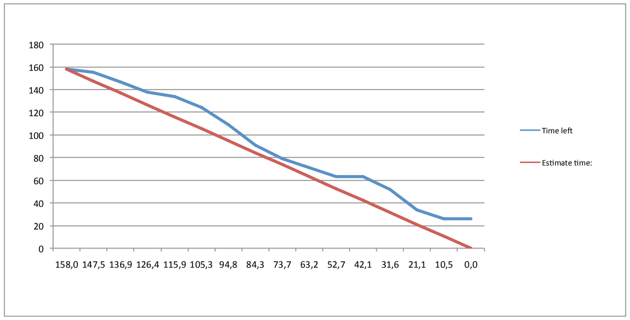


Figure E.1: Sprint 1 backlog and burndown chart

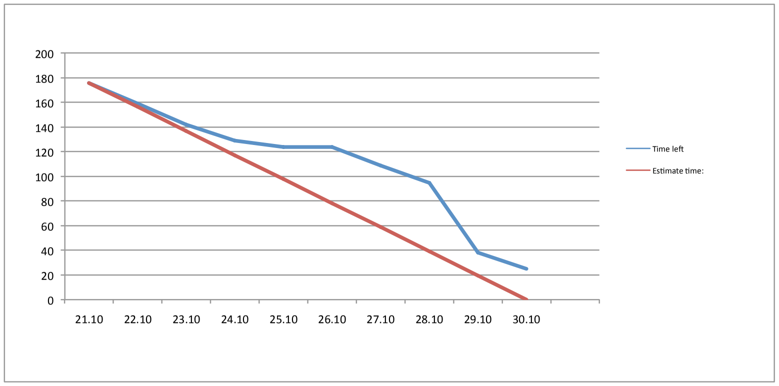
Ticket in trac	Summary	Component	Importance		Days in															
				Date:	05.10	06.10	07.10	08.10	09.10	10.10	11.10	12.10	13.10	14.10	15.10	16.10	17.10	18.10	19.10	20.10
				Effort left:	158	155	147	138	134	124	109	91	79	71	63	63	52	34	26	26
Backlog item 22	Gui design/prototype			16																
#82	Kartlegge verktøy i programmet	Gui design prototype	100		2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
#83	Skisser for use-case	Gui design prototype	100		4	4	2	2	0	0	0	0	0	0	0	0	0	0	0	0
#84	Qt-elementer som tilsvarer skissene	Gui design prototype	100		10	10	10	6	4	4	0	0	0	0	0	0	0	0	0	0
Backlog item 7	Place objects in 3D space			42																
#85	Lag havoverflate	Place objects in 3D space	90		10	10	10	10	10	8	8	6	0	0	0	0	0	0	0	0
#86	Finn koordinater til musepeker i scenen	Place objects in 3D space	85		8	8	4	4	4	4	0	0	0	0	0	0	0	0	0	0
#87	Få objekter til å følge musepeker i scenen	Place objects in 3D space	85		6	6	6	3	3	3	0	0	0	0	0	0	0	0	0	0
#88	Sett objekter i scenen vha museklikk	Place objects in 3D space	80		8	8	8	8	8	8	4	4	4	0	0	0	0	0	0	0
#89	Sett objekter i scenen vha numerisk input	Place objects in 3D space	80		10	10	10	10	10	10	10	10	6	6	6	6	6	2	0	0
Backlog item 5	Select and move/rotate object in 3D space			32																
#90	Select object	Select, move and rotate	75		4	4	4	4	4	4	4	0	0	0	0	0	0	0	0	0
#91	Visualise object selection	Select, move and rotate	70		4	4	4	4	4	4	4	0	0	0	0	0	0	0	0	0
#92	Visualise helper axes/circles around object	Select, move and rotate	60		8	8	8	8	8	8	8	8	8	4	0	0	0	0	0	0
#93	Interact with helper axes/circles	Select, move and rotate	60		16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
Backlog item 13	Save Project			28																
#78	Make project serializable	Save project	50		16	16	16	14	14	6	6	6	4	4	4	4	0	0	0	0
#79	Define file format to save project in	Save project	50		4	4	4	4	4	4	4	0	0	0	0	0	0	0	0	0
#80	Implement functions to write/read file format	Save project	50		8	8	8	8	8	8	8	4	4	4	0	0	0	0	0	0
Not in BL	Report writing			40																
#71	Write sprint 2 introduction	Report	80		2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0
#72	Design sprint2 backlog	Report	100		4	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
#73	Write Sprint 2 - Design chapter	Report	90		8	8	8	8	8	8	8	8	8	8	8	8	4	0	0	0
#74	Write Sprint 2 - Implementation chapter	Report	80		10	10	10	10	10	10	10	10	10	10	10	10	10	6	2	2
#75	Write Sprint 2 - Testing chapter	Report	80		8	8	8	8	8	8	8	8	8	8	8	8	8	4	4	4
#76	Write Sprint 2 - Results chapter	Report	70		4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	2
#77	Write Sprint 2- Evaluation chapter	Report	70		4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2



Estimat															
158	147,5	136,9	126,4	115,9	105,3	94,8	84,27	73,73	63,2	52,67	42,13	31,6	21,07	10,53	0

Figure E.2: Sprint 2 backlog and burndown chart

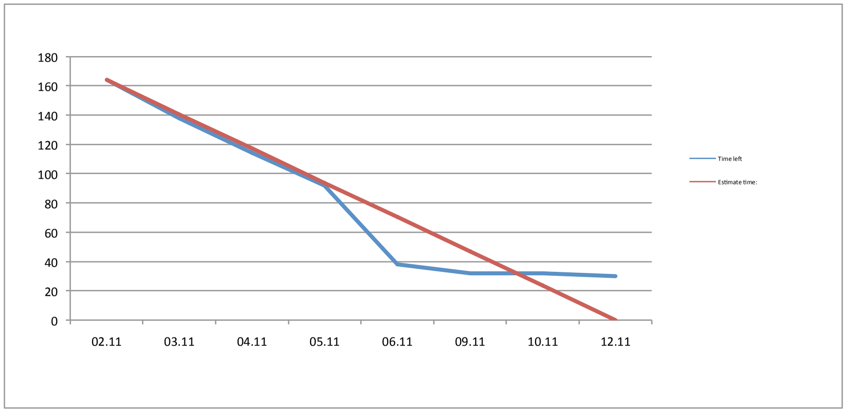
Ticket in trac	Summary	Component	Importance		Days in sprint / Effort left									
				Date:	21.10	22.10	23.10	24.10	25.10	26.10	27.10	28.10	29.10	30.10
				Effort left:	176	159	142	129	124	124	109	95	38	25
Backlog item 13	Save Project (continued from sprint 2)			10										
#80	Implement functions to write/read file format	Save project	100		10	6	4	2	2	2	2	2	0	0
#126	Test saving and loading of project	Save project	100		4	4	4	4	4	4	4	4	4	0
Backlog item 24	Collision detection/place object			20										
#102	Try intersect with static geometry	Col detection/place obj	100		4	0	0	0	0	0	0	0	0	0
#103	Change to heigthmap scenemanager	Col detection/place obj	100		6	6	0	0	0	0	0	0	0	0
#104	Get coordinats rayquery>worldfragment	Col detection/place obj	100		6	6	6	0	0	0	0	0	0	0
#105	Test collision detection	Col detection/place obj	100		4	4	4	4	0	0	0	0	0	0
Backlog item 23	Reference to coordinate system			16										
#106	Use coordMgr	Reference to coord system	90		6	6	6	6	6	6	0	0	0	0
#107	Compass rose	Reference to coord system	90		4	3	0	0	0	0	0	0	0	0
#108	GUI to switch between local/global coordinates	Reference to coord system	85		2	2	2	2	2	2	2	2	0	0
#109	Test reference to coordinate system	Reference to coord system	85		4	4	3	3	3	3	3	2	2	0
Backlog item 9	Object Metadata			40										
#125	Structurise code	Object metadata	90		16	10	5	0	0	0	0	0	0	0
#110	Make list with data for each object	Object metadata	75		10	10	10	10	10	10	5	5	0	0
#111	Make metadata serializable	Object metadata	70		10	10	10	10	10	10	10	10	0	0
#112	Test metadata	Object metadata	60		4	4	4	4	4	4	4	4	0	0
Backlog item 16	Edit metadata in menu			20										
#113	Make gui object tab with metadata	Edit metadata in menu	75		4	4	4	4	4	4	2	0	0	0
#114	Make metadata editable through menu	Edit metadata in menu	70		4	4	4	4	4	4	4	2	0	0
#115	Object blueprint (?)	Edit metadata in menu	60		3	3	3	3	3	3	1	0	0	0
#116	Sync button (?)	Edit metadata in menu	30		5	5	5	5	5	5	5	5	0	0
#117	Test editing of metadata	Edit metadata in menu	55		4	4	4	4	4	4	4	4	3	0
Backlog item 25	Userdefined metadata			16										
#122	GUI to add/remove metadata in object	Userdefined metadata	50		6	6	6	6	6	6	6	3	0	0
#123	GUI to add/remove metadata in blueprint	Userdefined metadata	50		6	6	6	6	6	6	6	6	0	0
#124	Test add/remove metadata on objects and blueprints	Userdefined metadata	50		4	4	4	4	4	4	4	3	3	0
Backlog item 21	Costestimation based on Metadata			10										
#118	Make standard for cost metadata	Costestimation	75		2	2	2	2	2	2	2	0	0	0
#119	Loop through all elements and summarise	Costestimation	70		2	2	2	2	2	2	2	2	0	0
#120	GUI-elements to show cost estimation	Costestimation	60		2	2	2	2	2	2	2	2	2	2
#121	Test cost estimation	Costestimation	60		4	4	4	4	4	4	4	2	0	0
Not in BL	Report writing			40										
	Write sprint 3 introduction	Report	80		2	2	2	2	2	2	2	2	2	2
	Design sprint3 backlog	Report	100		4	2	2	2	1	1	1	1	1	0
	Write Sprint 3 - Design chapter	Report	90		8	8	8	8	8	8	8	8	4	4
	Write Sprint 3 - Implementation chapter	Report	80		10	10	10	10	10	10	10	10	8	8
	Write Sprint 3 - Testing chapter	Report	80		8	8	8	8	8	8	8	8	1	1
	Write Sprint 3 - Results chapter	Report	70		4	4	4	4	4	4	4	4	4	4
	Write Sprint 3 - Evaluation chapter	Report	70		4	4	4	4	4	4	4	4	4	4



Estimate time:									
176	156,44	136,89	117,33	97,778	78,222	58,667	39,111	19,556	0

Figure E.3: Sprint 3 backlog and burndown chart

Ticket in trac	Summary	Component	Importance		Days in sprint / Effort left							
				Date:	02.11	03.11	04.11	05.11	06.11	09.11	10.11	12.11
				Effort left:	164	138	114	92	38	32	32	30
Backlog item 3	Import objects from SolidWorks via menu			10								
#127	Object creator (form?)	Import objects	100		10	10	10	10	0	0	0	0
#129	Mesh converter	Import objects	100		16	16	8	0	0	0	0	0
#130	Object importer	Import objects	100		8	8	8	8	0	0	0	0
#131	Test import object	Import objects	100		4	4	4	4	0	0	0	0
Backlog item 10	Connect different objects			44								
#132	Connection point creator	Connect objects	80		12	4	0	0	0	0	0	0
#133	Connection point metadata	Connect objects	80		4	4	4	0	0	0	0	0
#134	Snap to connection points	Connect objects	80		8	0	0	0	0	0	0	0
#135	Object tree/group structure	Connect objects	80		16	8	0	0	0	0	0	0
#136	Test connection objects	Connect objects	80		4	4	4	2	0	0	0	0
Backlog item 20	Basic physics applied to anchor lines			16								
#137	Generate spline between connection points	Basic physics	70		8	8	4	0	0	0	0	0
#138	Apply physics to splines	Basic physics	70		6	6	6	6	0	0	0	0
#139	Test spline physics	Basic physics	70		2	2	2	2	0	0	0	0
Backlog item 6	Map color plugin			26								
#140	GUI for color scale	Map color plugin	90		16	16	16	12	0	0	0	0
#141	Map color scale to depth on mesh	Map color plugin	90		8	8	8	8	4	0	0	0
#142	Test map color	Map color plugin	90		2	2	2	2	2	0	0	0
Not in BL	Report writing			40								
	Write sprint 4 - introduction	Report	80		2	2	2	2	0	0	0	0
	Design Sprint 4 backlog	Report	100		4	2	2	2	2	2	2	0
	Write Sprint 4 - Design chapter	Report	90		8	8	8	8	8	8	8	8
	Write Sprint 4 - Implementation chapter	Report	80		10	10	10	10	6	6	6	6
	Write Sprint 4 - Testing chapter	Report	80		8	8	8	8	8	8	8	8
	Write Sprint 4 - Results chapter	Report	70		4	4	4	4	4	4	4	4
	Write Sprint 4 - Evaluation chapter	Report	70		4	4	4	4	4	4	4	4



Estimate time:							
164	140,6	117,1	93,71	70,29	46,86	23,43	0

Figure E.4: Sprint 4 backlog and burndown chart

Appendix F

Textual use cases

This appendix contains the use cases produced during the requirement engineering process:

Use case name	**Start a new project**
Backlog item #	1, 11, 18
Actors	User, system
Flow of events	**1** The user launches the application **2** The application shows options for loading an earlier project or starting a new project. **3** The user chooses to start a new project. **4** The application shows available map areas. **5** The users chooses map area. **6** The application loads the map data into the 3D-window and the tools.
Extensions	**1.1** The user launches the application **3.1** The application shows options for loading an earlier project or starting a new project.
Entry condition	The application is running
Exit condition	The user closes the application, or the project is initiated with or without map data

Table F.1: Use case 1: Starting a new project

Use case name	**Place imported 3D object in scene**
Backlog item #	7
Actors	User, system
Flow of events	**1** The user chooses a 3D object from the 3D-object list. **2** When the user moves the cursor into the 3D-window the object follows it and snaps to the reasonable grid for the object to be placed in. **3** The user clicks the mouse button to place the object in the preferred place. **4** The object stays there, and will stop following the cursor.
Extensions	**3.1** The user gets rid of the object at the cursor by hitting esc
Entry condition	The application is running and a project with map data is open
Exit condition	The user hits esc or closes the application

Table F.2: Use case 2: Placing 3D object in scene

Use case name	**Select and modify placed object**
Backlog item #	8, 16
Actors	User, system
Flow of events	**1** The user clicks an object placed in the 3D-window. **2** The system highlights the selected object, and shows a toolbox and data for the object. **3** The user selects tool to use. **4** By dragging the mouse cursor while the mouse button is held down, the user manipulates the object.
Extensions	**3.1.1** The user selects rotation **3.1.2** The system displays helper circles around the object to allow for axis specific manipulation. **3.2.1** The user selects translation **3.2.2** The system displays helper axes around the object to allow for axis specific manipulation.
Entry condition	The application is running and a project with map data is open
Exit condition	The user hits esc or closes the application

Table F.3: Use case 3: Modifying an already placed 3D object

Use case name	**Navigate the 3D scene**
Backlog item #	2
Actors	User, system
Flow of events	**1** The user moves the mouse cursor into the 3D scene **2** The user holds down one of the ciew manipulation buttons **3** The user moves the cursor **4** The view of the scene changes
Extensions	**2.1** The user holds down ctrl **2.2** The user holds down alt **2.3** The user holds down shift **4.1** The view is zoomed in **4.2** The view is rotated **4.3** The view is translated sideways in relation to the current viewing direction
Entry condition	The application is running and a project with map data is open
Exit condition	N/A

Table F.4: Use case 4: Navigating the 3D scene

Use case name	**Connect two objects in the scene**
Backlog item #	10, 20
Actors	User, system
Flow of events	**1** The user moves chooses the type of connection **2** The user clicks (in the near vicinity of) a connection point on one object **3** The user clicks (in the near vicinity of) a connection point on another object **4** The system creates a link between the two objects and visualises the connection as for example an anchorline
Extensions	**N/A**
Entry condition	The application is running and a project with two connectable objects is open
Exit condition	The user hits esc or closes the application to cancel the operation

Table F.5: Use case 5: Connecting two objects in the 3D scene

Use case name	**Import saved object into project**
Backlog item #	3
Actors	User, system
Flow of events	**1** The user selects “Import object” from the menu **2** The user locates the file containing the object and selects it **3** The system reads the file and places the object in the available object list
Extensions	**2.1** The user exits the import dialogue box
Entry condition	The application is running and a project is open
Exit condition	The system cannot read the file or the user closes the application

Table F.6: Use case 6: Import 3D object into project

Use case name	**Saving a project**
Backlog item #	13
Actors	User, system
Flow of events	**1** The user selects the "Save project" option from the menu **2** The user selects the location to store the project in **3** The system saves the project state to disk in the selected location
Extensions	**N/A**
Entry condition	The application is running and a project with map data and objects is open
Exit condition	The user exits the save dialogue

Table F.7: Use case 7: Saving a project

Use case name	**Saving an object**
Backlog item #	12
Actors	User, system
Flow of events	**1** The user selects the the object which is to be saved **2** The user selects "Save object" option from the menu **2** The user selects the location to store the object in **3** The system saves the object state to disk in the selected location
Extensions	**N/A**
Entry condition	The application is running and a project with objects is open
Exit condition	The user exits the save dialogue

Table F.8: Use case 8: Saving an object

Appendix G

Timesheet

Group number		Date:													
Activity	**Week**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**47**	**48**	**Total:**
Supervisor meeting	Estim. this week / Estim. accumulated so far					4 / -	4 / -	4 / -	5 / -	5 / -	5 / -	5 / -	5 / -		
	Actual this week / Actual accumulated so far	8	5	5	4	4 / 26	4 / 30	4 / 34	5 / 39	4 / 43	4 / 47	4 / 51	3 / 54		54
Customer meeting	Estim. this week / Estim. accumulated so far					10 / -	0 / -	4 / -	0 / -	5 / 0	5 / -	0 / -	10 / -		
	Actual this week / Actual accumulated so far	5	12	8	10	10 / 45	0 / 45	4 / 49	0 / 49	6 / 53	8 / 61	0 / 61	15 / 76		76
Internal meetings	Estim. this week / Estim. accumulated so far					15 / -	12 / -	12 / -	12 / -	10 / -	10 / -	10 / -	10 / -		
	Actual this week / Actual accumulated so far	5	23	17	21	15 / 81	15 / 96	11 / 107	12 / 119	10 / 129	10 / 139	9 / 148	11 / 159		159
Seminars	Estim. this week / Estim. accumulated so far					10 / -	16 / -	16 / -	0 / -	0 / -	0 / -	0 / -	20 / -		
	Actual this week / Actual accumulated so far	20	10	12	12	4 / 58	16 / 74	0 / 74	0 / 74	0 / 74	0 / 74	0 / 74	20 / 94		94
Document writing	Estim. this week / Estim. accumulated so far					50 / -	50 / -	40 / -	40 / -	30 / -	30 / -	30 / -	30 / -		
	Actual this week / Actual accumulated so far		24	24	41	54 / 133	44 / 177	30 / 217	35 / 252	30 / 282	32 / 314	34 / 340	38 / 378		378
Planning	Estim. this week / Estim. accumulated so far					10 / -	10 / -	10 / -	5 / -	5 / -	5 / -	5 / -	5 / -		
	Actual this week / Actual accumulated so far		20	10	4	8 / 42	12 / 54	10 / 64	5 / 69	5 / 74	5 / 79	5 / 84	5 / 89		89
Preliminary studies	Estim. this week / Estim. accumulated so far					5 / -	4 / -	0 / -	0 / -	4 / -	0 / -	0 / -	0 / -		
	Actual this week / Actual accumulated so far		15	39	44	6 / 104	4 / 108	0 / 108	0 / 108	3 / 111	0 / 111	0 / 111	0 / 111		111
Sprint 1 design	Estim. this week / Estim. accumulated so far					30 / -	0 / -	0 / -	0 / -	0 / -	0 / -	0 / -	0 / -		
	Actual this week / Actual accumulated so far					15 / 15	15 / 30	0 / 30	0 / 30	0 / 30	0 / 30	0 / 30	0 / 30		30
Sprint 1 implementation	Estim. this week / Estim. accumulated so far					40 / -	40 / -	0 / -	0 / -	0 / -	0 / -	0 / -	0 / -		
	Actual this week / Actual accumulated so far					32 / 32	53 / 85	0 / 85	0 / 85	0 / 85	0 / 85	0 / 85	0 / 85		85
Sprint 1 testing	Estim. this week / Estim. accumulated so far					5 / -	15 / -	0 / -	0 / -	0 / -	0 / -	0 / -	0 / -		
	Actual this week / Actual accumulated so far					10 / 10	15 / 25	0 / 25	0 / 25	0 / 25	0 / 25	0 / 25	0 / 25		25
Sprint 2 design	Estim. this week / Estim. accumulated so far							20 / 20	20 / 40	0 / 40	0 / 40	0 / 40	0 / 40		
	Actual this week / Actual accumulated so far							16 / 16	20 / 36	0 / 36	0 / 36	0 / 36	0 / 36		36
Sprint 2 implementation	Estim. this week / Estim. accumulated so far							25 / 25	30 / 55	10 / 65	0 / 65	0 / 65	0 / 65		
	Actual this week / Actual accumulated so far							20 / 25	50 / 70	10 / 80	0 / 80	0 / 80	0 / 80		80
Sprint 2 testing	Estim. this week / Estim. accumulated so far							5 / 5	5 / 10	0 / 10	0 / 10	0 / 10	0 / 10		
	Actual this week / Actual accumulated so far							5 / 5	10 / 10	0 / 10	0 / 10	0 / 10	0 / 10		10
Sprint 3 design	Estim. this week / Estim. accumulated so far									20 / 20	10 / 30	0 / 30	0 / 30		
	Actual this week / Actual accumulated so far									25 / 25	8 / 33	0 / 33	0 / 33		33
Sprint 3 implementation	Estim. this week / Estim. accumulated so far									30 / 30	40 / 70	0 / 70	0 / 70		
	Actual this week / Actual accumulated so far									40 / 40	45 / 85	0 / 85	0 / 85		85
Sprint 3 testing	Estim. this week / Estim. accumulated so far									5 / 5	10 / 15	0 / 15	0 / 15		
	Actual this week / Actual accumulated so far									5 / 5	10 / 15	0 / 15	0 / 15		15
Sprint 4 design	Estim. this week / Estim. accumulated so far											20 / 20	10 / 30		
	Actual this week / Actual accumulated so far											16 / 16	12 / 28		28
Sprint 4 implementation	Estim. this week / Estim. accumulated so far											40 / 40	10 / 50		
	Actual this week / Actual accumulated so far											45 / 45	20 / 65		65
Sprint 4 testing	Estim. this week / Estim. accumulated so far											5 / 5	10 / 15		
	Actual this week / Actual accumulated so far											4 / 4	10 / 14		14
	Total hours estimated														0
	Total hours actual	38	109	115	136	154	163	100	137	138	122	117	134		1453