Sensur av hovedoppgaver

Høgskolen i Sørøst-Norge
Fakultet for teknologi og maritime fag

Prosjektnummer: **2016-11**
For studieåret: **2015/2016**
Emnekode: **SFHO3201**

Project Argos – Real-time Virtual Reality
Prosjekt Argos – Virtuell Virkelighet i Sanntid

Utført i samarbeid med: Kongsberg Defence Systems
Ekstern veileder: Alexander Gosling

Sammendrag: Project Argos er et system som lar deg operere et kjøretøy ved hjelp av kameraer, sensorer og VR-briller. Sjåføren får full oversikt over omgivelsene, noe som gir økt situasjonsforståelse og sikkerhet.

Stikkord:
- Virtual Reality
- Real-time
- Software

Tilgjengelig: DELVIS, kildekode er ikke offentlig tilgjengelig.

Prosjekt deltagere og karakter:

Navn	Karakter
Thomas Hansen	
Ingvild Damtjernhaug	
Trond Egil Hammer	
Leiv Fredrik Berge	
Morten J. Barbala	
Mathias Havdal	

Dato: 9. Juni 2016

_____	_____	_____
Radmila Juric	**Karoline Moholth**	**Alexander Gosling**
Intern Veileder	Intern Sensor	Ekstern Sensor

2016-11

ARGOS

Real-time Virtual Reality

23. May 2016

Ingvild Damtjernhaug
Leiv Fredrik Berge
Mathias Havdal
Morten J. Barbala
Thomas Hansen
Trond Egil Hammer

KONGSBERG

HSN
University College
of Southeast Norway

University College of Southeast Norway,
Faculty of Technology and Maritime Sciences

Project Argos

This is the documentation for the bachelor group 2016-11, Project Argos. We will in the following documents detail what Project Argos is, how we have worked with Project Argos and the technical aspects of Project Argos. On the next page you will find an overview of all the printed documents. Following that page the table shows all the documents produced in the project and if they are available in print, in the final PDF, in the assessment DVD or on the USB memory stick.

The source code of TinyArgos, the software in Project Argos, is not public. The source code and Doxygen documentation is only available on the assessment DVD.

The hyperlinks in the documents will only work from the file structure from the DVD or the USB drive. They will not work with documents downloaded from the documentation wiki. We recommend using the documentation wiki to navigate the documents by opening the index.html in the root directory.

A man who carries a cat by the tail learns something he can learn in no other way

Mark Twain.

Acknowledgements

We would like to extend our gratitude to the following people for their help and guidance.

- **Karoline Moholth** as internal sensor
- **Alexander Gosling** as external sensor and supervisor
- **Radmila Juric** as internal supervisor
- **Erik Torp** as project owner
- **Ellen Svarverud** as technical guide for visual perception
- **Fagskolen Tinius Olsen** for providing an electric car for the demo

Blue documents are project process documents.

Green documents are technical documentation.

Documents	Print	PDF	USB	CD
Project Plan	X	X	X	X
Risk Analysis	X	X	X	X
Requirements Documents	X	X	X	X
Test Specification	X	X	X	X
Test Logs	X	X	X	X
Architecture Notebook	X	X	X	X
TinyArgos Technical Solutions	X	X	X	X
Iteration Reports	X	X	X	X
Evaluation	X	X	X	X
Future	X	X	X	X
Network Solutions	X	X	X	X
GigE Vision SDK	X	X	X	X
VR Goggles	X	X	X	X
Graphics Library	X	X	X	X
Architecture Style	X	X	X	X
Lenses	X	X	X	X
Motion Sickness	X	X	X	X
User Guides	X	X	X	X
Glossary	X	X	X	X
Contracts	X	X	X	X
Sensurark	X	X	X	X
Meeting Reports		X	X	X
Weekly Followup		X	X	X
Build Guides		X	X	X
Presentations		X	X	X
Argos Poster		X	X	X
Argos Brochure	X	X	X	X
MS Project file, with Gantt chart			X	X
Doxygen Code Documentation				X

Project Plan 2.0

Created by: Leiv Fredrik Berge
21.1.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

HSN

KONGSBERG

Table of Contents

1. Document Overview

The purpose of the project plan is to show the organization and planning for the bachelor project of 2016 in Project Argos. The document describes the project background, the project purpose and the project owners, as well as giving an overview of the project goals and scope. It also describes the process used by the team, key milestones in the project and the project schedule, and the iterations of the project. Finally, the roles of the team members, our praxis, communication plan and risk management plan is detailed along with a budget for the project.

Describes

- the project background.
- why the project exists.
- what has been done in the project before.
- who is responsible for the project.
- the aim of the project.
- the goals of the project.
- the project scope.
- the project process.
- the project schedule with project phases.
- the time schedule with iterations and milestones.
- the organization of the project, with human resource plan and team member roles.
- the communication plan.
- the project budget.
- briefly the risk management plan (see also Risk Analysis document).

1.1 Document History

Version	Change	Date	Created by
0.1	First version	13.01.2016	Leiv Fredrik Berge
0.2	added project praxis and measurements	20.01. 2016	Leiv Fredrik Berge
0.3	Translated to English	21.01.2016	Leiv Fredrik Berge
0.4	Added phases	28.01.2016	Leiv Fredrik Berge
0.5	Added references, background, goals	02.02.2016	Leiv Fredrik Berge
0.6	Added references, made corrections, adjustments	04.02.2016	Ingvild Damtjernhaug
1.0	Fixed references	04.02.2016	Mathias Havdal
1.1	Changed colour on table headers	16.02.2016	Trond Egil Hammer
1.2	Added iteration plan for I4	08.03.2016	Leiv Fredrik Berge
1.3	Changed format on name and date in document history	10.04.2016	Trond Egil Hammer
1.4	Deleted iteration plan for I4 Added project praxis	20.04.2016	Ingvild Damtjernhaug

1.5	Change in estimated hours Added success criteria	21.04.2016	Ingvild Damtjernhaug
1.6	New overview, added challenges, evaluation	29.04.2016	Ingvild Damtjernhaug Leiv Fredrik Berge
1.7	Added organization chart, added group overview	04.05.2016	Leiv Fredrik Berge
1.8	Minor corrections	05.05.2016	Ingvild Damtjernhaug
1.9	Added supervisor and sensor roles	05.05.2016	Trond Egil Hammer
1.10	Reorganized the whole document, added human resources, project background, document overview and more	06.05.2016	Ingvild Damtjernhaug
1.11	Updated document overview, added project goals , team members, iteration diagram and process elements	09.05.2016	Ingvild Damtjernhaug
1.12	Added section 12 Budget	11.05.2016	Ingvild Damtjernhaug
1.13	Updated timeline, added milestones	12.05.2016	Leiv Fredrik Berge
1.14	Updated document overview, reorganized sections, deleted figure, added role information	15.05.2016	Ingvild Damtjernhaug
1.15	Updated references, added time schedule, added project life cycle	17.05.2016	Leiv Fredrik Berge
1.16	Fixed headings and layouts. Rewriting, corrections and clarifications	18.05.2016	Morten J. Barbala Ingvild Damtjernhaug
2.0	Final review	20.05.2016	Thomas Hansen Trond Egil Hammer Ingvild Damtjernhaug

1.2 Referenced Documents

Title	Document	Version
Glossary	doc-1113_glossary_2_0.docx	2.0
Argos Project	Argos_project_1_0.mpp	1.0
Risk analysis	doc-1113_risk_analysis_2_0.docx	2.0
Iteration Reports	doc-112_Iteration_report_2_0.docx	2.0

1.3 List of Figures

1.4 List of Tables

2. Project Background

Project Argos is a cross-disciplinary student project from Kongsberg Defence Systems (KDS). The project was initiated the summer of 2015 by summer interns at KDS. Through the summer projects KDS gets the opportunity to test technologies and develop systems that can be interesting for the future. We are the second group to work with Project Argos with our bachelor thesis in 2016, and succeeding project groups are expected to further develop and improve the system. Project Argos will serve as a technical demonstration and proof of concept for potential future products. The project can eventually go into the KDS' portfolio of command and control products that aims to increase the safety and situational awareness on the battlefield.

The armoured vehicles of today are only equipped with a small window or opening. In other words, the operator of the vehicle does not have great visibility. It is often required to open a latch and expose a crew member to a potentially hostile environment to acquire sufficient situational awareness. It could also be problematic in non-hostile environment, where the poor visibility from the armoured vehicle can cause or increase the severity of accidents. The reason for small windows in armoured vehicles is to reduce the risk of weaker armour and reduce cost of expensive armoured windows. The objective of Project Argos is to increase the situational awareness of the vehicle operator and to improve the safety of the crew through the use of virtual reality (VR). The driver of the vehicle should be able to use and drive the vehicle normally, without endangering the crew, the vehicle or the surroundings. The objective is also to record and store information to utilize in personnel training and recon.

To drive a vehicle with VR has not been possible due to hardware and software limitations. Graphics processing units (GPU) have not had the capability of processing the sheer number of pixels in the required time for a useable experience until the latest generation of GPUs. Camera technology has also had a rapid improvement in small, light sensitive and high quality sensors. Strides in CPU and GPU performance have reignited the VR development and multiple companies have plans to release VR headsets to the public in 2016. With faster GPUs and high quality cameras, there is now a possibility for real-time virtual reality.

At the end of the summer project in 2015 the previous team had produced a functional software prototype using USB web cameras. It showed live video and static heads-up display (HUD) objects. However, the live video was displayed with 200-250ms latency, which is more than double of the requirement, and there were no playback or recording functionalities. The basis of the virtual world, along with some configuration capabilities, was in place when our bachelor team took over.

2.1 Project Description

The project will develop a solution that enables the user to "see through amour" with real-time virtual reality. From the inside of a vehicle the user can see the surroundings through cameras and sensors mounted outside the vehicle.

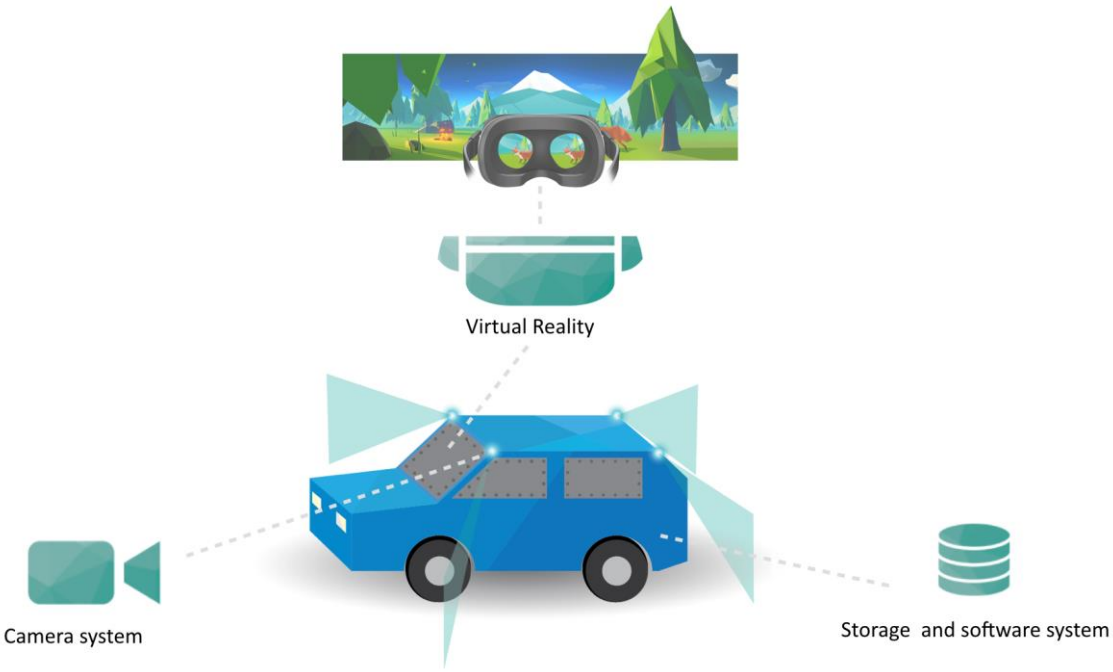

Figure 1: High level system architecture of Project Argos with software TinyArgos

By using a camera system with multiple high resolution cameras, Project Argos will have a continuous panoramic view of the surroundings outside the vehicle. The cameras will provide the storage and VR system with picture data through the key part of Project Argos, the software called TinyArgos. The software is the heart of the system, where everything is connected and information is distributed. The bachelor team of 2016 will develop and improve the existing software. See section 3. Project Scope for more information.

Instead of seeing through the windows, the user will use VR goggles to operate the vehicle. A projection of the environment is recreated in real-time in a virtual world. Low delay from camera to screen in VR goggles will make it possible to drive using only the video solution. This removes the drivers need for computer monitors. Considering the limited space inside the vehicle, it will be beneficial to use goggles instead of screens. This also enables separate views for the driver and the crew.

The aim is to give the driver better situational awareness, overview and understanding of the situation. Project Argos could improve the safety for an operator of an armoured vehicle and be integrated with other information systems to provide the crew with additional information. Project Argos consists of the cameras and sensor subsystems, the network, computer hardware, the VR headset and the TinyArgos software.

2.2 Project Owner Description

The project assignment is given by Kongsberg Defence Systems (KDS). KDS is a leading Norwegian space and defence contractor. The product portfolio includes command and control, weapon guidance and surveillance, communication solutions

and missiles. KDS also makes advanced composites and engineering productions for the aircraft and helicopter market. [1]

3. Project Scope

For this bachelor project we will continue to develop TinyArgos, the software in Project Argos. This will result in TinyArgos 2.0. The main focus is to implement the new GigE Vision cameras with the software. The live video from the camera rig will be merged into a continuous image and displayed in the Oculus Rift headset with low latency. We will also create functionalities to record the video, store it on disk and make it possible to play back the recordings with the Project Argos software. Lastly, the Project Argos software will support markers and information displayed on the HUD.

The software we inherited was a proof of concept displaying still images from low-end web cameras. Our scope is to expand the system to integrate new high-end fixed focus surveillance cameras with live video, and deliver a functioning real-time VR experience. The project will continue as a summer project for KDS, so an important focus of the project is to have a clean and clear documentation and code base.

We will not integrate the system on a vehicle or create the physical mounting points for the physical system components. We will not integrate Project Argos with other sensors or systems. Project Argos will not be ready for production at the end of this bachelor project, but rather a demonstration of the technology and the possibilities of real-time virtual reality for further development.

3.1 TinyArgos 2.0 Project Goals

The main aspects of development and main goals of our scope:

Project Argos, TinyArgos 2.0:

- Capture live video from four GigE Vision cameras
- Merge video streams to a single continuous image
- Store video stream to disk
- Playback of stored video
- Add markers and extra information on HUD

At delivery of the bachelor thesis we are expected to have a functioning technology demonstration of Project Argos. The tech demo will serve as a proof of concept and a technology showcase for KDS. It must be easy for a user to start and stop a recording of the live video, and to play the recordings back with the same experience as live video. The TinyArgos software must provide the user with a smooth and pleasant live VR experience. The experience should be so pleasant as to prove the viability of real world applications of real-time virtual reality. The specific engineering goals are described in the requirements document.

4. OpenUP: Project Process

This project follows OpenUP as project process. OpenUP is based on the principles of Unified Process of iterative and incremental approaches, but is designed specifically for small co-located software development teams. OpenUP is stripped down to the essentials, which allows us to extend the process with the activities we need and be effective with the system development. The focus of OpenUP is to create value for the stakeholder, not to get stuck with unproductive deliverables and formalities. [2]

OpenUP relies on an iterative approach within a project life cycle and has four principles, which are similar to the principles in the Agile Manifesto. [3]

- Collaborate to align interests and share understanding
- Balance competing priorities to maximize stakeholder value
- Focus on the architecture early to minimize risks and organize development
- Evolve to continuously obtain feedback and improve
  [2]

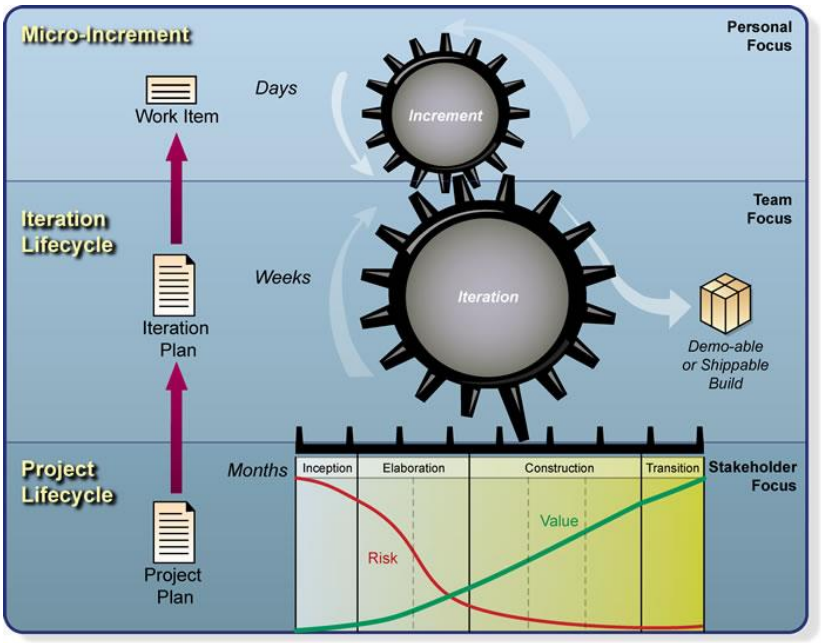

Figure 2: OpenUP layers
[4]

OpenUP has three layers of focus. The personal focus is the micro-increments that each person does every day by performing tasks and producing work items. The tasks are usually a day or shorter. The micro-increments drive the iterations forward, and at the end of the iteration new functionality is integrated in the system. This is seen on the team level. The last focus area is the stakeholders. The iterations drive the project forward and delivering on the stakeholder requirements.

OpenUP is meant to be minimal, complete and extendable [5]. That means it contains all necessary processes for a group to perform an entire project, but also allows the group to tailor and adapt the process to each specific project. It is minimal in the way that it is not overloaded with unproductive and formal tasks and

documents, and contains a minimum of roles, tasks, artefacts and guidance. This is to be lightweight, agile and flexible as a project process. In addition to the unified process influence, it also incorporates aspects from agile project models. OpenUP tries to be less pedantic than unified process and weighs documentation less. OpenUP also incorporates agile principles and techniques. OpenUP can be seen as a middle ground between Rational Unified Process [6] and agile models like Scrum [7] and XP [8].

[4] [9] [10]

4.1 OpenUP: Basic Elements

The basic elements that are the building blocks for the entire project model is:

- **Work product**: The item being produced
  Work product covers everything produced by the project, including documents, code, decisions or physical products. If the work product results in something concrete it is called an artefact. If the outcome is not something concrete, it is called a result. Multiple artefacts put together are called deliverance.

- **Task**: How to perform the work
  A task is performed by a role. The task comprises of a series of steps that includes creating or altering work products. The tasks are the main building blocks that the project members perform to produce results in the project.

- **Role**: Who will perform the tasks and who is responsible
  The OpenUP process requires there to be a stakeholder, an analyst, an architect, a developer, a tester and a project manager.
  See section 9.2 for more information.

- **Process**: Defines the work flow and breakdown of the work
  Processes collect and structure the task, work products and roles. Processes are built up by activities that enable us to plan the progress in the project by creating work breakdown structures (WBS). The processes are the starting point to produce the Gantt diagram.

- **Guidance**: Templates, lists, examples and concepts

[2]

4.2 Reasoning for choosing OpenUP

We did not have any first-hand experience with following a defined project process, so we had to do research and rely on advice from supervisors and the college. We found early on that we needed an iterative and flexible model to support our very software heavy project. We considered Rational Unified Process (RUP), Scrum and OpenUP. With our research our conclusion was that OpenUP seemed to be a good compromise between unified process and Scrum. We wanted a process that natively allows for deadlines, as the deadlines the college sets are rigid. We could have modified Scrum to fit our need, or we could have used RUP modified with agile

methods. The key deciding factor for us was that we wanted a well-documented process that would enable us to quickly get to working. The Eclipse Foundation curates the wiki for OpenUP, a site with lots of information, guidance and tools to help us get up and running.

5. OpenUP: Phases of the Project

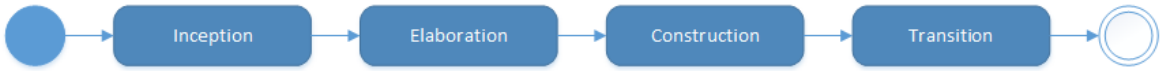

Figure 3: Project phases

OpenUP projects are divided into four phases, from inception to transition. The phases focus on different parts of the development process, and can contain multiple iterations inside the phase. This division is intended to promote collaboration and align interests and understanding. Also it's intended to get a stable architecture in place early in the project. The work from earlier phases is reviewable in later phases as understanding and knowledge grows. The focus on requirements early on is to achieve a stable architecture as early as possible. The architecture should be able to incorporate expansions of the system. However, if needed it is also possible to alter the architecture in later stages. Dividing the project into phases makes it clear what the objective at any time in the project is.

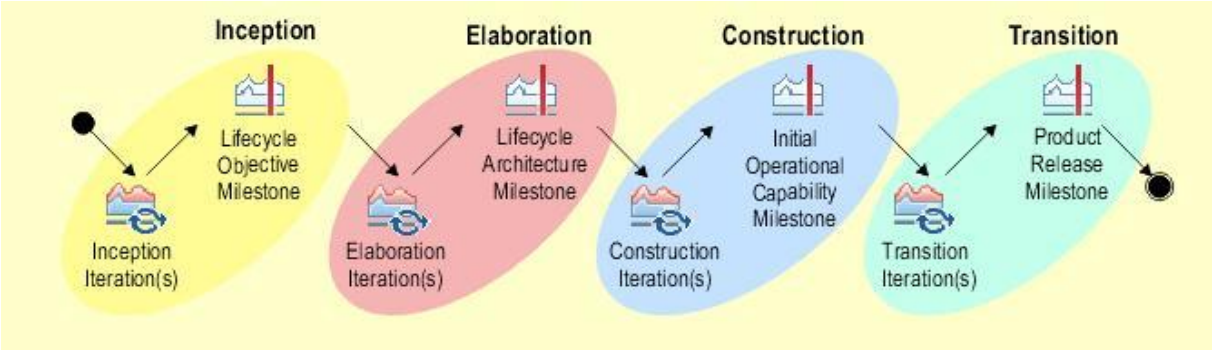

Figure 4: The OpenUP life-cycle

[4]

Phase	Objective
1 Inception	• Understand what to build • Identify key system functionality • Determine a possible solution • Understand cost, schedule and risk in the project
2 Elaboration	• Get a more detailed understanding of the requirements • Design, implement, validate and baseline an architecture

	• Mitigate risks • Produce an accurate schedule
3 Construction	• Iteratively develop a complete product • Minimize development costs
4 Transition	• Test to validate that user experience are met • Achieve stakeholder concurrence that deployment is complete • Improve future project performance

Table 1: Overview of the objectives in the phases in OpenUP

[2]

5.1 Inception

The focus in the inception phase is to assemble the project group, make some key decisions on the choice of project model, distribution of roles, align interests and understanding and prepare the environment.
[4]

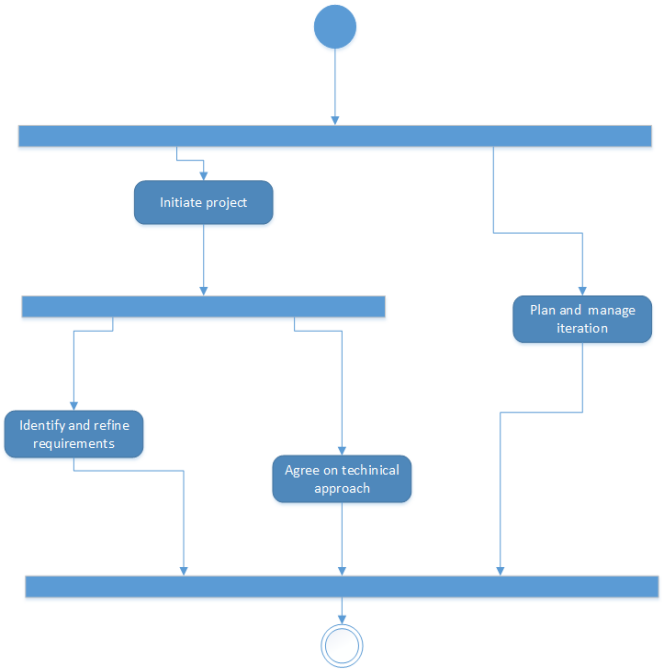

Figure 5: Activity diagram inception phase

5.2 Elaboration

In the elaboration phase tasks can be performed in parallel. The main objectives are to discover the functional and system requirements, develop the architecture for the project and create a project plan, which includes risk mitigation and work schedule.
[4]

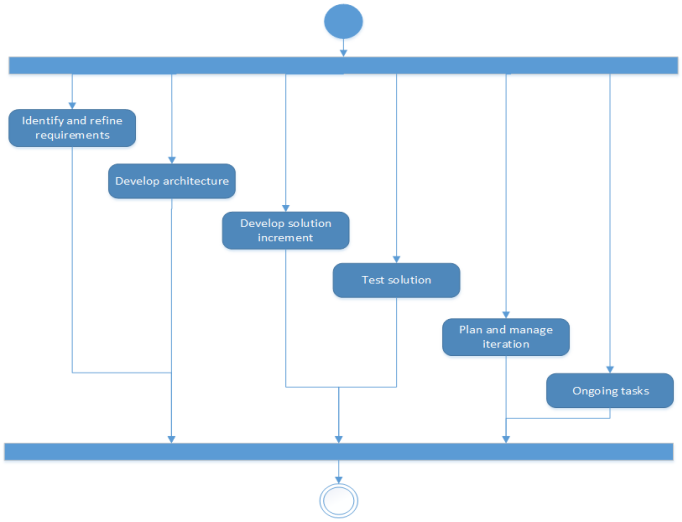

Figure 6: Activity diagram elaboration phase

5.3 Construction

In the construction phase the development of the project is iteratively performed inside the bounds of the architecture. Ideally the architecture is stable and robust enough to allow all requirements to be implemented within the structure. Functionalities will continuously be tested, validated and integrated into the code base. This ensures that the code base is complete and stable. Code and system can be tested and shown to the intended users in this phase.
[4]

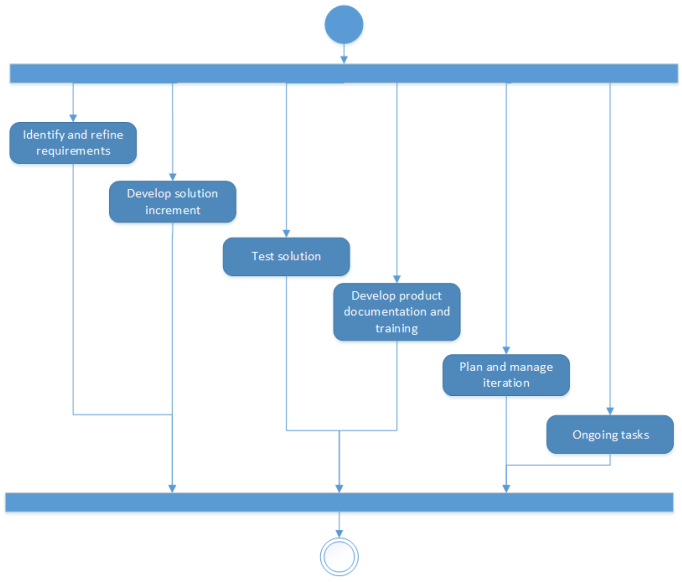

Figure 7: Activity diagram construction phase

5.4 Transition

In the final transition phase the main objective is to polish the finished product; fine tune functionalities, performance and overall quality from testing to the release. This is also the phase where documentation is finalized and users are given the appropriate instructions and training to operate the system.
[4]

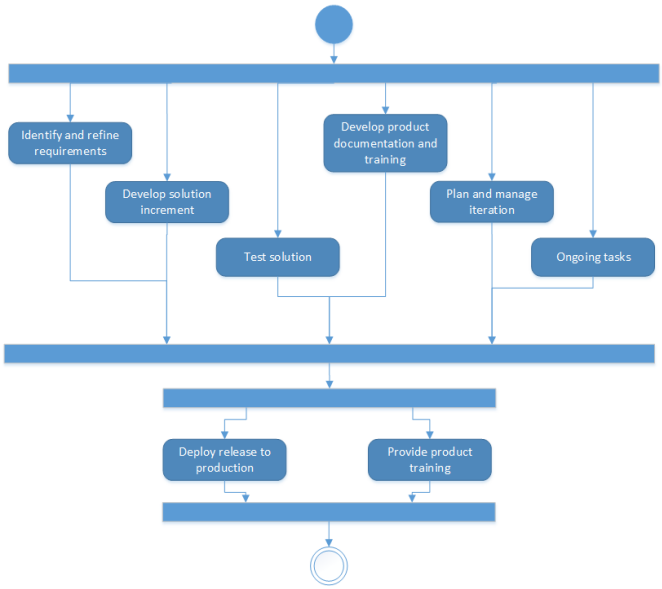

Figure 8: Activity diagram transition phase

6. OpenUP: Iterations

An iteration is a set period of time to accomplish a number of tasks to improve the product. At the end of each iteration, the code base should be stable and executable with the new functionalities to ensure that there is always added value to the project. The artefacts created in earlier iterations are also updated. This means older documents are updated as project knowledge and skills increase. Instead of developing artefacts one after another in a pipeline fashion, they are evolving throughout the project life cycle, although at different rates. [4]

The iterative approach means we implement functionality into the code base often, instead of a large single implementation at the end of the project. This approach can help us discover problems and issues earlier in the process, minimizing the cost to repair bugs, faults and errors. The iterative approach can also help divide the major development tasks into smaller more manageable chunks of work. [11]

This approach is different from a more traditional waterfall-based model, where each task follows the previous and is performed until it is finished. In pure software development the waterfall approach could be problematic as the implementation stage is pushed back towards the end of the project. You run the risk of not having a functioning code base until towards the very end, just individual software modules. It could be expensive, hard and time consuming to discover a problem at a late stage in development. The iterative approach tries to combat this issue. [12]

7. Milestones in Project Argos

The key milestones in the project are the presentations for the university college, the system acceptance test with the stakeholder, delivery of the final documentation and the deliverance presentation for KDS.

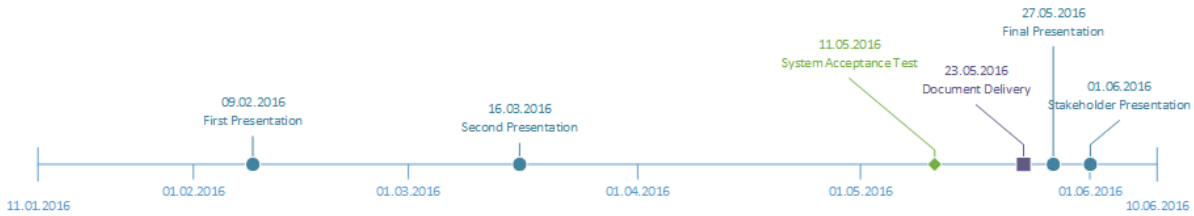

Figure 9: Project milestones

- First presentation 09.02.2016
  - Project plan
  - Requirements specification
  - Test specification
- Second presentation 16.03.2016
  - Project progress
  - System architecture
  - Software architecture
- System Acceptance Test with stakeholder 11.05.2016
  - Live demo in the lab with project owner

- Final Document Delivery 23.05.2016
  - All documentation
  - All code documentation
- Final Presentation 27.06.206
  - Project progress
  - Final product demo
  - Technical demonstration
- Stakeholder Presentation and deliverance 01.06.2016
  - Project handover
  - Technical demonstration

8. Project Schedule

The project started 11.01.2016 with final documentation delivery 23.05.2016. The final presentation is 27.05.2016 and final delivery to the stakeholder is 01.06.2016. We expect to spend 500 to 600 hours per group member, totalling around 3500 hours of work with all aspects of the project, from planning to execution and presenting. This is based on projections and expectations from the college. The bulk of the hours will be spent developing the solution and expanding the functionality and code base. We have used the software Microsoft Project to create the detailed project schedule. See for the detailed Gantt diagram.

8.1 Time Schedule

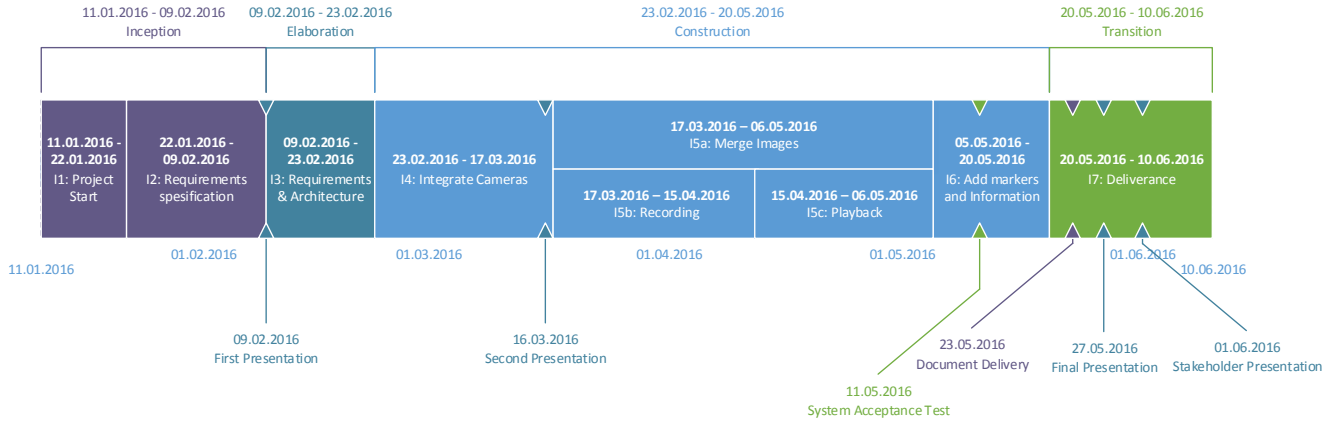

Figure 10: Project timeline

Based on OpenUP we have four phases in the project with seven iterations. In the first phase with two iterations the focus is to understand the complexities of the tasks at hand and to organize the work in the best way possible. It will also be focused on gathering of requirements and to create tests to verify and validate all the requirements. This phase will end with the first presentation as the milestone.

The elaboration phase contains one iteration and will focus on the architecture and refinement of the requirements based on feedback from the stakeholders. The

architecture at the end of the elaboration phase should be stable enough to handle most, if not all the challenges in the following construction phase, and the architecture notebook is the key artefact from this phase.

The construction phase includes four iterations, with three iterations working in parallel. This is the largest phase of the project. It has two major milestones: The second presentation and the external systems acceptance test. The focus in the construction phase is extending the code base with new and improved functionalities. The architecture will also be updated along the way to be current with the code base. Tests will be performed along the way, verifying functionalities before they are merged in to the default software branch. This will ensure that our default branch of the code base always works and new functionalities are constantly integrated. The key artefacts are the iteration reports and burndown reports to keep track of the progress.

The final transition phase contains the delivery of the project. The major milestones are documentation delivery, the final presentation and the deliverance with the stakeholder presentation. In this phase the development work shall be finished, and the focus is to perform the final polish to the documentation and deliverables.

8.2 Milestones and Objectives in Iterations

We have planned for seven iterations, with three running in parallel. Each iteration has clearly defined tasks, milestones and success criteria. At the end of the iterations we will review the work done and assess any tasks needing more work. The iterations will last roughly two weeks, but we have done some accommodations in iteration length to make sure the team is utilized to the best of our abilities. Some hours will be lost to the Easter holiday, and the tasks in I5a Merge Images does not make sense to break into smaller pieces to fit it into a two-week iteration. The iterations I5a, I5b and I5c will run in parallel to utilize the entire team. We are able to achieve this parallelism since the recording and playback functionality does not depend on the functionality for merge images, and the other way around. We split the work into different iterations to clarify the objectives of the major development goals the project had. This is to avoid convoluting an iteration with objectives from different major development tasks.

The iterations contain the objectives and milestones of the project. All our work is geared towards developing the system within the timeframe and reaching the milestones. The objectives in every iteration are directly linked to the task that is performed in each iteration. Some tasks can also be performed multiple times in a single iteration.

Iteration	Objective	Start date and milestones
1.1 - I1	**Inception, project start** Plan project Mitigate risks Develop vision Create environment	11.1.2016 – 19.1.2016
1.2 - I2	**Inception, requirements** Discover requirements Develop use case Develop test specification Envision architecture	20.1.2016 – 9.2.2016 Presentation 1: 9.2.2016
2.1 - I3	**Elaboration, requirements and architecture** Refine architecture Refine system-wide requirements Refine use case Mitigate risk	9.2.2016 – 21.2.2016
3.3 - I4	**Construction, integrate cameras** Develop, test and integrate functionality Revise requirements Mitigate risks	22.2.2016 – 17.3.2016 Presentation 2: 16.3.2016
3.4 - I5a	**Construction, merge images** Develop, test and integrate functionality Revise requirements Mitigate risks	18.3.2016 – 4.5.2016
3.5 - I5b	**Construction, record video feed** Develop, test and integrate functionality Revise requirements Mitigate risks	18.3.2016 – 14.4.2016
3.6 - I5c	**Construction, video playback** Develop, test and integrate functionality Revise requirements Mitigate risks	15.4.2016 – 4.5.2016
3.7 - I6	**Construction, add markers and overlays** Develop, test and integrate functionality Revise requirements Mitigate risks	5.5.2016 – 20.5.2016
4 - I7	**Transition** Deliver the documentation Deliver the project to project owner	20.5.2016 – 10.6.2016 Documentation: 23.5.2016 Presentation 3: 27.5.2016

8.3 Work Breakdown Structure

To make sure we reach the objectives of the project we have broken the large development job into small tasks. The tasks are the micro increments that together build the project. There are five top-level tasks in the work breakdown structure (WBS). These correspond with the phases of the project and ongoing tasks. The top-level and mid-level tasks are comprised of low-level tasks. It is the low level tasks that are the work items that the group members will perform. The last column in the WBS table is the WBS number. This is the unique identifying number for that task. Documents produced by a task carry the number of that task. The following table contains the top level and the second level tasks, showing the phases and iterations.

Name	Start	Finish	WBS
Inception	Mon 11.01.16	Tue 09.02.16	1
Project start - I1	Mon 11.01.16	Fri 19.01.16	1.1
Requirement specification - I2	Wed 20.01.16	Tue 09.02.16	1.2
Risk review meeting - inception	Tue 19.01.16	Tue 02.02.16	1.3
Elaboration phase	Tue 09.02.16	Tue 23.02.16	2
Requirements & architecture - I3	Tue 09.02.16	Tue 21.02.16	2.1
Risk review meeting - elaboration	Tue 16.02.16	Tue 16.02.16	2.3
Construction phase	Mon 22.02.16	Fri 20.05.16	3
Plan and manage iteration	Tue 23.02.16	Wed 24.02.16	3.1
Identify and refine requirements	Tue 23.02.16	Tue 23.02.16	3.2
Integrate cameras - I4	Mon 22.02.16	Thu 17.03.16	3.3
Presentation	Wed 09.03.16	Thu 17.03.16	3.9
Merge image - I5a	Thu 17.03.16	Fri 04.05.16	3.4
Record video stream - I5b	Thu 17.03.16	Fri 14.04.16	3.5
Video playback - I5c	Fri 15.04.16	Wed 04.05.16	3.6
Add markers and information - I6	Thu 05.05.16	Fri 20.05.16	3.7
Risk review meeting - construction	Tue 15.03.16	Tue 26.04.16	3.8
Transition phase	Fri 20.05.16	Thu 02.06.16	4
Evaluate project	Wed 25.05.16	Thu 26.05.16	4.3
Documentation deadline	Wed 25.05.16	Wed 25.05.16	4.1
Perform systems test	Fri 20.05.16	Mon 23.05.16	4.4
Review systems test	Fri 20.05.16	Fri 20.05.16	4.5
Presentation	Wed 01.06.16	Thu 02.06.16	4.2
Ongoing tasks	Mon 11.01.16	Wed 01.06.16	5

Supervisor meeting	Mon 11.01.16	Mon 11.01.16	5.1
Stakeholder meeting	Mon 11.01.16	Mon 11.01.16	5.2
General research	Mon 11.01.16	Mon 11.01.16	5.3
Administrative work	Mon 11.01.16	Thu 14.01.16	5.4
Web page	Mon 11.01.16	Fri 15.01.16	5.5

Table 2: Top level WBS

The following table shows the entire WBS. Indentations are used to clarify where each task belongs in the hierarchy.

Name	WBS
Inception	1
Project start - I1	1.1
Initiate project	1.1.1
Initial project meeting	1.1.1.1
Develop document standard	1.1.1.2
Plan project	1.1.1.3
Develop technical vision	1.1.1.4
Plan and manage iteration	1.1.2
Plan iteration	1.1.2.1
Prepare environment	1.1.2.2
Tailor the process	1.1.2.2.1
Set up tools	1.1.2.2.2
Verify tool configuration and installation	1.1.2.2.3
Deploy process	1.1.2.2.4
Manage iteration	1.1.2.3
Assess results	1.1.2.4
Requirement specification - I2	1.2
Identify and refine requirements	1.2.1
Identify and outline requirements	1.2.1.1
Detail use-case scenarios	1.2.1.2
Detail system-wide requirements	1.2.1.3
Create test cases	1.2.1.4
Agree on technical approach	1.2.2
Envision the architecture	1.2.2.1
Presentation	1.2.4
Create presentation	1.2.4.1
Presentation 1 rehearsal	1.2.4.2
Presentation 1	1.2.4.3
Risk review meeting - inception	1.3
Risk review meeting - inception 1	1.3.3
Risk review meeting - inception 2	1.3.4
Elaboration phase	2
Requirements & architecture - I3	2.1
Plan and manage iteration	2.1.1
Identify and refine requirements	2.1.2

Technical research: VR-Goggles	2.1.2.1
Technical research: Lenses	2.1.2.2
Technical research: Network	2.1.2.3
Develop architecture	2.1.3
Refine architecture	2.1.3.1
Develop solution increment	2.1.3.2
Design solution	2.1.3.2.1
Implement solution	2.1.3.2.3
Integrate and create build	2.1.3.2.5
Risk review meeting - elaboration	2.3
Risk review meeting - elaboration 1	2.3.3
Construction phase	3
Plan and manage iteration	3.1
Identify and refine requirements	3.2
Integrate cameras - I4	3.3
Plan and manage iteration	3.3.1
Identify and refine requirements	3.3.2
Develop solution increment	3.3.3
Design solution	3.3.3.1
Implement solution	3.3.3.3
Integrate and create build	3.3.3.5
Merge image - I5a	3.4
Plan and manage iteration	3.4.1
Identify and refine requirements	3.4.2
Develop solution increment	3.4.3
Design solution	3.4.3.1
Implement solution	3.4.3.3
Integrate and create build	3.4.3.5
Record video stream - I5b	3.5
Plan and manage iteration	3.5.1
Identify and refine requirements	3.5.2
Develop solution increment	3.5.3
Design solution	3.5.3.1
Implement solution	3.5.3.3
Integrate and create build	3.5.3.5
Video playback - I5c	3.6
Plan and manage iteration	3.6.1
Identify and refine requirements	3.6.2
Develop solution increment	3.6.3
Design solution	3.6.3.1
Implement solution	3.6.3.3
Integrate and create build	3.6.3.5
Add markers and information - I6	3.7
Plan and manage iteration	3.7.1
Identify and refine requirements	3.7.2
Develop solution increment	3.7.3
Design solution	3.7.3.1

Implement solution	3.7.3.3
Integrate and create build	3.7.3.5
Risk review meeting - construction	3.8
Risk review meeting - construction 1	3.8.1
Risk review meeting - construction 2	3.8.2
Risk review meeting - construction 3	3.8.3
Presentation	3.9
Create presentation 2	3.9.7
Presentation 2 rehearsal	3.9.8
Presentation 2	3.9.9
Transition phase	4
Documentation deadline	4.1
Presentation	4.2
Create final presentation	4.2.1
Final presentation rehearsal	4.2.2
Final presentation	4.2.3
Evaluate project	4.3
Perform systems test	4.4
Review systems test	4.5
Ongoing tasks	5
Supervisor meeting	5.1
Stakeholder meeting	5.2
General research	5.3
Administrative work	5.4
Web page	5.5

Table 3: All low level task WBS

8.4 Iteration Plans

To make it easier to handle the amount of work that the project needs to get done, the iteration is a great tool to focus on the tasks at hand. The iterations allow us to incrementally improve the project with new functionality or artefacts and updates to older work items. The task has a priority of 0-5, where 0 is the highest priority and 5 the lowest. These tables contain the specific technical development tasks in each iteration. There is also implementation and testing work that is covered under the development tasks. The planning of the iteration and other administrative work are not included in the following tables, they are meant to give a clear picture of the development tasks in the iterations. More details about the iterations can be found in the iteration report document.

8.4.1 Iteration I1 Project Start

The priority in the first iteration is to get the project started and develop a common ground to start the development with. Everybody needs to be familiarized with the project vision and goals, and we need to create a suitable office space to set up our tools.

Task	Priority	Responsible	Estimated hours	WBS
Develop document standard	1	Ingvild Damtjernhaug	12	1.1.1.2
Initial project plan	0	Leiv Fredrik Berge	100	1.1.1.3
Develop technical vision	0	Trond Egil Hammer	48	1.1.1.4
Tailor the process	1	Leiv Fredrik Berge	30	1.1.2.2.1
Set up tools	2	Mathias Havdal	75	1.1.2.2.2
Verify tools and configurations	2	Morten J. Barbala	30	1.1.2.2.3
Deploy process	1	Leiv Fredrik Berge	20	1.1.2.2.4

Table 4: Iteration I1 WBS plan

8.4.2 Iteration I2 Requirements Spesification

The second iteration starts the work with gathering and structuring the requirements and use cases. This ends with the first presentation where the requirements specification and test specification is presented.

Task	Priority	Responsible	Estimated hours	WBS
Identify and outline requirements	0	Trond Egil Hammer	75	1.2.1.1
Detail use case scenarios	0	Thomas Hansen	75	1.2.1.2
Detail system-wide requirements	0	Thomas Hansen	100	1.2.1.3
Create test cases	1	Morten J. Barbala	40	1.2.1.4
Set up tools	2	Mathias Havdal	40	1.1.2.2.2
Verify tools and configurations	2	Morten J. Barbala	30	1.1.2.2.3
Risk Review meeting	1	Leiv Fredrik Berge	12	1.3
Create first presentation	0	Ingvild Damtjernhaug	100	1.2.4.1
First presentation	0	Leiv Fredrik Berge	12	1.2.4.3

Table 5: Iteration I2 WBS plan

8.4.3 Iteration I3 Requirements & Architecture

In the third iteration we focus on refining the requirements and developing a stable architecture to start the development process. We will get the inherited code to compile in our build environment and to technical research.

Task	Priority	Responsible	Estimated hours	WBS
Refine architecture	0	Thomas Hansen	200	2.1.3.1
Develop solution process diagram	1	Mathias Havdal	40	2.1.3.2
Risk review meeting	1	Leiv Fredrik Berge	12	2.3
Refine risk document	0	Trond Egil Hammer	40	2.3.3
Update SDKs	1	Mathias Havdal	20	2.1.3.2.3
Migrate VS 2013 to 2015	1	Mathias Havdal	20	2.1.3.2.3
Compile code base	1	Mathias Havdal	10	2.1.3.2.3
Research network solutions	1	Morten J. Barbala	40	2.1.2.3
Research lenses	0	Trond Egil Hammer	70	2.1.2.2
Technical document VR goggles	1	Ingvild Damtjernhaug	40	2.1.2.1

Table 6: Iteration I3 WBS plan

8.4.4 Iteration I4 Integrate Cameras

The forth iteration is the first in the construction phase. We will focus on developing the functionality we need to implement the new type of cameras with the GigE Vision SDK. This includes the second presentation as well, this presentation has a more technical focus, detailing the architecture.

Task	Priority	Responsible	Estimated hours	WBS
Refine architecture	0	Thomas Hansen	100	3.3.2
Choose GigE SDK	0	Mathias Havdal	20	3.3.3.1
Implement GigE SDK	0	Mathias Havdal	50	3.3.3.3
Order lenses	0	Trond Egil Hammer	30	3.3.1
Implement XML config parsing	1	Leiv Fredrik Berge	50	3.3.3.3
Create functions to receive picture data	0	Mathias Havdal	130	3.3.3.3

Create second presentation	1	Ingvild Damtjernhaug	100	3.9.7
Second presentation	0	Leiv Fredrik Berge	12	3.9.9

Table 7: Iteration I4 WBS plan

8.4.5 Iteration I5a Merge Images

The fifth iteration is in parallel with I5b first, and later with I5c. This is a longer iteration because there is only a few large tasks, that does not make sense to divide into smaller chuncks. This is very important development tasks, where the video streams will be displayed in the Oculus and stitched.

Task	Priority	Responsible	Estimated hours	WBS
Refine architecture	0	Leiv Fredrik Berge	50	3.4.2
Transfer video data to OpenGL render medium	0	Mathias Havdal	100	3.4.3.3
Create and position OpenGL geometry to render video on	0	Thomas Hansen	100	3.4.3.3
Solutions for merge areas	1	Leiv Fredrik Berge	50	3.4.3.1
Review config system	0	Ingvild Damtjernhaug	100	3.4.3.3

Table 8: Iteration I5a WBS plan

8.4.6 Iteration I5b Record Video

I5b is the first of the two part development of the playback features. This first part creates the recording function. This will be quite hard to verify with test before the I5c is complete with the playback functions.

Task	Priority	Responsible	Estimated hours	WBS
Add interface to control recording	0	Trond Egil Hammer	40	3.5.3.3
Add HUD notification for recording status	0	Morten J. Barbala	75	3.5.3.3
Implement system for recording to file	0	Trond Egil Hammer	150	3.5.3.3

Table 9: Iteration I5b WBS plan

8.4.7 Iteration I5c Video Playback

The second part of the playback functionality is the player. This will get the data from a recording, and reuse as much as possible from live video to create the playback.

Task	Priority	Responsible	Estimated hours	WBS
Add functionality fir creating buffer from raw data on disk	0	Morten J. Barbala	40	3.6.3.3
Add functionality for displaying buffers in renderer with correct timing and framerate	0	Mathias Havdal	80	3.6.3.3
Add functionality for controlling playback	1	Morten J. Barbala	60	3.6.3.3
Add functionality for parsing config file specific to the recording that is being played	1	Leiv Fredrik Berge	60	3.6.3.3

Table 10: Iteration I5c WBS plan

8.4.8 Iteration I6 Add Markers and Information

In the final construction iteration we will perform all final system tests and finalize code documentation. We will also update the HUD object system.

Task	Priority	Responsible	Estimated hours	WBS
Clean up HUD objects	0	Thomas Hansen	60	3.7.3.3
Generate test HUD objects	1	Ingvild Damtjernhaug	20	3.7.3.3
Perform system tests	0	Morten J. Barbala	60	3.7.3.5
Create systems guide	1	Mathias Havdal	30	3.7.3.1
Finalize code documentation	0	Leiv Fredrik Berge	75	3.7.2

Table 11: Iteration I6 WBS plan

8.4.9 Iteration I7 Delivery

The final iteration is all about delivery, with the final presentation, documentation deadline and the delivery of the project to the project owner.

Task	Priority	Responsible	Estimated hours	WBS
Finalize documentation	0	Ingvild Damtjernhaug	50	4.1
Finalize final presentation	0	Leiv Fredrik Berge	50	4.2.1
Perform final presentation	0	Leiv Fredrik Berge	12	4.2.3
Deliver the project	0	Leiv Fredrik Berge	30	4.2

Table 12: Iteration I5a WBS plan

9. Human Resource Plan

This section identifies the individuals and organizations with leading roles in Project Argos. Their roles and responsibilities are described.

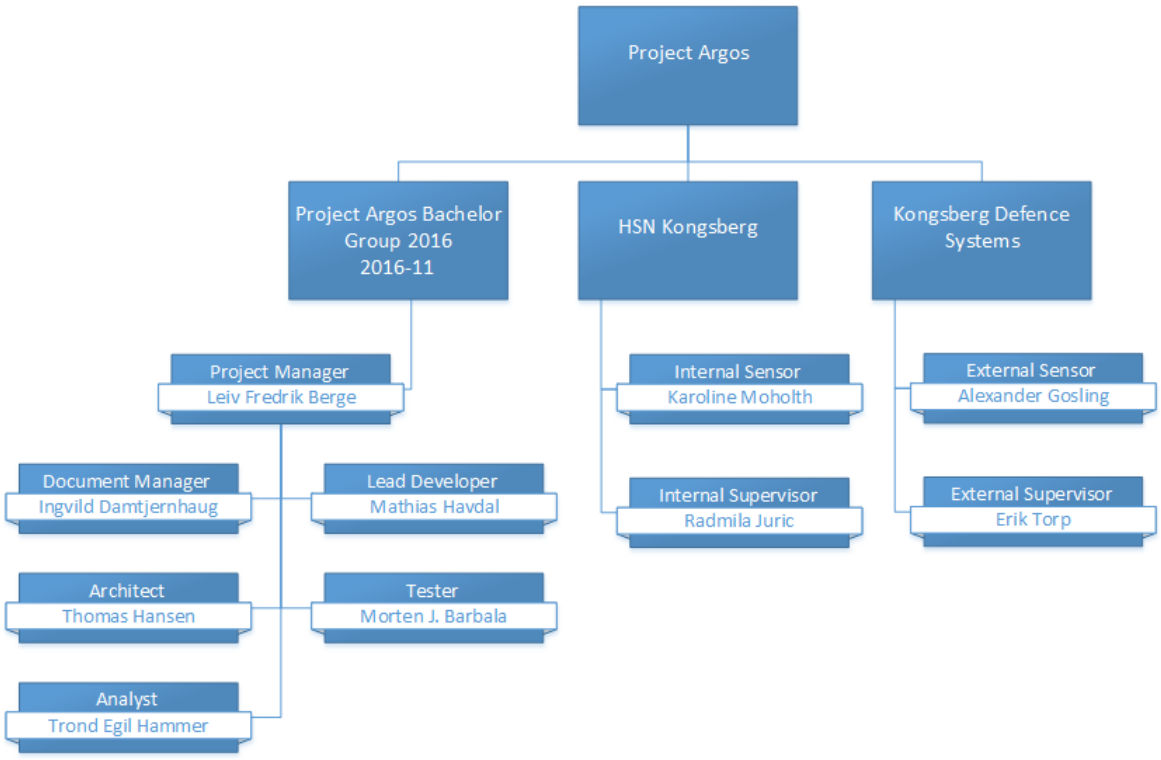

Figure 11: Project organization chart

9.1 Project Team

The bachelor team consists of six computer engineering students at University College of Southeast Norway (HSN). We are two students from virtual systems and four from embedded systems. One of the group members had a summer internship at KDS during the summer of 2015 in the same department as Project Argos, and this is how we got involved with the project. Project Argos is the final project of our bachelor degree. We selected our own team so we knew each other well before the project started, and this made the job of dividing roles easier. We selected the roles the first week of the project where we considered strengths, weaknesses and other factors that could impact the person's ability to perform their tasks.

9.2 Roles of the Team Members

OpenUP relies on a minimum number of roles to cover all necessary tasks in the project model. The project manager, tester, architect, analyst and developer are all compulsory roles in OpenUP. The roles define what the team members are responsible for, both tasks and work products. We have extended the model with two extra roles, the document manager and the lead developer. This is to make sure we meet all the requirements from the college in regards to documentations.

Figure 12: Generic role figure

Every team member has a role with well-defined areas of responsibility. Every team member is also considered to be developers, which means that all of us will be responsible for parts of the software. We have a flat organization structure with emphasis on joint decisions. By utilizing OpenUP we want to reduce risk and maximize stakeholder satisfaction. The work will be primarily performed from a single location where all group members share office, ensuring frequent and efficient communication.

Roles:

- Project manager: Leiv Fredrik Berge
- Analyst: Trond Egil Hammer
- Architect: Thomas Hansen
- Tester: Morten J. Barbala
- Document manager: Ingvild Damtjernhaug
- Lead developer: Mathias Havdal

9.2.1 Project Manager

Leiv Fredrik Berge
Software Engineer
452 41 510
lf@lfberge.com

The project manager will lead the planning of the project, coordinate the interaction between the project team and the stakeholders and keep the project moving in the right direction. He will also keep the team members motivated and focused on meeting the project goals and requirements.

Leiv Fredrik was a summer intern at KDS summer 2015 and was our link to the project owner and KDS. He is a natural leader and it felt right to give him the role as project manager. He has the ability to motivate and drive a project forward, is hard working and has a wide spectre of skills and knowledge.

- Is responsible for
  - Driving the project forward and ensuring its success
  - Acquiring acceptance from the stakeholder with regards to the product
  - Evaluating risk
  - Controlling and reducing risks with mitigation strategies
  - Ensuring that the project team delivers desired results for the project within the timeframe

Figure 13: Project manager

9.2.2 Analyst

Trond Egil Hammer
Software Engineer
482 93 433
trondehammer@gmail.com

The analyst is responsible for representing the customer and stakeholders needs. The analyst must understand the problem that the system is supposed to solve by gathering input from stakeholders, and he will capture and prioritize the requirements gathered from stakeholders.

The most important tasks of the analyst come early in the project, and this was a deciding factor in making Trond Egil the analyst. Trond Egil will get twins in April and that will naturally make him less available towards the later stages of the project.

- Is responsible for
  - Representing the stakeholders in internal settings
  - Ensuring that the stakeholders' needs are represented by requirements in the system
  - Ensuring that the architect of the system solves the problem the stakeholders need solved

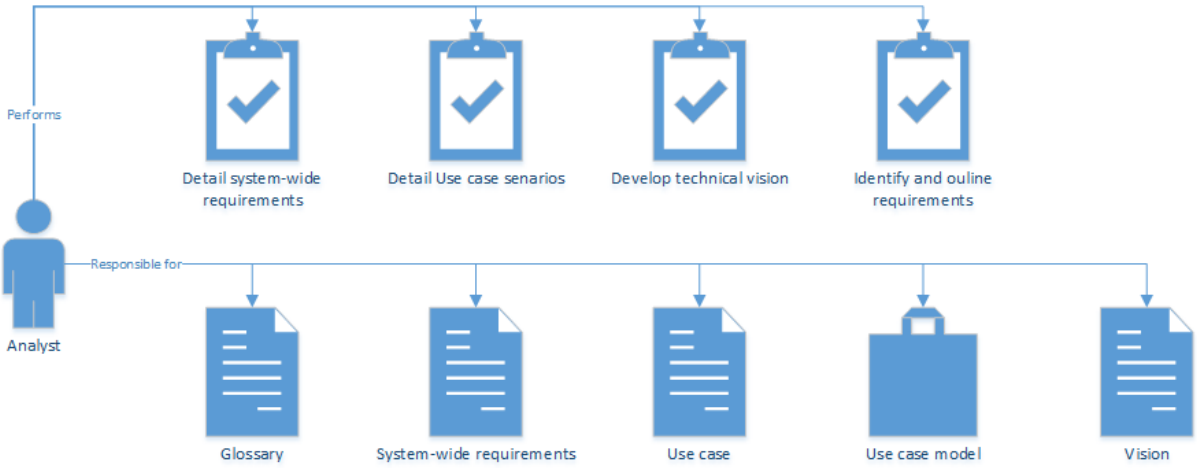

Figure 14: Analyst

9.2.3 Architect

Thomas Hansen
Software Engineer
406 28 686
mr.hansen@live.no

The architect is responsible for defining the major software architecture. This includes taking technical decisions that can limit the design and the implementation of the system. The architect will work with the project manager to reduce risk and create process models for the project, and he must also work closely with the analyst to ensure that the stakeholder requirements are covered in the technical architecture of the system.

Thomas is a strong developer with some experience with OpenGL. He understands the technical aspects of the project and can help the rest of the team to get on the same page with the software architecture.

- Is responsible for
  - Coordinating the technical design of the system
  - Identifying and documenting important aspects of the system with regards to the architecture
  - Documenting the reasoning for technical decisions with regards to stakeholder requirements and reducing technical risk
  - Ensuring the project group follows the technical architecture

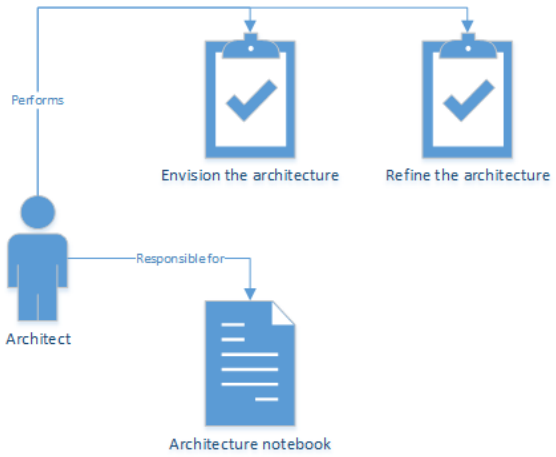

Figure 15: Architect

9.2.4 Tester

Morten J. Barbala
Software Engineer
918 30 834
netrom94@gmail.com

The tester is responsible for the core test activities. This includes identifying, defining, implementing and perform necessary tests as well as documenting and analysing the results.

Morten is detail oriented and a good developer. He has the ability to plan, execute and analyse the tests.

- Is responsible for
  - Identifying the necessary tests to perform
  - Identifying the appropriate test steps
  - Implementing individual tests
  - Documenting and verifying the performed tests
  - Analysing results and advising developers to improve the system
  - Ensuring that the team members are aware of test results

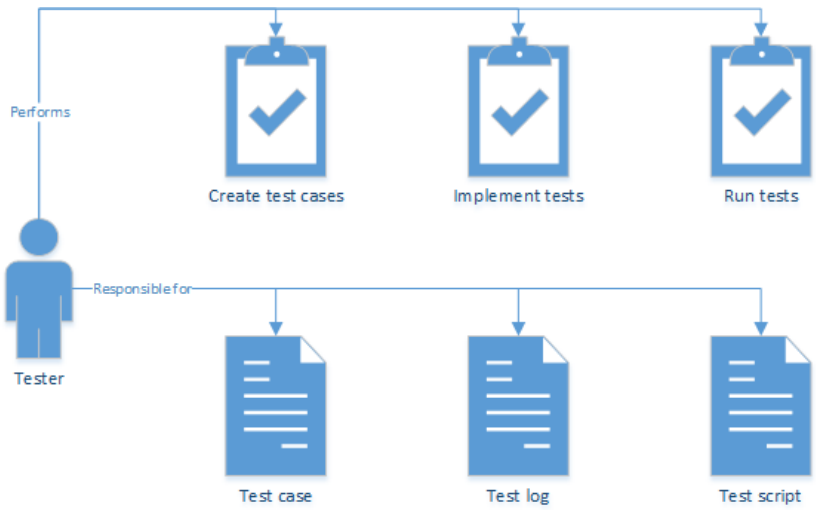

Figure 16: Tester

9.2.5 Document manager

Ingvild Damtjernhaug
Software Engineer
920 17 473
ingvild@damtjernhaug.no

The document manager will ensure that the project has a single, unified profile in documentation and presentation. She is responsible for ensuring that the documentation of the rest of the project team is following the set standards for the project. She is also responsible for creating and leading the work with presentations of the project.

Ingvild is a great organizer and has experience as a presenter. She has the ability to keep everyone focused and motivated on the key tasks at hand.

- Is responsible for
  - Creating document standards
  - Ensuring the quality of the team members' documentation
  - Leading the work with presentations of the project
  - Developing and designing the presentations

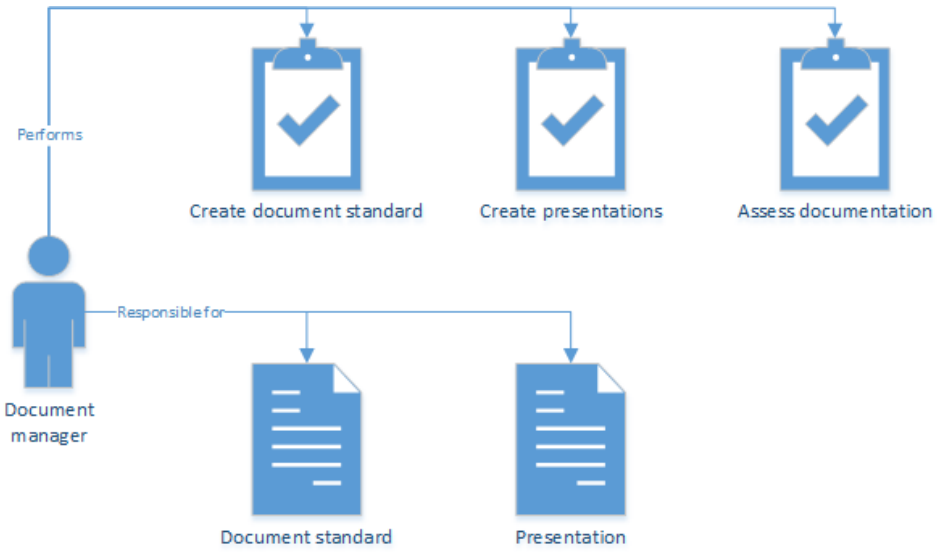

Figure 17: Document Manager

9.2.6 Lead Developer

Mathias Havdal
Software Engineer
452 74 007
matte3560@gmail.com

The lead developer will lead the work with development and, together with the project manager, ensure that the development team works in an efficient and productive manner.

Mathias is a great developer, with strong experience with C++. He has the abilities to help the rest of the team to produce high quality code in a timely manner.

- Is responsible for
  - Creating and reviewing the development iteration plan with the project manager
  - Creating a development iteration burndown report before each risk review meeting

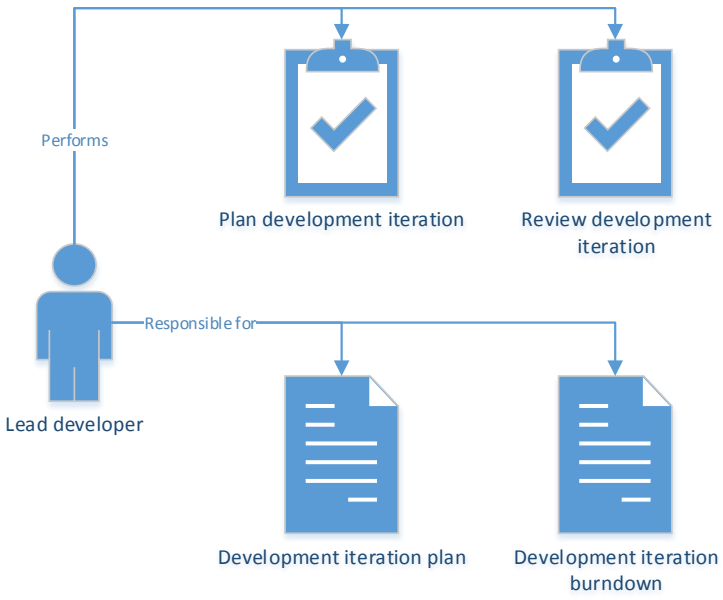

Figure 19: Lead developer

9.2.7 Developer

Every group member is a developer in the project. This is a mandatory role in OpenUP. The developers are responsible for developing all of the components of the software in the system. This includes designing the software to match the architecture and could mean creating prototyping for user experience, implementing, building, testing modules and integrating components in the code base.

- Is responsible for
  - Designing and developing parts of the system in line with the architecture
  - Building, implementing, testing and integrating components in the code base

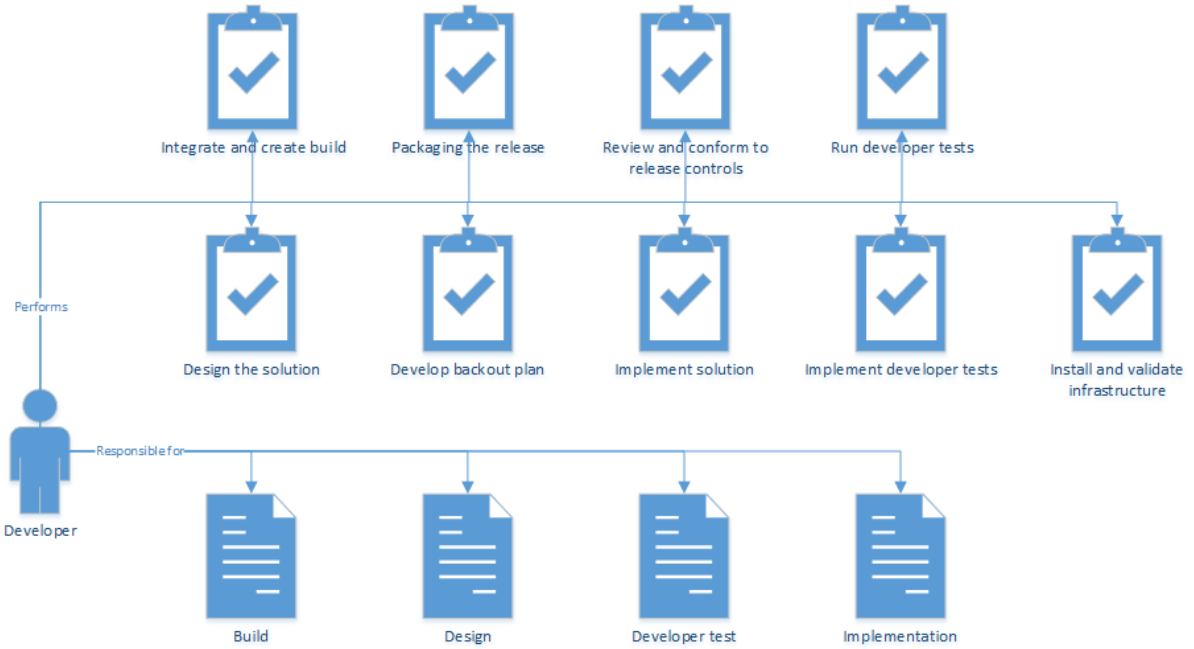

Figure 20: Developer

9.3 Sensors and Supervisors

In addition to the roles as stakeholders, some of the individuals and organizations involved in project Argos serve as sensors and supervisors.

External supervisor: Alexander Gosling
Mail: alexander.gosling@kongsberg.com

External supervisor will be the team contact person during the assignment and have a technical overview of the assignment. He will provide information and guidance to the group during the project. He will also have weekly meetings with the group to be updated on the work progress. External supervisor will be present on the presentations.

Internal supervisor: Radmila Juric
Mail: radmila.juric@hbv.no

Internal supervisor will act as the groups mentor. She will contribute to get the project moving in the right direction. She will be updated every week with work progress through follow-up documents.

External sensor: Alexander Gosling
Mail: alexander.gosling@kongsberg.com

External sensor will be present on the presentations. He will also be present at the meetings discussing the grades, both before and after each presentation. He will evaluate the group work based on the requirements and tasks given in the assignment.

Internal sensor: Karoline Moholth
Mail: Karoline.Moholth@hbv.no

Internal sensor will evaluate the group work throughout the project. Before the three presentations she will be studying the delivered documentation. She will also be present on meetings before and after the presentation as well as the presentations.

10. Project Praxis

We start every day with a stand up meeting. As the name indicates every team member stands upright in a circle facing each other. In turn, everyone tells what he or she did yesterday, what he/she accomplished, if there were any challenges and what the plans are for today. The other group members are not allowed to comment or ask questions during this session, but when everybody has had their turn we discuss any problems or questions. The daily stand up meetings give as a common understanding; everybody is up to date on the progress and it helps us keep track on the iterations. They also help everyone see where we are heading and where we are, in general; a nice way to give every group member attention and a moment to speak. The stand-up meetings are especially helpful if someone has been absent.

After the stand-up meeting we meet in front of the calendar, our project plan and our Kanban board. We have printed the calendar in a large format, which makes it visual to everyone, and it is easy to see timeframes, deadlines, iterations etc. We use small post-its/sticky notes so it is easy to change the calendar if plans are changing.

The project plan in the form of a Gantt diagram is also printed in a large format, which makes it easier to access and navigate. The Gantt diagram is digitally available for everyone, but as we always work together from the same place we experience that by making it the first thing you see every morning, the project plan becomes more elucidated and alive.

The Kanban board is a work and workflow visualization tool that enables us to optimize the flow of work. We have chosen to have a physical Kanban board hanging on the wall next to the Gantt diagram and our calendar. Kanban is actual the

Japanese word for “visual sign” or “card” [13]. Both small and large types of work are put on the board, and divided into four categories using coloured post-its: Code, documentation, testing and other. Furthermore, the board has six lanes: To do, in review, development, test, integrate and done. If a task comes up, someone will immediately put the task on the board to make sure tasks not are forgotten. By looking at the Kanban board together every morning after the stand-up meeting, no one need to wonder what to do that day. It helps us optimize our workflow and improves our effectiveness. Since tasks of all sizes are put on the board, it is convenient to turn to the board when you have some spare minutes, e.g. are waiting for something else or are done with a task and have just enough time to fix a little task or problem. The Kanban board greatly limits downtime and helps us communicate more easily what work needs to be done and when. The whole group appreciates the tactile, low-tech feel of the physical board.

We have also a section on the wall for the project goals, our personal goals and our questions to external supervisors and stakeholder. The different roles are also visualized in the form of diagrams on the wall. Together with the Kanban board, the calendar and the Gantt diagram, it makes the process more visual and gives an at-a-glance visibility of the current status of the work in progress and project overall.

We use Google Forms and Google Docs to register work hours. We have categorized the timesheet in over 50 different categories, covering all the different tasks according to the work-breakdown-structure. The team members are free to work whenever they want to, but everybody should be in our office between 9am and 3pm. We use Google Sheets to keep exact records of every hour spent. This gives a good overview in both spreadsheets and automatically generated diagrams. However, the work hour should preferably registered the same day the work took place, since the document manager registers the hours in a burndown report the next morning. The burndown reports gives an even more accurate insight in how the workhours are spent and makes it easier to locate discrepancy from the project plan.

Throughout the project we have five scheduled risk review meetings to monitor project progress and reduce risks. The risks will change during the process, both in probability and severity. In the meetings, the whole team sits down together and looks at the risk analyses. We discuss the risks, consider if different factors have increased or decreased the risk and finally adjust the risk analysis according to the result of or discussion.

The project will follow an iterative development process with continuous integration. We developed a stable architecture early in the process, which will iteratively be extended and adapted with new functionalities. This will ensure that we always have a functioning code base, as integration is a step in each iteration. Each iteration cycle is approximately two weeks while tasks are usually less than one day. Completed tasks are marked as 100%, incomplete tasks as 50%, and not started as 0%.

11. Communication Plan

The internal communication is primarily in person in our co-located office space at Krona. We have a calendar and the project plan on the wall, so messages and tasks

are written on the wall for everyone to see. However, if offsite communication is needed we have created a room in HipChat. This is a collaboration and communication tool from Atlassian, which can be tightly integrated with other tools from Atlassian, like the issue tracking application JIRA, collaboration tool Confluence and Mercurial client SourceTree. We were not allowed to connect our developer machines to the internet, so we were not able to use JIRA, Confluence or SourceTree, but HipChat still were useful to have an online communication tool. If there are particularly urgent messages we could use phone. All our code must stay in the offline development environment, meaning all development work must be performed in-house. The code is stored on our local server as a Mercurial repository. We use TortoiseHG as the server and the client so everyone can access the code. While developing, the default branch in Mercurial should always be tested and stable, so when new functionalities are added, we must create new branches and merge into default when the code is stable. Mercurial also gives us version control for the code, so if anything were to go wrong, we can always roll back. The main server for the documentation is also in the development environment. However particular documents can be checked out and worked on offsite, and then checked back into the server. We have not used any tools for this, but rather manual version control and a high degree of communication.

The communication with the internal supervisor, Radmila Juric, is primarily email based. While Radmila is out of the country we used a variety of tools to keep up the communication. The online video chat meetings are performed with Appear.in. Every Friday we send a weekly follow-up document to make sure our supervisor is up to date with the progress and challenges we have faced that week. The follow-up document contains the hours we have spent, the tasks that has been completed, the tasks that shall be complete next week, the problems we have had and the plan for next week.

The communication with external supervisor is primarily email based. However, the team will have a weekly status meeting with the stakeholders from KDS, represented by Alexander Gosling. In the weekly meeting we will describe how the project is progressing, milestones achieved and the work planned for next period.

An important part of the bachelor project is to show and inform the school, represented by Karoline Moholth. The communication with the school will primarily occur around the three scheduled presentations. This is where the team is expected to inform the school sensors about the process and project in general. For the final presentation we will create an event on Facebook so other students see our final presentation.

12. Risk Management Plan

We will perform five scheduled risk review meetings, reviewing the risk and updating the risk document. This will take into account changing conditions and the fact that we increase our knowledge of the system. Also some aspects of the project have increased severity the later in the project the faults were to occur. In these meetings we also look at the mitigation strategies to ensure that we reduce risks. As our project is mostly focused around software the most important mitigation strategies

are frequently and rigorously to perform backup of the documentation and the code base. To ensure that the loss would have minimal impact we perform double backup, with one of the hard drives off site. Risk management is an important part of our project management, and this is closely described in Risk Analysis. This is also closely tied to the project plan, iteration reports and burndown.

13. Budget

The budget is based on the actual costs and expenses in Project Argos. The cost associated with manpower and rental costs are not included in the budget.

Project Argos Bachelor 2016

What	Cost	Number	MVA	Total
Presentation	327,00	1	0 %	327,00
Office supplies	306,00	1	0 %	306,00
Poster	200,00	5	0 %	1000,00
Final print	500,00	1	0 %	500,00
Kowa LM6JC lenses	1 043,00	3	25 %	3 911,25
Ebus SDK license	857,14	1	0 %	857,14
Camera rig	126,70	1	0 %	126,7
SUM				7027,25

Project Argos summer 2015

What	Cost	Number	MVA	Total
Computer	22 726,00	1	25 %	28 407,50
Oculus	3 564,00	1	0	3 564,00
AV MAKO G-223C	19 294,52	4	25 %	96 472,60
Fujinon Fish Eye Lens 2.7mm 185 grader	7 003,27	1	25 %	8 754,09
HDI-Pro External Frame Grabber	11 086,27	1	25 %	13 857,84
24-port 19" Gigabit PoE Switch/Web Smart	7 040,80	1	25 %	8 801,00
Intel® PRO/1000 PT Quad Port	3 449,60	1	25 %	4 312,00
Webcams	299,00	2	0 %	598,00
T-shirts	165,00	5	0 %	825,00
Poster	200,00	1	0 %	200,00
SUM				165 792,03
TOTAL SUM				**172 819,28**

14. Project Life-cycle

Project Argos begun the summer of 2015 with the concept development and a functioning prototype of the system. The bachelor project of 2016 has further developed the functionality of the software, TinyArgos, in Project Argos. Moving forward Project Argos will have more summer students working on the project for the summer of 2016. The project will now start to be more multidisciplinary to work towards a functioning system that can actually be mounted on a vehicle. One member of the bachelor team will join the summer development team, ensuring continuity and speedier handover.

Bibliography

[1] Kongsberg Defence Systems, “Kongsberg Defence Systems,” Kongsberg Defence Systems, [Online]. Available: http://www.kongsberg.com/no/kds. [Accessed 15 February 2016].

[2] R. Balduino, “Eclipse.org,” [Online]. Available: http://eclipse.org/epf/general/OpenUP.pdf. [Accessed 10 February 2016].

[3] M. Fowler and J. Highsmith, “The Agile Manifesto, AgileAlliance.org,” August 2001. [Online]. Available: http://dimsboiv.uqac.ca/8INF851/web/part1/introcution/The_Agile_Manifesto.pdf. [Accessed 12 February 2016].

[4] Eclipse Foundation, “OpenUP Wiki,” [Online]. Available: http://epf.eclipse.org/wikis/openup. [Accessed 11 January 2016].

[5] Eclipse Foundation, “OpenUP Vision,” Eclipse, [Online]. Available: http://eclipse.org/epf/openup_component/openup_vision.php. [Accessed 12 February 2016].

[6] P. Kruchten, The Rational Unified Process An Introcution, Addison-Wesley Professional, 2004.

[7] K. Schwaber, Agile Project Management with Scrum, Microsoft Press, 2004.

[8] K. Beck, “Embracing Change with Extreme Programming,” *Computer,* no. October, pp. 70-77, 1999.

[9] B. Gustafsson, “OpenUP - The Best of Two Worlds,” *Methods & Tools,* no. Spring, 2008.

[10] P. Kroll and B. MacIsaac, Agility and Discipline Made Easy - Practices From OpenUP and RUP, Addison-Wesley Professional, 2006.

[11] P. Kroll, “Transitioning from waterfall to iterative development,” IBM Rational, 2004.

[12] E. M. Chocano, “A Comparative Study of Iterative Prototyping vs. Waterfall Process Applied To Small and Medium Sized Software Projects,” Massachusettes Institute of Technology, 2004.

[13] "Leankit," 2015. [Online]. Available: leankit.com/learn/kanban/kanban-board/. [Accessed Mars 2016].

Risk Analysis 2.0

Created by Leiv Fredrik Berge
04.02.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

Table of Contents

1. Document Overview

The purpose of the risk analysis is to show and mitigate the risks involved in the project. The risk analysis is updated after the scheduled risk review meetings. This document is used to categorize and create an overview of important risk factors.

Describes

- potential risks in the project.
- how likely they are to occur.
- how severe it is if they occur.
- mitigation strategies.
- how we lower chances of risk occurring and deal with a risk if it does occur.
- changes to risks after risk review meetings.

1.1 Document History

Version	Change	Date	Created by
0.1	First version	18.01.2016	Leiv Fredrik Berge
0.2	Added R3.5	19.01.2016	Leiv Fredrik Berge
0.3	Added correct standard	19.01.2016	Leiv Fredrik Berge
0.4	Translated to English	21.01.2016	Leiv Fredrik Berge
0.5	Added referenced documents	02.02.2016	Leiv Fredrik Berge
1.0	Changed ID-numbers, corrections	04.02.2016	Ingvild Damtjernhaug
1.1	Changed color on table header	16.02.2016	Trond Egil Hammer
1.2	Risk review	18.02.2016	Ingvild Damtjernhaug, Thomas Hansen, Trond Egil Hammer, Morten J. Barbala, Mathias Havdal, Leiv Fredrik Berge
1.3	Implemented DoD risk matrix standard. Minor corrections and clarifications	19.02.2016	Morten J. Barbala
1.4	Added lens risk	02.03.2016	Morten J. Barbala
1.5	Changed format on name and date in document history	14.04.2016	Trond Egil Hammer
1.6	Risk review, updated risk values	15.04.2016	Leiv Fredrik Berge Ingvild Damtjernhaug
1.7	Changed document overview	29.04.201	Ingvild Damtjernhaug
1.8	Minor changes, added references	11.05.2016	Trond Egil Hammer
1.9	Added spaces after headings, fixed risk calculations, corrected document overview	16.05.2016	Morten J. Barbala
1.10	Fixed header and footer. Fixed risk tables and added captions. Corrections and clarifications.	19.05.2016	Morten J. Barbala

2.0	Final review	20.05.2016	Ingvild Damtjernhaug, Morten J. Barbala, Thomas Hansen

1.2 Referenced Documents

Title	Document	Version
Glossary	doc-1113_glossary_2_0.docx	2.0

1.3 List of Figures

1.4 List of Tables

2. Risk Analysis

An important aspect with OpenUP is to handle and reduce risk in all iterations. The risk analysis is based on a risk matrix that sorts risks by probability and severity [1]. The figure 1 details an overview of all the risks we have identified in Project Argos. It is divided into three different top level categories of risks; hardware, system and human faults.

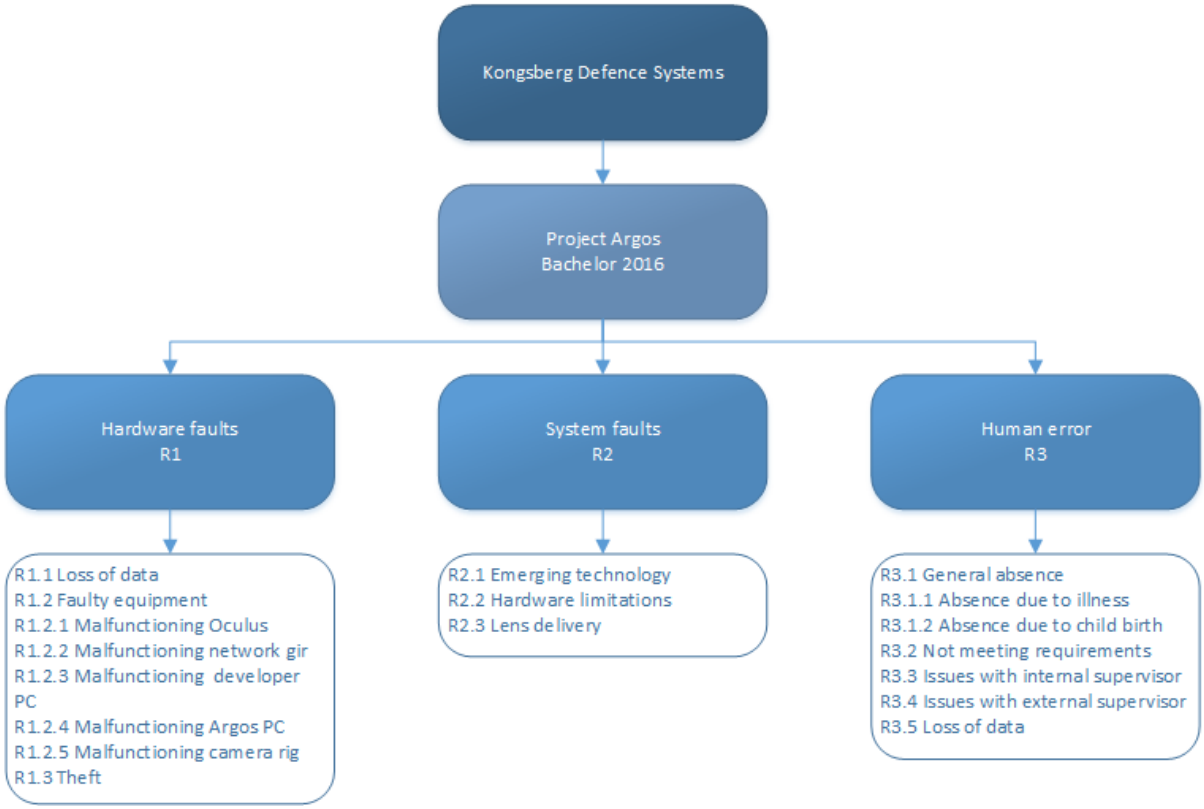

Figure 1: Risk overview diagram

The analysis is divided into main categories with corresponding risks listed below. Each risk is identified by an ID starting with “R”. The main categories start with a top level risk, which is generated from the average of the contained risk in that category. The effective risk is our consideration of the actual risk after mitigation strategies has been implemented. The risk scale is described in section 2.1. The numbers are based on our best estimations and research.

2.1 Hardware Faults

This section contains the faults and issues that can occur with our hardware components. This would be problems like a power supply in a developer PC that randomly fails or a hard drive crash.

ID	Risk	Description	Probability	Severity	Risk level	Effective risk
R1	**Hardware faults**	Technical or other hardware related failures	1,27	4,53	5,73	3,6
R1.1	**Loss of data** Date: 14.1.2016	Loss of data due to mechanical or technical faults. Faulty hard drives, USB-stick or other hardware malfunctions.	1	5	5	2
			Mitigation measures: Daily backup to memory stick. Weekly backup to offsite storage.			
R1.2	**Faulty equipment** Date: 14.1.2016	Dead on arrival or other faulty hardware	1,8	3,6	7,2	5,8
			Mitigation measures: Immediately contact supplier, get new hardware or refund.			
R1.2.1	**Malfunctioning Oculus** Date: 14.1.2016	Hard to get replacement	3	4	12	6
			Mitigation measures: R1.2			
R1.2.2	**Malfunctioning network gear** Date: 14.1.2016	Expensive equipment	1	4	4	4
			Mitigation measures: R1.2			
R1.2.3	**Malfunctioning developer PC** Date: 14.1.2016	R1.2	1	2	2	2
			Mitigation measures: R1.2			
R1.2.4	**Malfunctioning Argos PC** Date: 14.1.2016	R1.2	1	3	3	2
			Mitigation measures: R1.2			
R1.2.5	**Malfunctioning camera rig** Date: 14.1.2016	R1.2	3	5	15	15
			Mitigation measures: R1.2 Update 14.04.2016: One of the four cameras failed. This increases the severity if one more camera fails.			
R1.3	**Theft** Date: 14.1.2016	Equipment gets stolen	1	5	5	3
			Mitigation measures: Lockable office space. Lock equipment in cabinets. Cooperation with the groups we share office with.			

Table 1: Hardware faults risks

2.2 System Faults

This section contains the faults and issues that can occur with our system. This differs from the hardware faults in that this is not malfunctioning components, but rather issues due to lack of functionality or faults related to poor driver and hardware support in for example the VR headset. In other words, components that does function correctly, but do not satisfy our demands anyway.

ID	Risk	Description	Probability	Severity	Risk level	Effective risk
R2	**System faults**	Limitations or lack of features in equipment.	1,67	2,33	5,67	4,67
R2.1	**Emerging technology** Date: 14.1.2016	Cutting edge technology means documentation and hardware support may be limited.	2	4	8	6
			Mitigation measures: Communication with technical expertise in IDS. Communication with Oculus and camera supplier. Pugh-matrices on Oculus and Mako G223-C			
R2.2	**Hardware limitations** Date: 14.1.2016	Limitations in features or capabilities of cameras, computer components or other hardware.	3	3	9	8
			Mitigation measures: Research into technical solutions. Communications with producers and suppliers.			
R2.3	**Lens delivery** Date: 02.03.2016	Supplier is unable to deliver lenses in time	0	0	0	0
			No risk or mitigation, lenses arrived in a timely manner.			

Table 2: System faults risks

2.3 Human Errors

This section contains the faults and issues that can occur with our system, project or hardware due to human causes. This could be a group member being absent for a longer period of time, or loss of data due to negligence.

ID	Risk	Description	Probability	Severity	Risk level	Effective risk
R3	**Human errors**	Faults or problems arriving due to human causes.	1,9	3,5	6	2,9
R3.1	**Absence** Date: 14.1.2016	Longer general absence for a team member. A team member has to give information if absent more than three days.	3,5	2,5	8	3,5
			Mitigation measures: Social activities to create a friendly environment. Open communication channels.			
R3.1.1	**Absence due to illness** Date: 14.1.2016	R3.1. A team member has to give information if absent more than three days.	2	3	6	5
			Mitigation measures: R3.1			
R3.1.2	**Absence due to child birth** Date: 14.1.2016	R3.1. Trond Egil and his wife expect twins in April/May	5	2	10	2
			Mitigation measures: Allow Trond Egil to get more tasks early in the project.			
R3.2	**Not meeting requirements** Date: 14.1.2016	The project group doesn't satisfy stakeholder requirements.	2	4	8	3
			Mitigation measures: Use OpenUP as project model. Weekly meetings with stakeholders early in the process. Acquire acceptance for vision document and requirements document as early as possible. Open communication.			

R3.3	**Issues with internal supervisor** Date: 14.1.2016	The internal supervisor is absence, sick or for any other reason doesn't follow up with the project group.	1	4	4	3
			Mitigation measures: Open communication with supervisor. Communication with the college, represented by Olaf Graven.			
R3.4	**Issues with external supervisor** Date: 14.1.2016	The external supervisor is absence, sick or for any other reason doesn't follow up with the project group	1	4	4	3
			Mitigation measures: Open communication with supervisor. Communication with the college, represented by Olaf Graven. Open communication with IDS.			
R3.5	**Loss of data** Date: 14.1.2016	One or more group members overwrite, delete or in other ways destroy files.	2	3	6	2
			Mitigation measures: R1.1			

Table 3: Human errors risks

2.4 Risk Matrix

We use the standard from the US Department of Defence for risk management. [1] This matrix measures the impact on the project with regards to likelihood and probability. On the x axis the probability is mapped from unlikely to near certain, and on the y axis the consequence of the error to occur is mapped from minimal to severe. This means that the upper left corner of the matrix is the preferred area, and the bottom right area is critical. The risk that is in the area of 12 and above should be mitigated and closely watched.

Likelihood / Consequence		Not Likely	Low Likelihood	Likely	Highly Likely	Near Certainty
		~10%	~30%	~50%	~70%	~90%
		1	2	3	4	5
Minimal No, negligible impact	1	1	2	3	4	5
Minor The project is barely affected	2	2	4	6	8	10
Moderate The project experiences problems without stopping	3	3	6	9	12	15
Significant The project is highly affected and measures should be considered	4	4	8	12	16	20
Severe The project stops and measures must be evaluated.	5	5	10	15	20	25

Table 4: Risk matrix

2.5 Mitigation Strategies

This section contains a list of all the mitigation strategies mentioned in the risk lists. The table below shows the risks and the corresponding mitigation strategies.

Risk	Mitigation ID	Mitigation strategy
R1.1	M1.1.1	Daily backup to memory stick
R1.1	M1.1.1	Weekly backup to offsite storage
R1.2.x	M1.2.1	Lockable office space
R1.2.x	M1.2.2	Lockable cabinets for smaller items
R1.2.x	M1.2.3	Training in operation of equipment
R1.3	M1.3.1	Cooperation with the groups sharing office space
R2.1	M2.1.1	Cooperation with technical personnel at IDS
R2.1	M2.1.2	Communication with Oculus and GigE supplier
R2.2	M2.2.1	Research into technical solutions
R2.2	M2.2.2	Communication with producers and suppliers
R2.3	M2.3.1	Research into lenses and suppliers
R2.3	M2.3.2	Make it clear to suppliers we can only order lenses if the delivery is less than 3 weeks
R3.1.x	M3.1.1	Social activities for a positive environment
R3.1.x	M3.1.2	Open internal lines of communication
R3.1.x	M3.1.3	Facilitate that Trond Egil can work more in the early stages of the project
R3.2	M3.2.1	Use and follow up the OpenUP project model
R3.2	M3.2.2	Weekly stakeholder meeting in the early stages of the project
R3.2	M3.2.3	Acquire acceptance from stakeholder for vision and requirement documents
R3.3	M3.3.1	Open communication with supervisors
R3.3	M3.3.2	Communication with college, represented by Olaf Graven

Table 5: List of mitigations

3. Risk Review

During the risk review meetings some alteration to the risk analysis may occur. This reflects changes in the risks for various reasons. For example, the risk of something not being delivered can fall away as the item is delivered, meaning there is no longer a risk that we will not receive the item in time.

ID	Risk	Changes	Comments	Date
R1.2	Faulty equipment	Changed mitigation measures; immediately contact supplier, get new hardware or refund.	If supplier can't provide new hardware, try to find other suppliers.	18.02.16
R1.2.1	Malfunctioning Oculus	Added description; hard to get replacement, sold-out from Oculus and changed the severity from 2 to 3.	May purchase a second-hand from finn.no/ebay.com or wait for CV1 edition.	18.02.16
R1.2.2	Malfunction network gear	Added expensive equipment to description and changed the severity from 2 to 3.		18.02.16
R1.3	Theft	Changed the effective risk from 3 to 3,2	Effective risk is a little higher, lack of locked cabinets	18.02.16

R2.1	Emerging technology	Changed sentences in description, create Pugh-matrices Oculus and Mako G-223C and lenses.		18.02.16
R3.1	Absence	Changed the severity from 2,33 to 2(typo). A team member has to give information after 3 days of absence.		18.02.16
R3.1.2	Absence due to child birth	Changed description; Trond Egil and his wife expects twins in April/May, and a change in effective risk	Effective risk is changed from 5 to 2, known factor since September 2015.	18.02.16
R2.1	Emerging technology	Changed description; Key functionality has successfully been implemented	Reduce probability as less new functionality needs to be added to the project	15.04.16
R2.3	Lens delivery	Lenses have been delivered in a timely manner.		15.04.16
R1.2.1	Malfunctioning Oculus	Changed description; Increased severity if it fails, as it's less time to replace the Oculus.	It might be possible to borrow Oculus from HSN if ours were to fail	15.04.16
R1.2.2	Malfunctioning network gear	Changed description; Increased severity		15.04.16
R1.2.5	Malfunctioning camera rig	Changed description; One of the cameras failed. This increases the severity if another camera were to fail.	We have increased both severity and likelihood, as we might have underestimated the failure rate of the cameras. This is a critical area, and we have established communication with LMC, the supplier of the cameras.	15.04.16

Table 6: List of changes and comments of risk review meetings

Bibliography

[1] "United States Department of Defense, Risk Management Guide for DoD Acquisitions, August 2006.," [Online]. Available: www.jodypaul.com/SWE/DoDRiskMgmt2006.pdf. [Accessed 19 February 2016].

Requirements Document 2.0

Created by: Trond Egil Hammer
22.01.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

Table of Contents

1. Document Overview

The purpose of the requirement document is to give the reader a clear understanding of the system requirements. After reading this you should know the system requirements and our analysis of the system behaviour.

Describes

- the behaviour of the system
- the functional requirements of the system through use-cases.
- the system's functional, non-functional and environmental requirements and constraints.
- the traceability between requirements and test cases.

1.1 Document History

Ver.	Changes	Date	Created by
0.1	First version	22.01.2016	Ingvild Damtjernhaug, Trond Egil Hammer
0.2	New requirements	01.02.2016	All Bachelor team members
0.3	New requirements and document layout	02.02.2016	Thomas Hansen, Trond Egil Hammer
0.4	Changed colours	03.02.2016	Thomas Hansen
0.5	Change priority on some requirements	04.02.2016	Mathias Havdal, Morten J. Barbala
1.0	Added test ID and references. Removed A.NF.3 and renumbered requirements	04.02.2016	All Bachelor team members
1.1	Replaced A.F.1 with A.F.6, A.F.7, A.F.8.Changed User stories/scenarios to 2.1, Use case diagrams to 2.2 and descriptions to 2.3.Changed state to accepted Changed color on table header	16.02.2016	Morten J. Barbala Trond Egil Hammer
1.2	Updated use case diagram	08.03.2016	Leiv Fredrik Berge
1.3	Changed format on name and date in document history	10.03.2016	Trond Egil Hammer
1.4	Fix front page and layout. Rewriting, corrections and clarifications.	03.05.2016	Morten J. Barbala
1.5	Added requirements overview diagram	04.05.2016	Leiv Fredrik Berge
1.6	Added new latency test T.17	09.05.2016	Morten J. Barbala
1.7	Changed document overview and added requirement to test traceability table	11.05.2016	Trond Egil Hammer

1.8	Pre final review	18.05.2016	Thomas Hansen
1.9	Wrote description for all types of requirements.	20.05.2016	Morten J. Barbala
2.0	Final review	21.05.2016	Thomas Hansen, Trond Egil Hammer, Morten J. Barbala

1.2 Referenced Documents

Title	Document	Version
Test Specification	doc-1214_test_specification_2_0.docx	2.0
Glossary	doc-1113_glossary_2_0	2.0

1.3 List of Figures

1.4 List of Tables

2. Use Cases

Use cases are produced during the early stages of the project. They help us to visualize the functional requirements of the system in addition to giving a basis for sequence diagrams and eventually class diagrams. Use case diagrams and the associated documents help to analyse the behaviour of the system. The use cases in this document are distinguished by unique ID numbers for identification and traceability, starting with the letters UC and then a number.

2.1 User Stories and Scenarios

User story (driving):
Optimal:
The user needs to drive his tank through a dangerous area. The user gets into the driver seat of the tank and turns on the Argos system. When the user is ready to drive the user puts on the VR googles. In the goggles the user sees live video from the outside cameras. When the user turns his head, the view in the VR googles turns as well, showing a continuous view of the surroundings. In the VR view the user sees a compass and a map that can be used to navigate. The user drives safely through the dangerous area this way. When the user reaches a safe area, stops the tank, takes off his VR googles and turns off the system.

Fail 1:
The user is driving his armoured vehicle through a dangerous zone. Suddenly, one of the cameras stops working.

Fail 2:
While driving through a dangerous zone, a hostile appears nearby.

Fail 3:
Other failures in the video system.

User Story (Record):
Optimal:
The user is about to drive through an area of interest. Having video footage of this area could be useful for later training or other investigation. Knowing this, the user starts a recording. The user will be given an indication that a recording is ongoing. This will help the user remember to stop the recording when leaving the area of interest, so that storage space is not wasted.

Fail 1:
Storage space is completely full. The user is given a notification.

User Story (Playback):
Optimal:
The user needs to drive through an area that is dangerous. This area is difficult to navigate, and it is vital that the user does not make any wrong turns or need to stop to figure out where to go. There are several key landmarks along the way that the

user can use to navigate successfully through this area. The route has previously been navigated successfully and recorded. The user will go through the recorded video using the VR goggles, having the ability to look around and take note of any significant obstacles and landmarks. To speed up the learning process, video can skip forwards and backwards as well as pausing. This way the user can view the critical parts of the video with ease.

2.2 Use Case Diagram

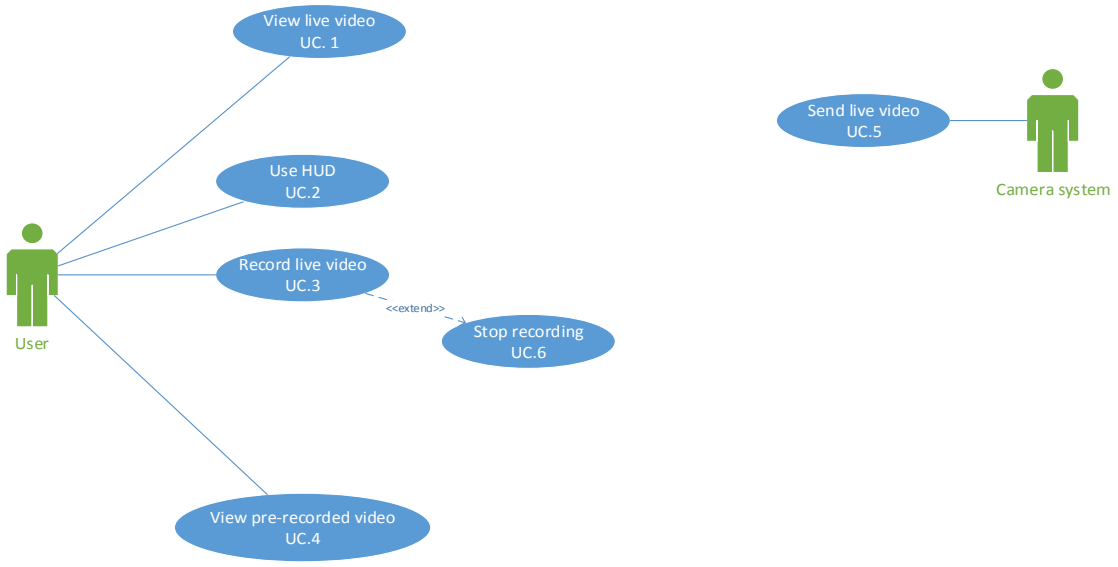

Figure 1: Use case diagram

2.3 Descriptions

View live video:
ID: UC.1
Actor: User
Goal: See surroundings outside of the vehicle

Actor	System
Turns head around to view the surroundings outside	Moves the virtual world to match the head movement

Table 1: Use Case description for UC.1

Use HUD:
ID: UC.2
Actor: User
Goal: Get information from other systems

Actor	System
Looks at the heads up display inside the virtual world	Show relevant information in the virtual world

Table 2: Use Case description for UC.2

Record live video:
ID: UC.3
Actor: User
Goal: Record video for later viewing

Actor	System
The user starts the recording	Record live video until the user stops the recording

Table 3: Use Case description for UC.3

View pre-recorded video:
ID: UC.4
Actor: User
Goal: View pre-recorded video

Actor	System
Starts the pre-recorded video	Shows the pre-recorded video in the virtual world. The user can pause and skip forwards and backwards in playback.

Table 4: Use Case description for UC.4

Send live video:
ID: UC.5
Actor: Camera system
Goal: Send live video to software system

Actor	System
Sends live video from all cameras	Receive the video data

Table 5: Use Case description for UC.5

Send live video:
ID: UC.6
Actor: User
Goal: Stop the recording

Actor	System
Sends command to stop recording	Stops the recording, finish writing video recording to disk.

Table 6: Use Case description for UC.6

3. System Requirements

This section will cover the different types of system requirements: Functional, non-functional, constraints and environmental. It also contains a table over traceability between requirements and tests. The requirements have a priority according to their importance for the project, and the ID numbers are explained like this: Priority.Type.Number.

Priority	Explanation
A	These requirements must be achieved
B	These requirements should be achieved
C	These requirements should be achieved, but have a lower priority

Table 7: Requirements priority levels

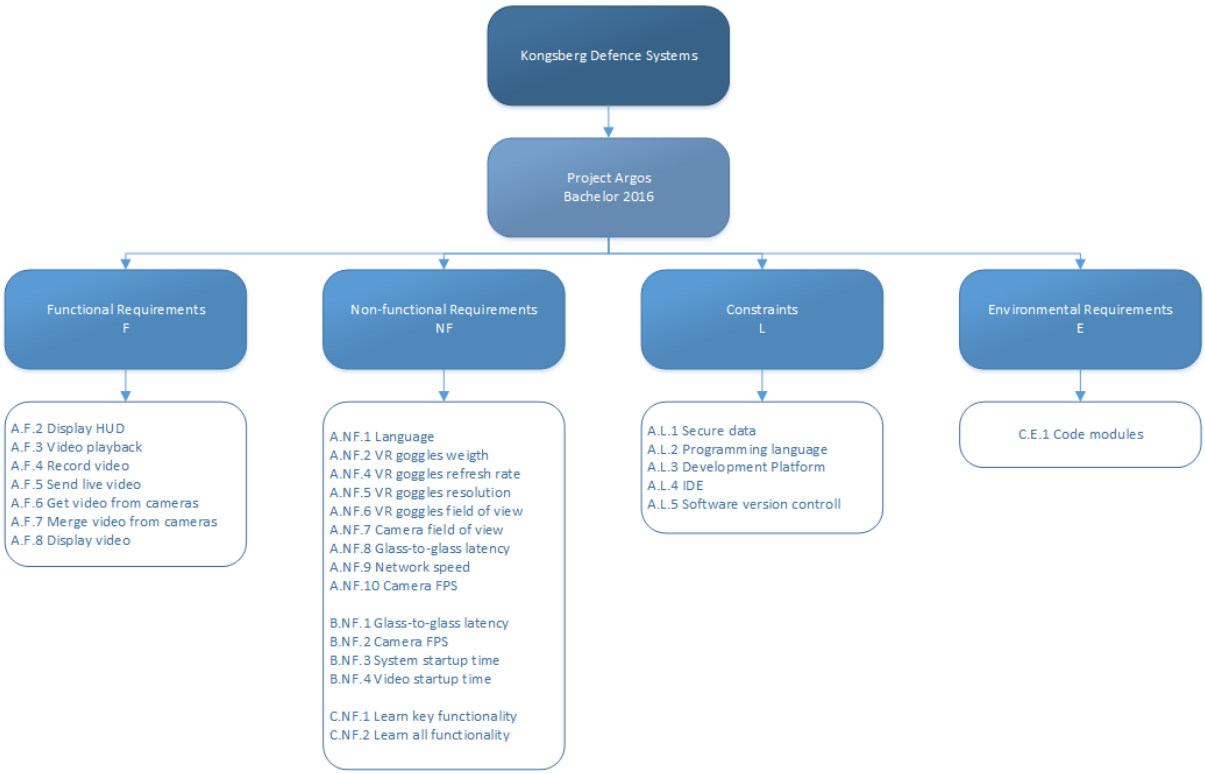

Figure 2: Overview of all requirements

3.1 Functional Requirements

Functional requirements are system requirements which show functionalities the system absolutely must have. All functional requirements have priority A and must be fulfilled for the project to be considered a success. They originate from the project description we received from Kongsberg Defence Systems (KDS), and the early meetings with stakeholders. The use cases are visualizations of the functional requirements.

ID	Title	Description	Date	State	Test ID	Use Case ID	Origin
~~A.F.1~~	**Show image**	The system must show a seamless image in real time	14.01.2016	Removed	T.1	UC.1	KDS
	Comment	Replaced by A.F.6, A.F.7, A.F.8					
A.F.2	**Display HUD**	Display must show information on a HUD	14.01.2016	Accepted	T.2	UC.2	KDS
	Comment						
A.F.3	**Playback**	The system must be able to playback recorded video	14.01.2016	Accepted	T.4	UC.4	KDS
	Comment						
A.F.4	**Record**	The system must be able to record at least 2 minutes of video	14.01.2016	Accepted	T.3	UC.3 UC.6	KDS
	Comment						
A.F.5	**Send live video**	The cameras must be able to send live video to system	14.01.2016	Accepted	T.1 T.3	UC.5	KDS
	Comment						
A.F.6	**Get video from cameras**	The system must be able to capture and handle video from all four cameras	16.02.2016	Accepted	T.1	UC.5	KDS
	Comment						
A.F.7	**Merge video from cameras**	The system must be able merge video from all cameras to one continuous image	16.02.2016	Accepted	T.1	UC.5	KDS
	Comment						
A.F.8	**Display video**	The system must be able to display the continuous image in VR goggles	16.02.2016	Accepted	T.1	UC.5	KDS
	Comment						

Table 8: Functional requirements

3.2 Non – Functional Requirements

Non-functional requirements are system requirements which show various non-functional aspects of the system. Rather than describing functionalities, they describe properties which need to be in place to enable the system to provide the required functionalities. They also describe properties which should be achieved to improve the overall quality of the system.

ID	Title	Description	Date	State	Test ID	Origin
A.NF.1	**Language**	Information and messages must appear in English	02.02.2016	Accepted	T.13	Bachelor team
	Comment					
A.NF.2	**VR Goggles**	The VR goggles must weigh less than 400 grams	02.02.2016	Accepted	T.10	Bachelor team
	Comment					
A.NF.3	**VR Goggles**	The VR goggles must have refresh rate higher than 75 Hz	02.02.2016	Accepted	T.10	Bachelor team
	Comment					
A.NF.4	**VR Goggles**	The VR goggles must have at least 960x1080 resolution per eye	02.02.2016	Accepted	T.10	Bachelor team
	Comment					
A.NF.5	**VR Goggles**	The VR googles must have at least 100 degrees field of view	02.02.2016	Accepted	T.10	Bachelor team
	Comment					
A.NF.6	**Camera FOV**	Cameras must have a combined field of view at least 180 degrees	02.02.2016	Accepted	T.1	Bachelor team
	Comment					
A.NF.7	**Latency**	Glass to glass latency must be less than 75 ms	02.02.2016	Accepted	T.1 T.9 T.17	Bachelor team
	Comment					
A.NF.8	**Network Speed**	Network must have a minimum throughput of	02.02.2016	Accepted	T.12	Bachelor team

		512 MB/s				
	Comment					
A.NF.9	**Camera FPS**	Cameras must be able to capture at least 50 FPS	02.02.2016	Accepted	T.11	Bachelor team
	Comment					

Table 9: Non-functional requirements, priority A

ID	Title	Description	Date	State	Test ID	Origin
B.NF.1	**Latency**	Glass to glass latency should be less than 20 ms	02.02.2016	Accepted	T.9 T.17	Bachelor team
	Comment					
B.NF.2	**Camera FPS**	Cameras should be able to capture at least 75 FPS	02.02.2016	Accepted	T.11	Bachelor team
	Comment					
B.NF.3	**System start-up**	System start-up should take maximum 20 seconds	02.02.2016	Accepted	T.5	Bachelor team
	Comment					
B.NF.4	**Up and running**	Showing live video in VR goggles should take at most 5 clicks	02.02.2016	Accepted	T.6	Bachelor team
	Comment					

Table 10: Non-functional requirements, priority B

ID	Title	Description	Date	State	Test ID	Origin
C.NF.1	**Learning**	It should be possible to learn key functionality in 10 minutes	02.02.2016	Accepted	T.7	Bachelor team
	Comment					
C.NF.2	**Learning**	It should be possible to master all functionality in less than 1 day	02.02.2016	Accepted	T.8	Bachelor team
	Comment					

Table 11: Non-functional requirements, priority C

3.3 Constraints

Constraints are requirements for the development team, not the system, and originate directly from stakeholders similarly to the functional requirements. This could include any standards or aspects of projects which need to be followed to satisfy the project owner. The constraints for Project Argos set the development environment: IDE, programming language, version control etc.

ID	Title	Description	Date	State	Test ID	Origin
A.L.1	**Secure data**	The system must not be available to the public	02.02.2016	Accepted	T.15	KDS
	Comment					
A.L.2	**Programming Language**	Programming language must be C++	02.02.2016	Accepted	T.15	KDS
	Comment					
A.L.3	**Development Platform**	OS must be Microsoft Windows 7	02.02.2016	Accepted	T.15	KDS
	Comment					
A.L.4	**IDE**	IDE must be Microsoft Visual Studio	02.02.2016	Accepted	T.15	KDS
	Comment					
A.L.5	**Software Version Control system**	Version Control system must be Mercurial (Hg)	02.02.2016	Accepted	T.15	KDS
	Comment					

Table 12: Constraints

3.4 Environmental Requirements

Environmental requirements are system requirements which reach beyond the confines of the system. We only have one environmental requirement as Project Argos is still in an early phase, being developed in lab for proof of concept and research. We have to make sure the code is modular, i.e. extendable and modifiable. In future development, literal environmental requirements as well as interfaces to outside systems come into play.

ID	Title	Description	Date	State	Test ID	Origin
C.E.1	**Code modules**	Code should have modules to support new Oculus hardware	02.02.2016	Accepted	T.16	Bachelor team
	Comment	Code is updated to 0.8.0.0, the latest SDK that officially support Oculus Rift DK2.				

Table 13: Environmental requirements

3.5 Traceability to Tests

Requirement	Test
~~A.F.1~~	~~T.1~~
A.F.2	T.2
A.F.3	T.4
A.F.4	T.3 T.16
A.F.5	T.1 T.3
A.F.6	T.1
A.F.7	T.1
A.F.8	T.1
A.NF.1	T.13
A.NF.2	T.10
A.NF.3	T.10
A.NF.4	T.10
A.NF.5	T.10
A.NF.6	T.1
A.NF.7	T.1 T.9 T.17
A.NF.8	T.12
A.NF.9	T.11
B.NF.1	T.9 T.17
B.NF.2	T.11
B.NF.3	T.5
B.NF.4	T.6
C.NF.1	T.7
C.NF.2	T.8
A.L.1	T.14
A.L.2	T.14
A.L.3	T.14
A.L.4	T.14
A.L.5	T.14
C.E.1	T.15

Table 14: Requirement to test traceability

Test Specification 2.0

Created by: Morten J. Barbala
22.01.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

KONGSBERG

Table of Contents

1. Document Overview

The purpose of this document is to describe the approach for verifying the system requirements from stakeholders and the early requirements analysis. We define a standard test procedure and detail four different types of tests. Furthermore, this document contains our test plan, which shows traceability between test cases and requirements, states dependencies and sets the order in which to run all tests. The plan describes three test phases for verification and validation of the system.

Describes

- the test procedure, how to perform tests.
- test logs, how to document testing.
- all types of tests, differences and purposes of each type.
- test cases: Test steps, acceptance criteria, which requirements are tested etc.
- module tests: Test steps, acceptance criteria, which module is tested etc.
- reference tests: Test steps and which test cases the results are relevant for.
- the final acceptance test: Test steps and acceptance criteria.
- the test plan: Traceability to requirements, test dependencies and test phases for verification and validation of the system.

1.1 Document History

Version	Changes	Date	Created by
0.1	Create the document	22.01.2016	Morten J. Barbala
0.2	Fill in points according to requirements	01.02.2016	Morten J. Barbala, Mathias Havdal
0.3	Establish and implement test spec. template, add more test cases	02.02.2016	Morten J. Barbala, Mathias Havdal
0.4	Add test procedure flow chart and test report template	03.02.2016	Morten J. Barbala, Mathias Havdal
0.5	Add priority to existing test cases, and add test cases to cover remaining requirements. Added references to other documents.	04.02.2016	Morten J. Barbala, Mathias Havdal
1.0	Removed T.14 and renumbered tests Small corrections before first submission	04.02.2016	Morten J. Barbala, Mathias Havdal
1.1	Replaced A.F.1 with A.F.6, A.F.7, A.F.8 Changed colour on table header	16.02.2016	Morten J. Barbala, Trond Egil Hammer
1.2	Added disk writing test T.16 and fixed test numbering	25.03.2016	Morten J. Barbala
1.3	Describe types of tests, added module tests	14.04.2016	Morten J. Barbala
1.4	Corrections in tests T.10 and T.3	18.04.2016	Morten J. Barbala

1.5	Add delay test for vimba viewer, RT.1	22.04.2016	Morten J. Barbala
1.6	Added new test for glass-to-glass latency: T.17	25.04.2016	Morten J. Barbala
1.7	Fix front page. Rename test reports to test logs. Correct module test MT.1	02.05.2016	Morten J. Barbala
1.8	Rework module tests and write tests for remaining software modules	03.05.2016	Morten J. Barbala
1.9	Added acceptance tests	05.05.2016	Morten J. Barbala
1.10	Rewrote introduction and objectives into document overview. Added bullet points to document overview. Added test dependencies and test plan.	11.05.2016	Morten J. Barbala
1.11	Added introduction for all sections. Added section for test overview and updated test overview diagram. Restructured document.	12.05.2016	Morten J. Barbala
1.12	Wrote remaining module tests: MT.3, MT.4, MT.5	13.05.2016	Morten J. Barbala, Mathias Havdal
1.13	Updated test overview. Corrections and clarifications.	15.05.2016	Morten J. Barbala
1.14	Added long skips to MT.2, finalized module tests.	18.05.2016	Thomas Hansen, Mathias Havdal, Morten J. Barbala
1.15	Added section for test results	20.05.2016	Morten J. Barbala
2.0	Final review	21.05.2016	Trond Egil Hammer, Morten J. Barbala, Thomas Hansen

1.2 Referenced Documents

Title	Document	Version
Requirements document	doc-1213_requirements_2_0.docx	2.0

1.3 List of Figures

1.4 List of Tables

2. Test Procedure

The test procedure is a set of steps required to perform tests. The purpose of defining a test procedure is to ensure the team that the proper documentation is generated, and that any changes, faults or errors are the results of system changes rather than imprecise testing. This section provides an activity diagram, which describes the test procedure, as well as templates for test logs.

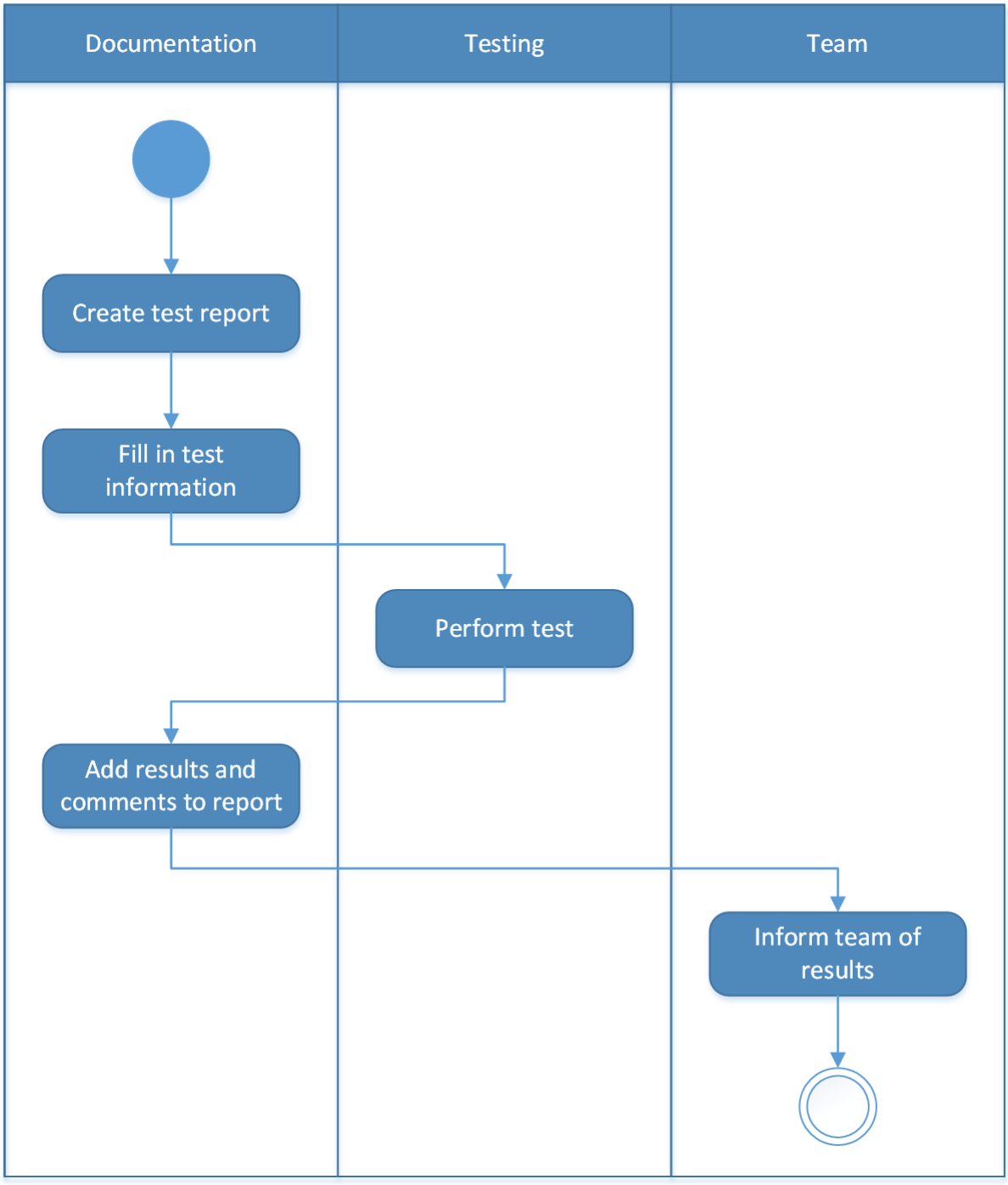

Figure 1: Activity diagram of test procedures

2.1 Test Logs

The test logs serve as references and documentation. They show which requirements that have been verified and which requirements that still need some work, as well as comments and recommendations from the tester. Various project documents also reference tests to show specific numbers, e.g. disk writing speed. The test logs contain information using the following templates, in addition to any attachments and data related to the test.

2.1.1 Test Case and Module Test Logs

Test ID		Date	[dd.mm.yy]
Test Number		Result	[Pass/Fail]
Tester			
Specific Results			
Comment			

Table 1: Test case and module test logs

2.1.2 Reference Test Logs

Test ID		Date	[dd.mm.yy]
Test Number			
Tester			
Specific Results			
Comment			

Table 2: Reference test logs

2.1.2 Acceptance Test Logs

Acceptance test [internal] or [with stakeholder]		Date	[dd.mm.yy]
		Result	[Pass/Fail]
Tester	All team members, [name of stakeholder]		
Specific Results			
Comment			

Table 3: Acceptance test logs

3. Test Overview

We have four main types of tests: System wide test cases, software specific module test, reference tests for documentation and comparison, and a final acceptance test. Each developer will also perform a number of developer tests for classes, functions, code snippets etc. while developing the main software modules, but these will not be documented because they all amount to: Change this code, compile and check if project builds or errors persist. Instead, the module tests provide the documentation for completed software modules while the test cases verify the functionalities. The final acceptance test, which is run twice, validates the results of the project with stakeholders.

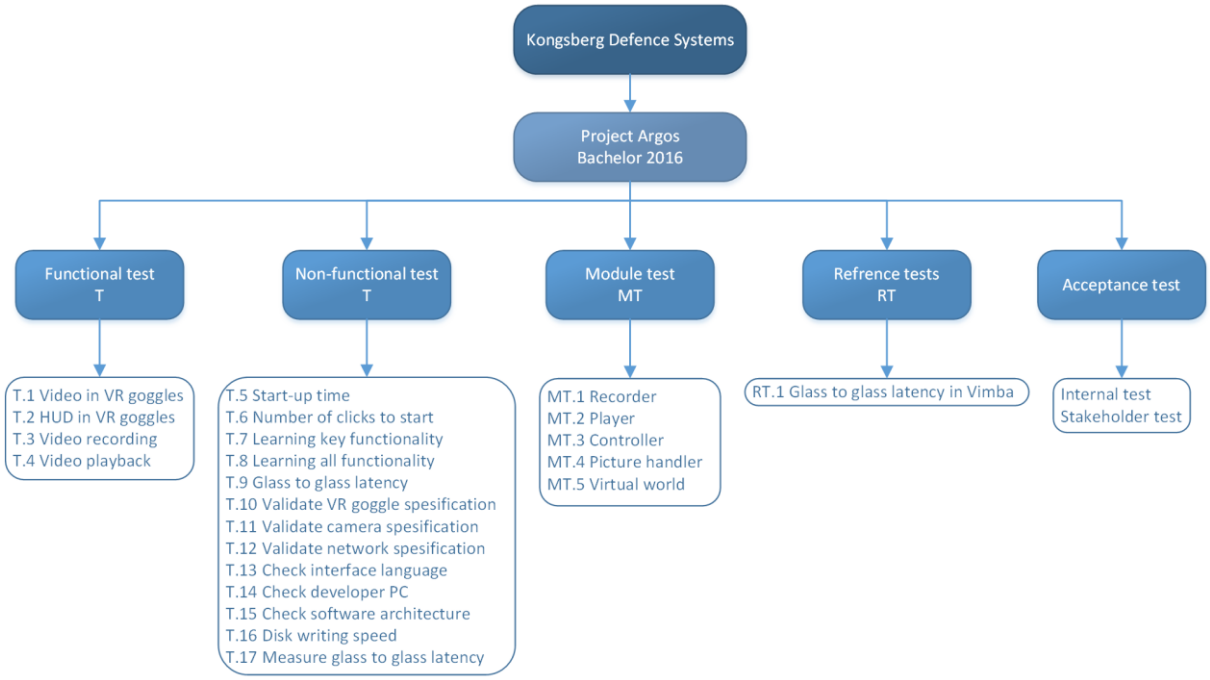

Figure 2: Test overview diagram

3.1 Test Cases

The test cases are system tests, which are run to document fulfilment of requirements. The test cases show which requirements are verified through a successful test and are split into two groups: functional and non-functional. However, a single test can cover multiple requirements, both functional and non-functional, and a single requirement can be covered by multiple tests. The test case group is therefore set according to if the test case covers functional requirements or not. Similarly, the priority is set after the highest requirement priority level.

3.1.1 Test Cases for Functional Requirements

Test ID	T.1	**Priority**	High
Requirements Tested	~~A.F.1~~, A.F.5, A.F.6, A.F.7, A.F.8, A.NF.6, A.NF.7		
Test Description	Video in VR goggles		
Resources	VR goggles, camera system		
Test steps	1. Turn on system 2. Put on VR goggles 3. Turn head and view the virtual world		
Acceptance criteria	• Virtual world is continuous for the intended field of view • Changes in real world are perceived as instant in virtual world • No perceivable seams, faults or errors in video that break user immersion		

Table 4: Functional test T.1

Test ID	T.2	**Priority**	High
Requirements Tested	A.F.2		
Test Description	HUD in VR goggles		
Resources	VR goggles, camera system		
Test steps	1. Turn on system 2. Put on VR goggles 3. View information on HUD		
Acceptance criteria	• HUD is available at all times • User can view and process information		

Table 5: Functional test T.2

Test ID	T.3	**Priority**	High
Requirements Tested	A.F.4, A.F.5		
Test Description	Video recording		
Resources	VR goggles, camera system, stopwatch		
Test steps	1. Turn on system 2. Start recording 3. Put on VR goggles 4. Take off VR goggles 5. Stop recording after 20 seconds 6. Open recording directory		

Acceptance criteria	• User can see an indicator showing that the system is recording video • After recording user can see files in storage • Files are in raw format and are not 0KB

Table 6: Functional test T.3

Test ID	T.4	**Priority**	High
Requirements Tested	A.F.3		
Test Description	Video playback		
Resources	VR goggles		
Test steps	1. Turn on system 2. Start playback of pre-recorded video 3. Put on VR goggles 4. Look around 5. Pause video 6. Un-pause video 7. Skip backwards in video 8. Skip forwards in video 9. Stop playback of pre-recorded video		
Acceptance criteria	• Video is continuous across intended field of view • User can manipulate playback as intended (pause/play, skip forward/backward)		

Table 7: Functional test T.4

3.1.2 Test Cases for Non-Functional Requirements

Test ID	T.5	**Priority**	Medium
Requirements Tested	B.NF.3		
Test Description	Start-up time		
Resources	Stopwatch		
Test steps	1. Restart computer to clear file system cache 2. Wait 5 minutes for start-up programs to launch 3. Launch software and start stopwatch 4. When main menu loads, stop stopwatch 5. Repeat steps 1-4 three times and record the results		
Acceptance criteria	• Average software launch time must be less than 20 seconds		

Table 8: Non-functional test T.5

Test ID	T.6	Priority	Medium
Requirements Tested	B.NF.4		
Test Description	Number of clicks to start live video in VR goggles		
Resources	None		
Test steps	1. Click shortcut to launch software 2. Navigate through menus to start live video in VR goggles, while counting the number of clicks		
Acceptance criteria	• The number of clicks to start live video in VR goggles must be less than or equal to 5		

Table 9: Non-functional test T.6

Test ID	T.7	Priority	Low
Requirements Tested	C.NF.1		
Test Description	Learning key functionality		
Resources	Untrained user		
Test steps	1. The user is given a 10 minute introduction to the system		
Acceptance criteria	• The user must be able to start recording and/or get live video in VR goggles without assistance		

Table 10: Non-functional test T.7

Test ID	T.8	Priority	Low
Requirements Tested	C.NF.2		
Test Description	Learning all functionality		
Resources	Untrained user		
Test steps	1. The user is given training 2. User is given instruction manual		
Acceptance criteria	• The user must have mastered all functionality after one day of training and studying the manual		

Table 11: Non-functional test T.8

Test ID	T.9	Priority	High
Requirements Tested	A.NF.7, B.NF.1		
Test Description	Measure glass-to-glass delay		
Resources	High speed (framerate) video camera, computer monitor, video player with frame by frame skip		
Test steps	1. Point VR camera at screen showing solid colour 2. Start video feed in VR goggles and move view to		

	show solid colour recorded by camera 3. Using a high speed video camera, record both screen showing solid colour and output on computer monitor 4. Change colour displayed 5. Analyse video from high speed camera frame-by-frame. Count the number of frames from when the colour on the first display changes, until the colour changes in the on computer monitor 6. Calculate the delay using the number of frames and the framerate of the high frequency camera
Acceptance criteria	• Calculated delay must be less than 75 milliseconds • Calculated delay should be less than 50 milliseconds • Calculated delay ideally less than 20 milliseconds

Table 12: Non-functional test T.9

Test ID	T.10	**Priority**	High
Requirements Tested	A.NF.2, A.NF.3, A.NF.4, A.NF.5		
Test Description	Verify VR goggle specifications		
Resources	VR goggle spec sheet		
Test steps	1. Read relevant information on spec sheet		
Acceptance criteria	• Weight must not exceed 500 grams • Resolution per eye must be at least 960x1080 • Refresh rate must be at least 75 Hz • Viewing optics must have at least 100 degrees field of view		

Table 13: Non-functional test T.10

Test ID	T.11	**Priority**	High
Requirements Tested	A.NF.9, B.NF.2		
Test Description	Verify camera specifications		
Resources	Camera spec sheet		
Test steps	1. Read relevant information on spec sheet		
Acceptance criteria	• Must have a framerate of at least 50 Hz • Should have a framerate of 75 Hz		

Table 14: Non-functional test T.11

Test ID	T.12	Priority	High
Requirements Tested	A.NF.8		
Test Description	Verify network specifications		
Resources	Spec sheet for switch, spec sheet for NIC on VR computer		
Test steps	1. Read relevant information on spec sheets		
Acceptance criteria	• Must have a throughput of at least 512 MB/s		

Table 15: Non-functional test T.12

Test ID	T.13	Priority	High
Requirements Tested	A.NF.1		
Test Description	Check interface language		
Resources	None		
Test steps	1. Start system 2. Navigate through menus 3. Check language of all text		
Acceptance criteria	• All text must be written in English		

Table 16: Non-functional test T.13

Test ID	T.14	Priority	High
Requirements Tested	A.L.1, A.L.2, A.L.3, A.L.4, A.L.5		
Test Description	Check dev computers		
Resources	None		
Test steps	1. Turn on computer 2. Check operating system version 3. Check internet connection 4. Check IDE 5. Check project configuration in IDE 6. Check version control system		
Acceptance criteria	• Computer must be running Microsoft Windows 7 Enterprise SP1 • It must not be connected to the internet • The IDE must be Microsoft Visual Studio 15 • Project must be developed in C++ • The version control system must be Mercurial (Hg)		

Table 17: Non-functional test T.14

Test ID	T.15	Priority	Low
Requirements Tested	C.E.1		
Test Description	Check software architecture		
Resources	None		
Test steps	1. View architecture diagrams		
Acceptance criteria	• The architecture should be modular to allow for implementation of new Oculus SDK versions and hardware		

Table 18: Non-functional test T.15

Test ID	T.16	Priority	Medium
Requirements Tested	A.F.4		
Test Description	Disk writing speed		
Resources	None		
Test steps	1. Turn on computer 2. Open HD Tune 3. Select disk to be tested 4. Run Benchmark with 64 KB block size 5. Run Benchmark with 512 KB block size 6. Run Benchmark with 4 MB block size 7. Run Benchmark with 8 MB block size 8. Calculate average writing speed		
Acceptance criteria	• The average writing speed is greater than 512MB/s		

Table 19: Non-functional test T.16

Test ID	T.17	Priority	High
Requirements Tested	A.NF.7, B.NF.1		
Test Description	Measure glass-to-glass delay in VR goggles		
Resources	High speed (framerate) video camera, computer monitor, video player with frame by frame skip		
Test steps	1. Point VR camera at screen showing solid colour 2. Start video feed in VR goggles and move view to show solid colour recorded by camera 3. Using a high frequency video camera, record both screen showing solid colour and output in VR goggles 4. Change colour displayed 5. Analyse video from high frequency camera frame-by-frame. Count the number of frames from when the colour on the first display changes, until the colour changes in the VR goggles.		

	6. Calculate the delay using the number of frames and the framerate of the high frequency camera
Acceptance criteria	• Calculated delay must be less than 75 milliseconds • Calculated delay should be less than 50 milliseconds • Calculated delay ideally less than 20 milliseconds

Table 20: Non-functional test T.17

3.2 Module Tests

Module tests are software tests, which are run to verify the modules in the chosen software architecture. Rather than documenting the fulfilment of system requirements, the module tests check if the separate modules in the software produce the expected results, e.g. outputs to connected interfaces and files on disk. Module tests are performed after the system tests and acceptance tests and they verify the software design, not system functionalities.

Test ID	MT.1	**Priority**	High
Module tested	Recorder		
Test Description	TinyArgos recording system		
Resources	None		
Test steps	1. Turn on computer 2. Run TinyArgos.exe 3. Wait until TinyArgos is running with video in VR-goggles 4. Press ‘R’ on keyboard to start recording 5. Record 2 minutes of video 6. Press ‘R’ on keyboard to stop recording 7. Repeat steps 4-6 two more times		
Acceptance criteria	• User can start recording • Icon is visible in top-right corner to indicate recording • User can stop recording • System creates directories for each recording session using the current timestamp in the format “yyyymmdd.hhmmss” • For each recording session there are subdirectories for all cameras. • Camera directories contain the recorded files numbered from 0 to the last frame captured.		

Table 21: Module test MT.1

Test ID	MT.2	**Priority**	High
Module tested	Player		
Test Description	TinyArgos playback system		
Resources	None		
Test steps	1. Turn on computer 2. Run TinyArgos.exe 3. Wait until TinyArgos is running with video in VR-goggles 4. Press ‘P’ on keyboard open file explorer 5. Navigate to recording directory 6. Select config file for recording to start playback 7. Press ‘right ctrl’ to pause playback 8. Press ‘right ctrl’ to resume playback 9. Press ‘left’ to skip backwards 10. Press ‘right’ to skip forwards 11. Press ‘Page Down’ to make long skip forward 12. Press ‘Delete’ to make long skip backward 13. Press ‘1’ to go back to live video		
Acceptance criteria	• File explorer allows user to select specific recording • Virtual world shows video from all recorded cameras • Video is continuous across intended field of view and synchronized • User can pause/resume playback • User can skip forwards/backwards • User can stop playback		

Table 22: Module test MT.2

Test ID	MT.3	Priority	High
Module tested	Controller		
Test Description	Test of TinyArgos controller		
Resources	None		
Test steps	1. Turn on computer 2. Run TinyArgos.exe 3. Wait until TinyArgos is running with video in VR-goggles 4. Use keys ‘1, 2, 3, 4, 5’ to change main config file 5. Use keys ‘6, 7, 8, 9, 0’ to change HUD config file 6. Use VR goggles to look around while changing config files		

Acceptance criteria	• System loads default config file and shows live video in virtual world • User can load config files to show different camera layouts • User can load config files to show different HUD layouts • VR-goggles operate the same way in all views

Table 23: Module test MT.3

Test ID	MT.4	**Priority**	High
Module tested	Picture handler		
Test Description	Picture handling system		
Resources	None		
Test steps	1. Turn on computer 2. Run TinyArgos.exe 3. Wait until TinyArgos is running with video in VR-goggles 4. Look around using VR goggles 5. Start playback 6. Stop playback		
Acceptance criteria	• Framerate is perceived as constant • Video has no perceivable corruptions or artefacts • Live video and playback is perceived as identical		

Table 24: Module test MT.4

Test ID	MT.5	**Priority**	High
Module tested	Virtual world		
Test Description	OpenGL and Oculus Rift		
Resources	None		
Test steps	1. Turn on computer 2. Run TinyArgos.exe 3. Wait until TinyArgos is running with video in VR-goggles 4. Look around using VR goggles 5. Use keys ‘0-9’ to load different configs 6. Press ‘space’ to recalibrate motion tracking 7. Start playback 8. Stop playback		

Acceptance criteria	• Motion tracking remains responsive at all times • No stuttering when changing config files • Motion tracking is correctly recalibrated when pressing ‘space’ in live view and in playback.

Table 25: Module test MT.5

3.3 Reference Tests

Reference tests do not verify fulfilment of requirements, nor software modules. They serve as documentation and test data for comparison with test cases. We only have one reference tests: RT.1. This test is about measuring the delay inherent in the Allied vision cameras to compare with T.9. The purpose is to document hardware limitation and check if our software introduces extra latency.

Test ID	RT.1
Relevant Test Case	T.9
Test Description	Glass-to-glass delay in Vimba Viewer
Resources	High frequency video camera, two screens
Test steps	1. Point VR camera at screen showing solid colour 2. Start video feed in Vimba Viewer and move view to show solid colour recorded by camera 3. Using a high frequency video camera, record both screen showing solid colour, and screen showing output in Vimba Viewer 4. Change colour displayed 5. Analyse video from high frequency camera frame-by-frame. Count the number of frames from when the colour on the first display changes, until the colour in Vimba Viewer changes. 6. Calculate the delay using the number of frames and the framerate (Hz) of the high frequency camera

Table 26: Reference tests RT.1

3.4 Acceptance Tests

In the final iteration of the project, one of the key activities is acceptance testing. An acceptance test is performed after all functionalities have been implemented and tested independently, or after the development has ended because of deadlines and lack of time. The purpose is to show the results of the development and document the final state of the system that the team has produced. The acceptance test is first run internally with the team so all results and any faults, bugs or shortcomings are known before a final test with the stakeholders.

Acceptance test	
Test Description	Validation of all system functionalities
Test steps	1. Turn on computer 2. Run TinyArgos.exe 3. Put on VR goggles 4. Using VR goggles, look around in the virtual world 5. Move view to the left edge of the live video and move head to the right edge to see full field of view 6. Temporarily remove VR goggles if needed to help find keys on keyboard 7. Use keys “0-9” to show different virtual worlds 8. Press “R” to start recording 9. Create movement and recognizable events in front of cameras 10. Press “R” to stop recording 11. Remove VR goggles 12. Press “P” to open file explorer for playback 13. Select directory with newly recorded video and select config.xml file to start playback 14. Put on VR goggles 15. Using VR Goggles, look around in the virtual world and note the recognizable events in the playback 16. Use controls “right ctrl”, “left” and “right” to play/pause, skip backwards and skip forwards respectively 17. Use keys “1-5” to load standard config file and stop playback 18. Exit TinyArgos
Acceptance criteria	• Virtual world has multiple possible layouts of live video and HUD • The live video from multiple cameras is continuous, synchronized and seamless • Moving head while wearing VR goggles changes view in virtual world smoothly • The system can record video while giving the user and indicator of ongoing recording • The system can play recorded video with working playback controls • The user can only distinguish between live video and playback because of playback controls • The user can stop playback • System exits without errors

Table 27: Acceptance test

4. Test Plan

As the project evolves the need for different types of tests emerges. The number of tests will also grow throughout the development of the system, and it is important to have a test plan. The tests not only ensure the team that requirements are fulfilled and the software verified, but also provide the proper documentation for the system. Furthermore, some tests depend on results or test data from other test and have to be performed in specific order. The first two phases of our test plan verify the system requirements and functionalities, and validate the system with the stakeholders, while the third and final phase verifies the software. The final phase also contains two low priority tests about teaching an outsider the system, to see if any keyboard controls or interfaces should be changed before final delivery.

4.1 Traceability to Requirements

Test	Requirement
T.1	~~A.F.1~~ A.F.5 A.F.6 A.F.7 A.F.8 A.NF.6 A.NF.7
T.2	A.F.2
T.3	A.F.4 A.F.5
T.4	A.F.3
T.5	B.NF.3
T.6	B.NF.4
T.7	C.NF.1
T.8	C.NF.2
T.9	A.NF.7 B.NF.1
T.10	A.NF.2 A.NF.3 A.NF.4 A.NF.5
T.11	A.NF.9 B.NF.2
T.12	A.NF.8
T.13	A.NF.1
T.14	A.L.1 A.L.2 A.L.3 A.L.4 A.L.5
T.15	C.E.1

T.16	A.F.4
T.17	A.NF.7 B.NF.1

Table 28: Test to requirement traceability

4.2 Independent Tests

The following tests have no dependencies and can be run in any order.

Test ID
T.1
T.2
T.5
T.6
T.9
T.10
T.11
T.12
T.13
T.14
T.15
T.16
T.17
MT.1
MT.2
MT.3
MT.4
MT.5
RT.1

Table 29: Independent tests

4.3 Dependent Tests

The following tests have dependencies

Test ID	Dependencies
T.3	T.2
T.4	T.3
T.7	T.2, T.3, T.4
T.8	T.2, T.3, T.4
Internal acceptance test	All tests for functional requirements: T.1, T.2, T.3, T.4
Stakeholder acceptance test	All tests from phase 1, Internal acceptance test

Table 30: Dependent tests

4.4 Test Phases

Phase	**1/3**	**Start date**	**25.02.2016**
Description		Various tests to help with development and document completed subsystems. Order is not important except for dependent tests. Any tests not completed in phase 1 are performed in the order specified in phases 2 and 3.	
Tests		Test cases: • T.1 • T.2 • T.3 • T.4 • T.5 • T.6 • T.9 • T.10 • T.11 • T.12 • T.13 • T.14 • T.16 • T.17 Reference tests: • RT.1	
End date		06.05.2016	

Table 31: Test phase 1

Phase	**2/3**	**Start date**	**09.05.2016**
Description		Acceptance testing and validation of the system. Order is important.	
Tests		1. Any remaining tests for functional requirements 2. Internal acceptance test 3. All remaining tests from phase 1 4. Stakeholder acceptance test	
End date		13.05.2016	

Table 32: Test phase 2

Phase	3/3	Start date	16.05.2016
Description		Verification of software architecture and software modules, tests about teaching functionalities to an outsider. Order is not important	
Tests		• T.7 • T.8 • T.15 • MT.1 • MT.2 • MT.3 • MT.4 • MT.5	
End date		20.05.2016	

Table 33: Test phase 3

5. Test Results

Phase 1, 25.02-06.05: Testing successful.

Test	Result	Date (earliest)	Comment
T.1	-	-	Performed in phase 2
T.2	PASS	06.05.2016	
T.3	PASS	19.04.2016	
T.4	PASS	02.05.2016	
T.5	-	-	Performed in phase 2
T.6	-	-	Performed in phase 2
T.9	-	-	Deprecated by test T.17
T.10	PASS	18.04.2016	
T.11	PASS	18.04.2016	
T.12	PASS	19.04.2016	
T.13	PASS	06.05.2016	
T.14	PASS	19.04.2016	
T.16	PASS	25.02.2016	
T.17	PASS	22.04.2016	
RT.1	-	-	Results not relevant for test phase

Table 34: Test results for phase 1

Phase 2, 09.05-13.05: Testing successful.

Test	Result	Date (earliest)	Comment
T.1	PASS	10.05.2016	Delayed from phase 1
Internal acceptance test	PASS	10.05.2016	No bugs or missing functionalities. Oculus hardware limitation discovered, see test log for details.
T.5	PASS	11.05.2016	Delayed from phase 1
T.6	PASS	11.05.2016	Delayed from phase 1
Stakeholder acceptance test	PASS	11.05.2016	No changes from internal acceptance test. Stakeholder impressed by results.

Table 35: Test results for phase 2

Phase 3, 16.05-20.05: Testing successful.

Test	Result	Date (earliest)	Comment
T.7	PASS	18.05.16	
T.8	-	-	We did not have the resources to perform this test. Test has low priority and is not critical for the system
MT.15	PASS	11.05.16	
MT.1	PASS	18.05.16	
MT.2	PASS	18.05.16	
MT.3	PASS	18.05.16	
MT.4	PASS	18.05.16	
MT.5	PASS	18.05.16	

Table 36: Test results for phase 3

Test Logs 1.0

Created by: Morten J. Barbala
22.05.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

HSN

KONGSBERG

Table of Contents

Test Cases

This document contains a collection of all the test logs. The logs refers to a particular test that has been performed with date, result and comments.

T.1

Test ID	T.1	Date	10.05.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	Continuous video Changes are perceived as instant Slight seam visible		
Comment	Seams have minimal impact on user immersion. Pleora eBUS watermark visible on all camera feeds.		

Test ID	T.1	Date	10.05.2016
Test Number	2	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	Continuous video Changes are perceived as instant Slight seam visible		
Comment	Seams have minimal impact on user immersion. Pleora eBUS watermark is no longer visible after adding license.		

T.2

Test ID	T.2	Date	06.05.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	Spinner HUD-element Recording icon Loading icon		
Comment	Loading icon is off center, consider changing position		

T.3

Test ID	T.3	Date	19.04.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala, Thomas Hansen		
Specific Results	Red recording icon in top right corner Files in recording directory with current date and time: 20160419.120146 Files numbered and in raw format No files with size 0KB		
Comment	Tested with three Mako G-223C cameras Same number of raw files in subdirectories for each camera		

Test ID	T.3	Date	02.05.2016
Test Number	2	Result	**PASS**
Tester	Morten J. Barbala, Mathias Havdal		
Specific Results	Red recording icon in top right corner Files in recording directory with current date and time in correct format Files numbered and in raw format Same number of raw files in subdirectories for each camera No files with size 0KB		
Comment	Tested with three Mako G-223C cameras		

T.4

Test ID	T.4	Date	28.04.2016
Test Number	1	Result	**FAIL**
Tester	Morten J. Barbala		
Specific Results	Can pause/unpause and skip forwards/backwards. Video is continuous, but with visible seams. Can start new playback, but not stop playback		
Comment	Missing textures to indicate playback is loading Delay when unpausing playback Delay when skipping forwards/backwards		

Test ID	T.4	Date	02.05.2016
Test Number	2	Result	**PASS**
Tester	Morten J. Barbala, Mathias Havdal		
Specific Results	Can pause/unpause and skip forwards/backwards Video is continuous, but with visible seams Can start new playback and stop playback		
Comment	Missing textures to indicate playback is loading Delay when unpausing playback and skipping forwards/backwards		

Test ID	T.4	Date	11.05.2016
Test Number	3	Result	**PASS**
Tester	Morten J. Barbala, Mathias Havdal		
Specific Results	Can pause/unpause and skip forwards/backwards Video is continuous, but with small visible seams Can start new playback and stop playback		
Comment	Icon to indicate playback loading is now visible on HUD. Seams do not break user immersion and the user forgets about them after longer use. Slight delay when unpausing playback and skipping forwards/backwards.		

T.5

Test ID	T.5	Date	11.05.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	Startup times: 4.48, 4.36, 4,41 Average: 4,42		
Comment	Fast startup time because of SSD in main Argos PC		

T.6

Test ID	T.6	Date	11.05.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala, Mathias Havdal		
Specific Results	1-3 clicks		
Comment	Number of clicks depends on if the user will use the default configs or not.		

T.7

Test ID	T.7	Date	18.05.2016
Test Number	1	Result	**PASS**
Tester	Thomas Hansen, Morten J. Barbala		
Specific Results	The test user was given written instructions and oral instructions before using the system. The user started the system with ease and started and stopped a recording.		
Comment	The test user had a bit problem resetting the view when first putting on the headset, but managed to figure it out after a short while.		

T.9

Test ID	T.9	Date	22.04.2016
Test Number	1	Result	**FAIL**
Tester	Morten J. Barbala		
Specific Results	33 frames delay 33/240 → 0.1375 seconds delay		
Comment	Test performed on three cameras Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz Filmed preview on screen, not inside the VR-googles, and this might give extra delay		

Test ID	T.9	Date	22.04.2016
Test Number	2	Result	**FAIL**
Tester	Morten J. Barbala		
Specific Results	30 frames delay 30/240 → 0.125 seconds delay		
Comment	Test performed on one cameras Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz Filmed preview on screen, not inside the VR-googles, and this might give extra delay		

T.10

Test ID	T.10	Date	18.04.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	Weight: 440 grams (without cable) Resolution per eye: 960x1080 Refresh rate: 75 Hz Field of view: 100 degrees		
Comment	VR-goggles are Oculus Rift DK2 [1] Specifications were found on article comparing DK1 to DK2 Consider contacting Oculus for official specifications and/or technical manual.		

T.11

Test ID	T.11	Date	18.04.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	Selected framerates from Allied Vision technical manual: 1088x2048: 49.5 FPS 900x2048: 59.7 FPS 700x2048: 76.6 FPS		
Comment	Cameras are Mako G-223C [2] FPS varies with ROI (region of interest) height Reducing width does not affect the FPS We can adjust resolution to get desired framerate		

T.12

Test ID	T.12	Date	19.04.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	Four ports 1 Gigabit per second per port		
Comment	NIC is Intel Pro/1000 PT Quad Port LP [3] 1 Gigabit per second is only with ideal load balancing		

T.13

Test ID	T.13	Date	06.05.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	File explorer, HUD and interface in English		
Comment	Interface language depends on operative system default language		

T.14

Test ID	T.14	Date	19.04.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	All computers running Microsoft Windows 7 Enterprise SP1 All computers only connected to local network All developers using C++ in Microsoft Visual Studio 15 Version control is Mercurial (HG) on local server		
Comments	Test delayed as team members struggled with VS15 installation		

T.15

Test ID	T.15	Date	18.05.2016
Test Number	1	Result	**PASS**
Tester	Thomas Hansen		
Specific Results	The architecture is modular		
Comment	The architecture is separated into 4 different components. Each component is its own module.		

T.16

Test ID	T.16	Date	25.02.2016
Test Number	1	Result	**PASS**
Tester	Morten J. Barbala		
Specific Results	Benchmark with 64 KB block size: 508.0 MB/s Benchmark with 512 KB block size: 973.1 MB/s Benchmark with 4 MB block size: 1208.8 MB/s Benchmark with 8 MB block size: 1248.5 MB/s **Average writing speed: 984.6 MB/S**		
Comments	Performed test on array with three disks in RAID 0		

T.17

Test ID	T.17	Date	22.04.2016
Test Number	1	Result	**FAIL**
Tester	Morten J. Barbala, Leiv Fredrik Berge		
Specific Results	Observations (number of frames delay): 18, 20		
Comment	Test performed on one camera Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz Filmed inside oculus rift goggles, no downscaling		

Test ID	T.17	Date	22.04.2016
Test Number	2	Result	**PASS**
Tester	Morten J. Barbala, Leiv Fredrik Berge		
Specific Results	Observations (number of frames delay): 20, 18, 18 21, 15, 16, 16, 21 17, 16, 20, 17 Average delay: 17.92 frames, 74.7ms		
Comment	Test performed on one camera in VR goggles High-speed Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz		

Test ID	T.17	Date	22.04.2016
Test Number	3	Result	**PASS**
Tester	Morten J. Barbala, Leiv Fredrik Berge		
Specific Results	Observations (number of frames delay): 15, 13, 18, 15 18, 22, 19 Average delay: 17.14 frames, 71.4ms		
Comment	Test performed on three cameras in VR goggles High-speed Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz		

Test ID	T.17	Date	09.05.2016
Test Number	4	Result	**PASS**
Tester	Morten J. Barbala, Trond Egil Hammer, Mathias Havdal		
Specific Results	Observations (number of frames delay): 16, 17, 17 16, 15, 17 Average delay: 16,33 frames, 68,1ms		
Comment	Test performed on one camera in VR goggles using new PBO system High-speed Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz		

Test ID	T.17	Date	09.05.2016
Test Number	5	Result	**PASS**
Tester	Morten J. Barbala, Trond Egil Hammer, Mathias Havdal		
Specific Results	Observations (number of frames delay): 16, 17, 17 19, 18, 17 Average delay: 17,33 frames, 72,2ms		
Comment	Test performed on three cameras in VR goggles using new PBO system High-speed Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz		

Module Tests

MT.1

Test ID	MT.1	Date	18.05.2016
Test Number	1	Result	**PASS**
Tester	Thomas Hansen		
Specific Results	Recording started when ‘R’ was pressed, rec icon was visible and recording stopped when ‘R’ was pressed. Directories were created containing all cameras and raw image files from the cameras.		
Comment	The test was preformed 3 times with the same result.		

MT.2

Test ID	MT.2	Date	18.05.2016
Test Number	1	Result	**PASS**
Tester	Thomas Hansen		
Specific Results	File explorer opens when ‘P’ is pressed. File explorer opens in last recording directory Right control pause and un-pause playback Right and left skip 10 seconds forward and backwards Page down and delete skip 1 min forward and backwards ‘1’ takes you back to live video		
Comment	If the user doesn’t know how files are stored it might be hard to find the right recording. If there is no recording file to find, the file explorer opens the computers “documents” folder.		

MT.3

Test ID	MT.3	Date	18.05.2016
Test Number	1	Result	**PASS**
Tester	Thomas Hansen		
Specific Results	The program starts with default config file for both surface and HUD All surface config file work with all HUD config files by pressing 1-5 and 6-0 for every surface config file.		
Comment			

MT.4

Test ID	MT.4	Date	18.05.2016
Test Number	1	Result	**PASS**
Tester	Thomas Hansen		
Specific Results	Framerate is smooth when looking around in the virtual world. There are no corruptions or artifacts in the video in the surfaces. Playback is identical to live view.		
Comment			

MT.5

Test ID	MT.5	Date	18.05.2016
Test Number	1	Result	**FAIL**
Tester	Thomas Hansen		
Specific Results	Motion tracking is responsive at all times except when changing HUD config. The video runs smoothly from the time when a new surface appears in the virtual world. The recalibration works fine in both live view and playback		
Comment	The video feed stops when changing config file. The video and motion tracking stopped when changing HUD config, this was less than 0.5 sec and the test was run from the server.		

Test ID	MT.5	Date	18.05.2016
Test Number	2	Result	**PASS**
Tester	Thomas Hansen		
Specific Results	Motion tracking was responsive at all times. The video runs smoothly from the time when a new surface appears in the virtual world. The recalibration works fine in both live view and playback		
Comment	The lag from the failed test completely gone when test was run from Argos pc.		

Reference Tests

RT.1

Test ID	RT.1	Date	22.04.2016
Test Number	1		
Tester	Morten J. Barbala		
Specific Results	30 frames delay 33/240 → 0.125 seconds delay		
Comment	Test performed on one camera Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz		

Test ID	RT.1	Date	25.04.2016
Test Number	2		
Tester	Morten J. Barbala, Leiv Fredrik Berge		
Specific Results	Delay: 21 and 25 frames 0.0958 seconds delay		
Comment	Test performed on one camera downscaled to achieve 100fps Camera used: GoPro Hero 4 Video settings: 720p at 240 Hz		

Acceptance Tests

Internal

Acceptance test, internal	Date	10.05.2016
	Result	**PASS**
Tester	All team members	
Specific Results	• Multiple distinct layouts of live video and HUD were functional. • Live video from multiple cameras produced continuous and synchronized view. Some small seams were visible, but did not break immersion or lower system performance. • Virtual world reacted properly and smoothly when moving the VR goggles. • Video was successfully recorded during longer test drive and recording icon was visible at all times in HUD. • Video playback and playback controls functioned as intended. • Playback perfectly recreated the recorded video as if it was live. • Playback could be stopped, or virtual world views changed, without causing crashes or errors. • System exited without errors.	
Comment	Test performed by placing the Argos system in a car and using a power generator. Multiple videos and pictures were taken for use in final presentation and documentation. We discovered a hardware limitation with the Oculus Rift DK2: The sensor is not designed to be in motion. When turning the car, especially sharper turns, the view in the VR goggles will drift out of calibration and start skipping.	

Stakeholder

Acceptance test with stakeholder	Date	11.05.2016
	Result	**PASS**
Tester	All team members, stakeholder: Alexander Gosling	
Specific Results	• No changes in results from internal acceptance test • Stakeholder was impressed with the final product	
Comment	Test performed in lab at Krona	

Bibliography

[1] "Oculus Rift Specs - DK1 vs DK2 Comparison," 01. January 2016. [Online]. Available: http://riftinfo.com/oculus-rift-specs-dk1-vs-dk2-comparison. [Accessed 18. April 2016].

[2] "Mako G Documentation - Allied Vision," Allied Vision, 24 November 2015. [Online]. Available: https://cdn.alliedvision.com/fileadmin/content/documents/products/cameras/Mako/techman/Mako_TechMan_en.pdf. [Accessed 02 May 2016].

[3] "Intel® PRO/1000 PT Quad Port Server Adapter Product Brief," Intel, 2008. [Online]. Available: http://www.intel.com/content/dam/doc/product-brief/pro-1000-pt-quad-port-lp-server-adapter-brief.pdf. [Accessed 02 May 2016].

Architecture Notebook 2.0

Created by: Thomas Hansen
03.02.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

HSN

KONGSBERG

Table of Contents

1. Document Overview

The purpose of the architectural notebook is to give the reader a clear understanding of the system and software architecture of Project Argos. After reading this you should know the design of the system and software architecture. The key physical components are described. Furthermore, this document describes interfaces and interactions between hardware and software components.

Describes

- the system architecture.
- show the selection of technologies, components and physical structure of Project Argos.
- the software architecture.
- functionality of key software components.
- show in detail how key parts of the software works and interacts with hardware and other software components.
- show the physical, logical and process view of Project Argos.

1.1 Document History

Version	Changes	Date	Author
0.2	Added coding style standards	12.02.2016	Morten J. Barbala Thomas Hansen
0.3	Made miscellaneous changes	16.02.2016	Thomas Hansen
0.4	Table header colour, added more requirements, Documentation / comments	17.02.2016	Trond Egil Hammer Thomas Hansen
0.5	Review document, fix minor errors	19.02.2016	Mathias Havdal
0.6	Changed software component diagram and removed some components.	26.02.2016	Thomas Hansen
0.7	Added Switch and NIC to system abstractions and miscellaneous changes	01.03.2016	Thomas Hansen
0.8	Made comments, changed goals	08.03.2016	Ingvild Damtjernhaug Leiv Fredrik Berge
0.9	Added references, reviewed the comments	09.03.2016	Thomas Hansen
0.10	Removed implementation part of architectural mechanisms and removed controller “mechanism”. Added sequence diagrams. Restructured layout. Removed one goal	10.03.2016	Thomas Hansen, Leiv Fredrik Berge
1.0	Release check for presentation 2.	11.03.2016	Thomas Hansen
1.1	Added physical sub system, physical view rewritten	05.04.2016	Leiv Fredrik Berge
1.2	New N^2 diagram, updated logical	06.04.2016	Leiv Fredrik Berge

	view, minor changes to physical view, added flow chart to software		
1.3	Added recorder handler	07.04.2016	Leiv Fredrik Berge
1.4	Changed format on name and date in document history	10.04.2016	Trond Egil Hammer
1.5	Changed architectural mechanisms, added and removed decisions and referred them to technical documents.	10.04.2016	Thomas Hansen
1.6	Added recording solution description	12.04.2016	Morten J. Barbala Trond Egil Hammer
1.7	Added configuration system description, updated controller diagram	12.04.2016	Leiv Fredrik Berge
1.8	Added System context, architecture overview, architectural decisions. Removed architectural goals, assumptions and constraints chapter	12.04.2016	Leiv Fredrik Berge
1.9	Corrections and clarifications regarding the recorder	13.04.2016	Morten J. Barbala Trond Egil Hammer
1.10	Added architecture decision section	13.04.2016	Leiv Fredrik Berge
1.11	Added to architecture overview, architectural decisions, rearranged document	20.04.2016	Leiv Fredrik Berge
1.12	Updated sequence diagrams, added playback	21.04.2016	Leiv Fredrik Berge
1.13	Added third-party software component diagram, changed document overview, removed introduction. Made changes in system context	26.04.2016	Ingvild Damtjernhaug Leiv Fredrik Berge
1.14	Remove empty headlines, correct headlines. Rewriting, clarifications and corrections. Added references for computer parts and SATA speed	26.04.2016	Morten J. Barbala
1.15	Removed architectural mechanisms as they are described in chapter 8	04.05.2016	Thomas Hansen
1.16	Add comments on a few issues (typos in diagrams, incorrect technical explanations) and add rough outline for “Picture loader” section	10.05.2016	Mathias Havdal
1.17	Added 8.3 and small changes	11.05.2016	Thomas Hansen
1.18	Made changes in 4.1, added a 3rd header level, changes in 7.1, fixed a lot of comments	12.05.2016	Thomas Hansen
1.19	Fixed layout, headings, table of contents and paragraph spacing	13.05.2016	Morten J. Barbala
1.20	Add rough draft of flow chart for	16.05.2016	Mathias Havdal

	picture loader in section 8.2		
1.21	Improve picture loader flowchart in section 8.2 and add figure number. Start fleshing out text in sections 8.2 and 8.2.1.	17.05.2016	Mathias Havdal
1.22	Slight changes in section 8.2.1, write section 8.2.2	18.05.2016	Mathias Havdal
1.23	Deleted “Playback” section in chapter 8, as it has been replaced by section 8.2.2. Start reviewing document, highlighting and correcting issues.	18.05.2016	Mathias Havdal
1.24	Highlight more issues.	19.05.2016	Mathias Havdal
1.25	Fix lots of issues. Rewriting and removing text.	19.05.2016	Mathias Havdal
1.26	Removed redundant information in chapter 4. Improved section 6.2.	19.05.2016	Mathias Havdal Leiv Fredrik Berge
1.27	Removed redundant section 6.3 and moved N^2 diagram to section 6.2.	19.05.2016	Mathias Havdal Leiv Fredrik Berge
1.28	Fixed 4.3, added references. Split the document with TinyArgos Technical Solutions. Fixes in Process view.	20.05.2016	Leiv Fredrik Berge
2.0	Final review	21.05.2016	Morten J. Barbala Thomas Hansen Leiv Fredrik Berge Ingvild Damtjernhaug

1.2 Referenced Documents

Title	Document	Version
Requirements Document	doc-1213_requirements_2_0.docx	2.0
Glossary	doc-1113_glossary_2_0.docx	2.0
Technical Document: Lenses	doc-21321_lenses_2_0	2.0
Technical Document: Network Solutions	doc-21321_network_solutions_2_0.docx	2.0
Technical Document: VR Goggles	doc-21321_VR_goggles_3_0.docx	3.0
Technical Document: Architecture style	doc-21321_Architecture_pattern_1_0.docx	1.0
Technical Document: Motion Sickness	doc-21321_motion_sickness_1_0.docx	1.0
Technical Document: Graphics Library	doc-21321_Graphics_library_technical_document_1_0.docx	1.0
Technical Document:	doc-21321_GigE_Vision_SDK_1_0.docx	1.0

GigE Vision SDK		
TinyArgos Technical Solutions	doc-333_tinyargos_technical_solutions_1_0.docx	1.0
T.16 Test log	doc-44_T_16_1.docx	N/A

1.3 List of Figures

1.4 List of Tables

2. System Context

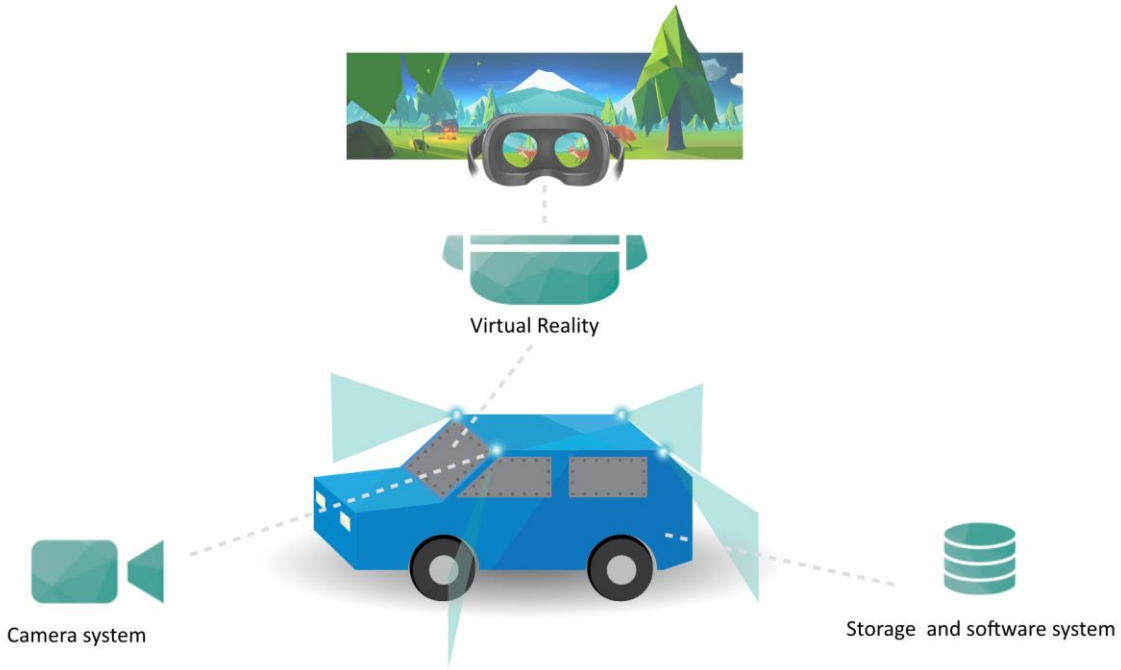

Figure 1: High level system architecture

Project Argos is a system which enables a user to drive a vehicle with a virtual reality (VR) headset. A live video feed from the outside is shown inside the VR goggles. This lets the driver observe surroundings through VR goggles instead of looking through the wind shields. The system consists of several parts. It has a camera rig to record the video, a network to transfer the data, a computer as a platform for the TinyArgos software and a VR headset. The main focus for the bachelor project of 2016 is to develop key functionalities in the software. The TinyArgos software and the hardware should then be integrated to create a working tech demo.

We consider all the components mentioned above as parts of the Argos system. We are building a tech demo, which will provide the project owner with a platform to test the viability of the concept. Power supply, vehicle camera mount, size of the complete system etc. are future challenges not considered to be part of the bachelor project.

As for the scope of this bachelor project, we do not interact with outside software. However, this is expected to be a key functionality at some later point in the development process and the architecture must therefore allow for interfaces towards other applications and systems. This includes, but is not limited to: global positioning system (GPS), map software, surveillance software or other Kongsberg systems. It is also important for us that the system building blocks are “off the shelf” parts. We want Project Argos to be modular. Individual components should be exchangeable with better hardware as it becomes available.

3. Software Architecture and System Architecture Overview

The architecture is divided into two main parts: The hardware components and the software. The hardware consists of the camera rig with multiple cameras and lenses, connected to a network for an interface towards the software, and other key hardware components like the VR-headset and the computer running the software. The software architecture describes the building blocks through: (i) the nature of the architectural components i.e. the nature of their computation, (ii) the way they interact with other components when composing a system and(iii) constraints on the way this composition is done. [1]
Most of the decisions in the architecture are made with the end goal of reducing latency and increasing speed and performance. The latency from the point an image is captured by the camera, till it is displayed to the user in the headset (glass to glass latency) is critical for the experience to be pleasant. This means we want to reduce everything that slows the process down, both in hardware and software. As of now the software and storage system is contained in a single computer, but in the future this could be split across multiple devices to ease the computational load on the virtual reality computer. The other latency we are concerned about is in the VR headset itself, meaning the time from the user moved his or her head until the motion is reproduced to match in the virtual world.

The software has the same objective as the rest of the architecture; reduce latency. This means we want to utilize parallelism and all the available computational power of the computer, both central processing unit (CPU) and graphical processing unit (GPU). We also need a high level of modularity to easily support new camera technology and virtual reality hardware. VR headsets are still an immature technology and we expect great leaps in comfort, performance and features over the coming years. Our software must therefore be versatile enough to be upgradable as Project Argos will continue to be developed by other teams after the bachelor project. In short, the code must be easy to understand and improve. We accomplish this by using common and current libraries and tools, and splitting the code into smaller modules.

4. System Architecture and Software Architecture Decisions

This section will cover the key decisions that we have made with impact on the system and software architecture. This provides our reasoning for choosing the technology and standards we did, and in some cases what the alternatives are and why they were not chosen. Some of the key components are covered in more detail in the technical documents. The focus for most of our decisions is to lower the glass to glass latency of the system and to increase usability and comfort.

4.1 Software Structure

The structure of the software is very important to ensure high data throughput. We want to create as few obstacles for the video data as possible from point of capture till it is displayed. We have selected aspects from two main software architecture design styles: Pipe filter as the main data process structure, and component based architecture (CBA) style to separate the different functions of the software into easily maintainable modules. We also implemented Modell-View-Controller (MVC) pattern to separate the data from control functions and user interface in this structure

- Pipe filter style
  - See the technical document on software architecture style.
- Component based architecture style
  - CBA is a way to describe software as a set of components. These components communicate through loosely coupled interfaces. [2]
- MVC pattern
  - Model-View-Controller pattern is a common software pattern to separate data from the control functions and the interaction with the user. The user only interacts with the view and the view only with the controller and the data only with the controller. This allows for a very clean and sensible allocation of functionality that protects the data. [3]

4.2 Identification of Third-Party Software

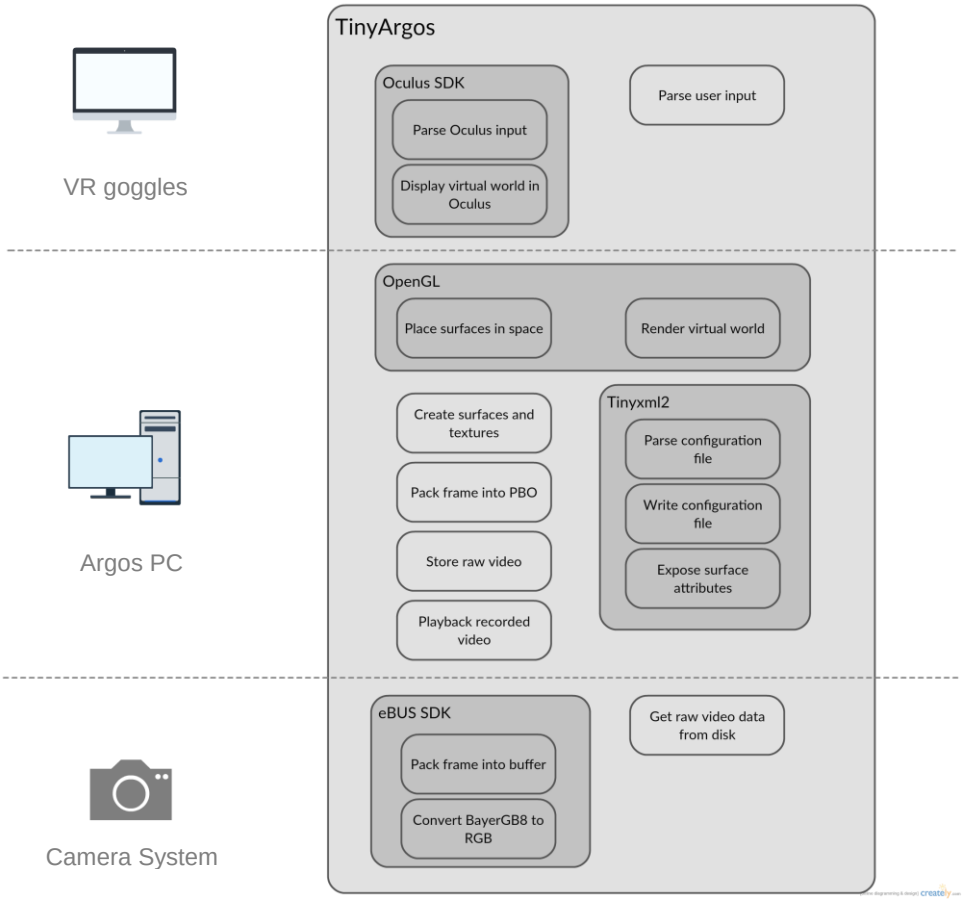

Figure 2: TinyArgos software composition with third-party software

We need a number of software development kits (SDK) and libraries to improve performance, ease of use and development. Some of these are given from the technology choice, but there are still some choices to be made.

- Oculus SDK 0.8.0.0
  - There is no common standard for VR headsets yet, so we are forced to use the SDK provided by Oculus. 0.8 was the latest SDK available when the bachelor project started. Version 1.3 has now been released, but it does not have official support for the Oculus Rift DK2 headset we are using. To add to this, 0.8.0.0 is a beta version, while 1.3 is a release version. This means that there could be several breaking changes, which in turn increases the effort required to migrate to the newer version. [4]
- Pleora eBUS SDK 4.1.5
  - The choice of cameras dictates the transfer standard, and limits the potential SDK choices. Our cameras follow the GigE Vision standard, so we have selected the eBUS SDK made by Pleora. See the GigE Vision technical document for more details.
- TinyXML2
  - Our configuration files are written in XML. XML is a simple way of storing data, and has the advantage of being human readable and editable in any basic text editor. Instead of writing our own XML parser for the TinyArgos software, we have decided to use TinyXML2. TinyXML2 is lightweight and stable, and saves us significant effort in implementing a configuration system [5].
- OpenGL
  - To create a virtual world to display in the VR headset we needed a graphics API for rendering. OpenGL works with the Oculus SDK. For more details, see the technical document on graphics library.

4.3 Choice of Technology

The technology has a huge impact on the physical architecture and the software. We want the system to be as modular as possible so individual parts of the system can be replaced with relatively little effort. This means in particular that we want to choose technology that complies with established standards. As VR still is in its infancy, standards in regards to this aren't stable yet, but in the other aspects of the system good standards are available.

We inherited the project at a point in time were much of the choice of technology had already been taken. The major impacting components like the cameras, VR goggles, development platform and programming language had been decided. In our technical documentation we have reviewed some of the choices, and made our recommendations for upgrades down the line. We have made changes in upgrading the Visual Studio integrated development environment (IDE) from 2013 to 2015. We did consider upgrading to the latest Oculus Rift CV1, but availability and time constraints stopped us from pursuing this further.

5. Architecturally Significant Requirements

These are the key requirements that have a direct influence on the architecture. The most important requirements are A.NF.8 and B.NF.3, which refers to the glass to glass latency. For the entire list of requirement see the requirements document.

Requirement number	Description
B.NF.1	Glass to glass latency should be less than 20 ms
B.NF.3	The system should be up and running in less than 20 seconds
B.NF.4	It should take less than 5 clicks to get a video feed in the VR goggles
A.F.5	The cameras must be able to send live video to system
A.F.6	The system must be able to capture and handle video from all four cameras
A.F.7	The system must be able merge video from all cameras to one continuous image
A.F.8	The system must be able to display the continuous image in VR goggles
A.NF.8	Glass to glass latency must be less than 75 ms
C.NF.1	It should be possible to learn key functionality in 10 minutes

Table 1: Key requirements

6. Physical View

The physical view is meant to give the reader a clear understanding of what physical parts the system is composed of and the environment it will operate in. The system is composed of a camera system to capture live video around the vehicle, a powerful, high-end consumer grade computer and an Oculus Rift VR headset to display the images. The components are connected in a 1 Gbit Ethernet local area network, with power over ethernet for the cameras. The cameras are machine vision and surveillance cameras with C-mount for the lenses. The rest of the physical components are consumer grade hardware, to keep cost down and provide us with the performance needed for the system. This will also allow us to replace components easily when new and faster hardware becomes available. As consumer electronics keep improving, we believe it will be sufficient to meet our environmental requirements, also moving forward towards a production system.

The main purpose of the physical system architecture is to enable us to capture live video and display it in real time in VR goggles inside the car. The user should be able to drive only with the help of cameras and VR. Latency is the biggest enemy for this to be achieved, so the system architecture must provide the most efficient route from video capture to display in VR. It can be a challenge to see where the physical view ends and the logical view begins. This is because our system exists in a cyber-physical space. As an example the Oculus Rift is a physical component, but at the same time it acts as both an input device and an output device, and incorporates quite a bit of logic through the Oculus SDK and API. In addition, the cameras are

physical components with a major software part that is mix of physical and logical elements.

6.1 System Models

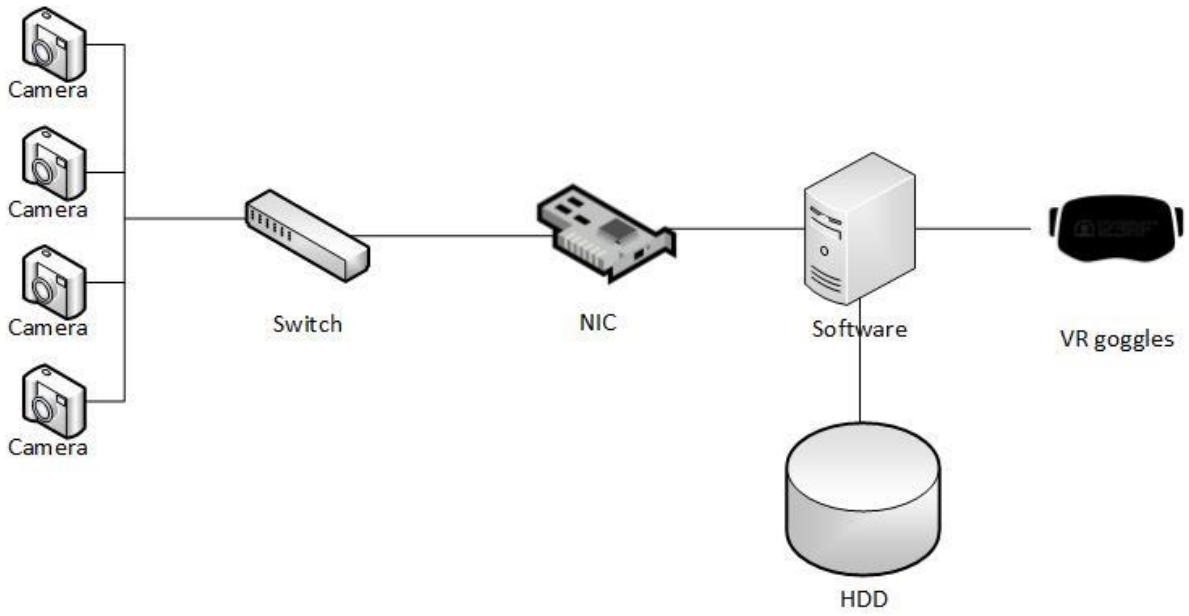

Figure 3: System component diagram

The system component diagram shows an overview of the major physical components and the interfaces that tie them together. We have made efforts to use hardware that supports common standards wherever possible. This way we can upgrade and replace components with minimal changes elsewhere in the system, reducing the effort required to take advantage of newer and better hardware.

The cameras and the graphics card are components that have a key impact on the systems performance, and are areas where technology is constantly improving. In the future, this could lead to increased requirements for network and storage, but hardware with significantly higher performance than our current setup already exists in these fields. [6]

6.2 Key Physical Subsystems

The N^2 diagram shows the interaction between the subsystems in Project Argos. The dark blue shows the hardware subsystems, the green is the TinyArgos software and the orange is the cyber physical VR goggle system. The key physical subsystems are the cameras, network switch, Argos PC and VR goggles. They all share the same goals of achieving the lowest possible latency and providing the user with a pleasant VR experience. The following sections describe and justify the hardware we selected for each of the key physical subsystems.

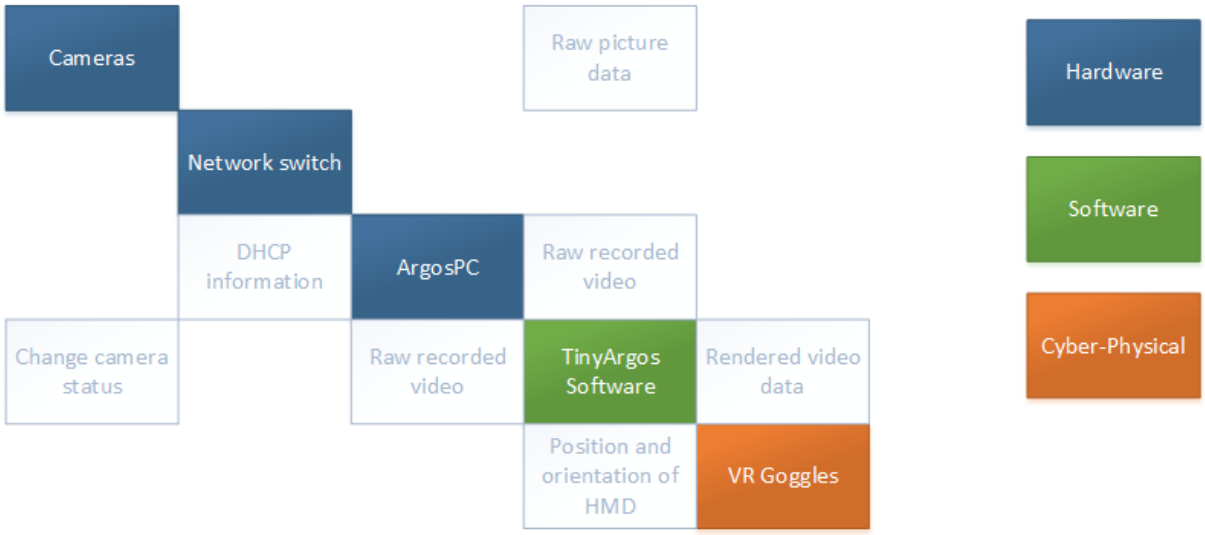

Figure 4: N² diagram for the system

6.2.1 Camera System

- 4x Allied Vision Mako G223-C GigE Vision cameras with power over ethernet (PoE) support
- 1x Fujinon FE185C086HA-1 fish eye lens
- 3x Kowa LM6JC wide angle lenses

The cameras in our system are high-end cameras used in surveillance and machine vision. They provide a relatively high resolution for this type of camera at 2048 x 1088, and a high frame rate of 49.5 FPS.

For Project Argos it is a great advantage that the cameras can draw power from the ethernet cable, rather than requiring a battery pack or standalone power. This means we only need one power source and each camera only needs a single cable. The transfer rate of up to 128 Mb/s theoretically maxes out the 1 Gbit connection to the switch, and is needed to transfer the sheer amount of uncompressed data.

The greatest advantage of the Mako G223-C is that it does not buffer images before sending them. Most consumer grade cameras, like for example GoPro, buffer at least a couple of frames to ensure smooth playback. [7] We value low latency over smoothness, so any extra buffering in the camera is undesirable.

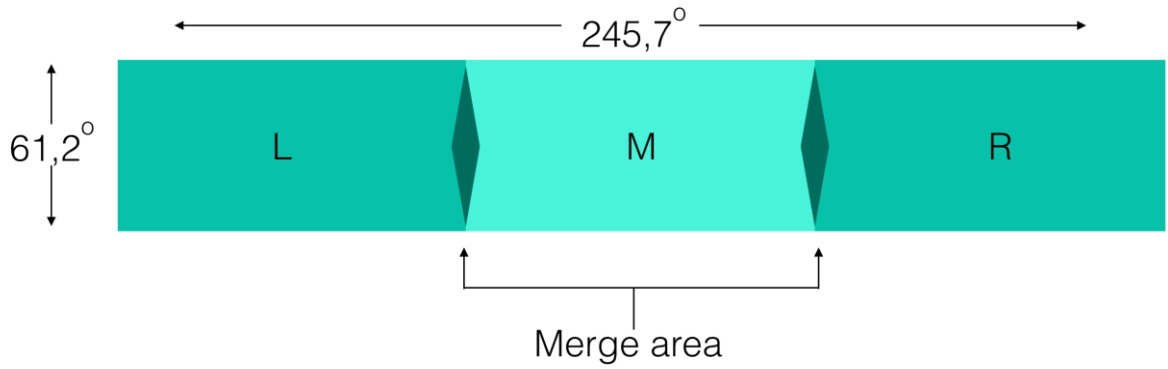

Figure 5: Front view diagram

We will use three front facing cameras with wide angle lenses. The video feeds from these cameras will form a continuous image, covering up up to 245 degrees of the horizontal field of view. We will also have a rear facing camera with a super wide angle fish eye lens. This camera will act as a rear view mirror, and will not be joined with the three front facing cameras. We mean this creates a more efficient and comfortable user experience. This also avoids a potential issue with the Oculus, where the tracker only tracks the front of the Oculus. So if the user turns his head to see behind, it will not be able to track the HMD.

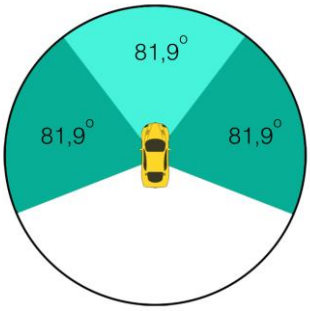

Figure 6: Field of view of front facing cameras

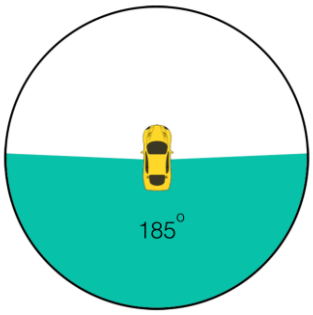

Figure 7: Field of view of rear facing camera

Figure 8: Total field of view of camera system

With this solution our horizontal field of view covers up to 360 degrees. We believe this is the best compromise between field of view and computational time to stitch the video feeds together. More details can be found in the technical document for lenses.

6.2.2 Network subsystem

- 24 port PoE Gigabit Network Switch
- Intel Pro/1000 PT Quad Port LP
- Argos PC with DHCP server

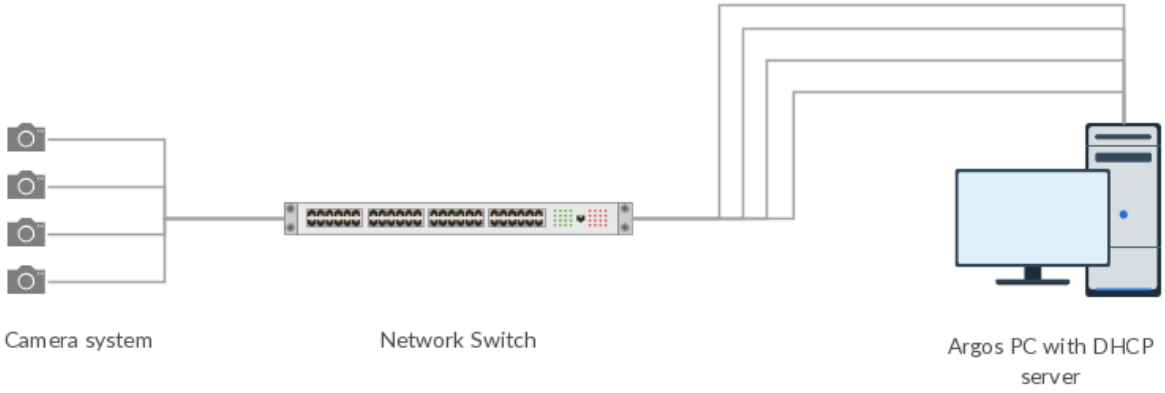

Figure 9: Network diagram

The backbone of the network is a switch with 24x1 Gbit ports and PoE support. The Argos PC is equipped with a 4x1 Gbit network interface card (NIC). We have configured the NIC to behave as a single logical port, with automatic load balancing across the four physical ports. For more details see the network solutions technical document.

We have configured the Argos PC as a dynamic host configuration protocol (DHCP) server, assigning IP addresses to the cameras. This allows us to assign each camera a static IP address based on its media access control (MAC) address. Having static IP addresses makes it easier to be sure that the physical placement of the camera matches the placement of the video stream in the virtual world.

6.2.3 VR System

- Argos PC
  - Intel i7 5930K
  - 32 GB RAM
  - GeForce GTX Titan X
  - 240 GB SSD for OS
  - 720 GB SSD Array in RAID 0 for storage
- Oculus Rift DK2

Powering the VR system is a high end consumer PC. It uses the Intel i7-5930K processor, which is one of the most powerful consumer grade processors according Futuremark's CPU benchmark [8]. Processing multiple high quality video streams in parallel and presenting them in a virtual world can be demanding. For that reason, it is important to have a processor that has a high clock speed and IPC (instructions per cycle) as well as a sufficient number of cores.

If we can achieve a sufficient degree of parallelism, we might see better performance with a server grade processor. Server grade processors like Intel's Xeon series typically have more cores than consumer grade processors, at the cost of having a lower maximum clock speed. The cost of a Xeon processor is also typically higher, and often requires a special motherboard and RAM. In the future, a deeper analysis

of the performance scaling of the TinyArgos software could be done to determine if this upgrade is appropriate.

The GPU is important for all high performance 3D rendering. This is especially true for VR applications, because the virtual world has to be rendered twice every frame. This is because humans have two eyes, and each eye must see the virtual world from two different perspectives. For the best possible experience, the framerate also needs to be very high. See the motion sickness document for more details.

The Argos PC is equipped with the Nvidia GeForce Titan X, one of the most powerful graphics cards available as per April 2016 [9]. It features a massive 3072 Cuda cores, 192 texture units, a core clock of 1000 MHz and 12GB of GDDR5 memory [10]. It would be possible to add another Titan X and utilize SLI, Nvidia's multi GPU technology, but getting good performance scaling typically requires some optimization and increases complexity. [11]

Another key part of the Argos PC is the storage array. Each camera can send up to 128 MB/s. With the four cameras we are using, the computer will receive up 512 MB of data each second. Because processing and displaying the video streams is already so demanding, we will not be able to compress the video before storing it. This makes it obvious that we will need a lot of storage space to be able to record any significant amount of video.

To be able to write all the received footage to disk, we will need a storage medium with write speeds of at least 512 MB/s. The third generation SATA 6 Gb/s interface has a maximum write speed of 600 MB/s [12], but the actual write speed is usually limited by the physical storage medium and not the interface itself. To ensure sufficient write speed, we have used three 240 GB SSDs in a striped volume (RAID 0). In a striped volume, files are split into multiple parts, and the parts are distributed evenly between the drives. This gives a linear increase in storage space and can theoretically also give a linear increase in read/write performance. Our testing has shown write speeds of 800-1200 MB/s, depending on the block size of the files being written to the drive; see the test log for T.16.

The downside with RAID 0 is that if one drive becomes corrupted, the whole volume is also corrupted. In a production system, it might be worth using one of the three drives as a parity drive (RAID 10). This will give some of the performance benefit of RAID 0, with the added redundancy of RAID 1. However, we will only be using this computer for a tech demo, so the data we store is not that important.

For long term storage, it might be a good idea to copy the stored data to mechanical disks. A mechanical disk is slower, but often gives more capacity for the same cost as an SSD. [13] Note that if the recorded video streams are to be played, they will have to be copied back to a faster storage medium. Otherwise real-time playback will not be possible because of disk read speed starvation.

The VR system of choice is the Oculus Rift DK2. This was the only VR headset on the market when we started our bachelor project. See the VR goggles technical document for more information.

7. Logical View

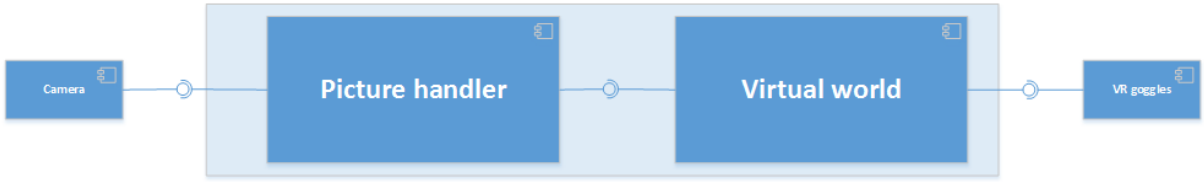

Figure 10: Software architecture overview of the pipeline from glass to glass

The logical view refers to the static software components that make up the TinyArgos application. The diagram above shows the allocation of functionality and the interface between the software modules. This describes the relationships and the static nature of the software architecture. To see the dynamic aspects of our software solution see the TinyArgos technical solutions documents. TinyArgos is the main application for gathering video data from the GigE vision cameras, creating the virtual world and displaying the virtual world in the Oculus Rift. As mentioned in the physical view, the line between hardware and software is a bit more blurred than what it used to be, with systems having both software and hardware component.

The logical architecture must be as streamlined as possible to make sure we have the data throughput we need of the system. We also need the software to respond to the user's actions and inputs.

7.1 Architectural Framework

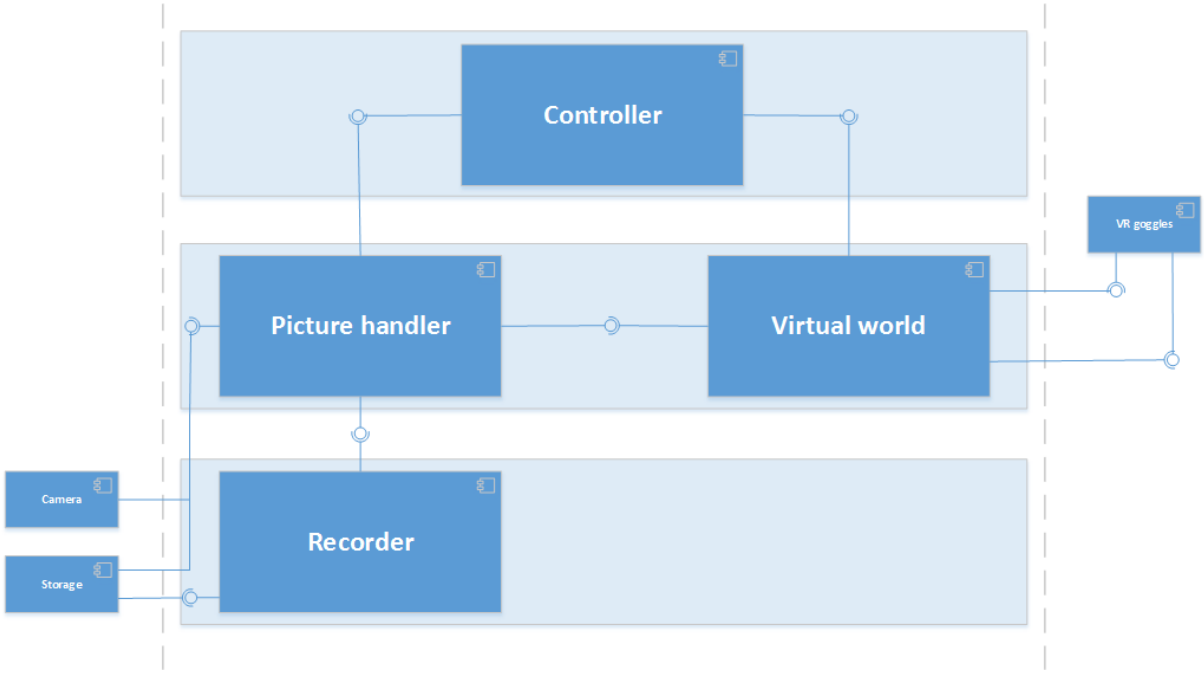

Figure 11: Our software architecture, pipe-filter and components based architecture and MVC pattern

Because the main feature of this system is to stream video to the VR goggles some implementation of the "Pipe-Filter style" is the architectural style we believe to be the best choice. The style describes a way of constructing a route by creating a chain of

filters where the output of one filter is fed into the input of another filter in the pipeline. It is important to maintain as much continuous dataflow as possible so that the pipe filter style does not turn into a batch processing style. We need a bit more control that what the pipe and filter style offers, so we have created our own component based style. See the architectural style document for more details.

The pipeline starts at a picture handler module that receives the picture data from the cameras and packs this data into pixel buffer objects (PBO) that it sends to the virtual world and if recording is activated it copies the raw picture data to the recorder module. The recorder handler includes the environment for the recording to occur. We set up the recording environment with four frame buffer recorders, one for each of the cameras in the system. This way we can store raw, uncompressed data from all of the cameras, without stealing much of the resources from the rest of the system. This does not change the data, so when we play back the video streams we can reuse much of the code from live video.

The virtual world module incorporates the live video stream into the Oculus to create the virtual reality experience for the user. This includes displaying the right video stream at the right place in the virtual world. The virtual world also has heads up display (HUD) features. This allows us to display text and other data on top of the video from the cameras, and fill the areas outside of the view of video with useful information like a compass and a map. This can also be extended to gather data from external system, and add markers to the HUD. To create the virtual world we use OpenGL and the Oculus SDK. It's also a key functionality of the virtual world to gather motion input from the Oculus, so we can move the image inside the virtual world to match the movement of the users head. This is an integral part of creating the immersion in the virtual world and essential for the experience to be realistic.

8. Process View

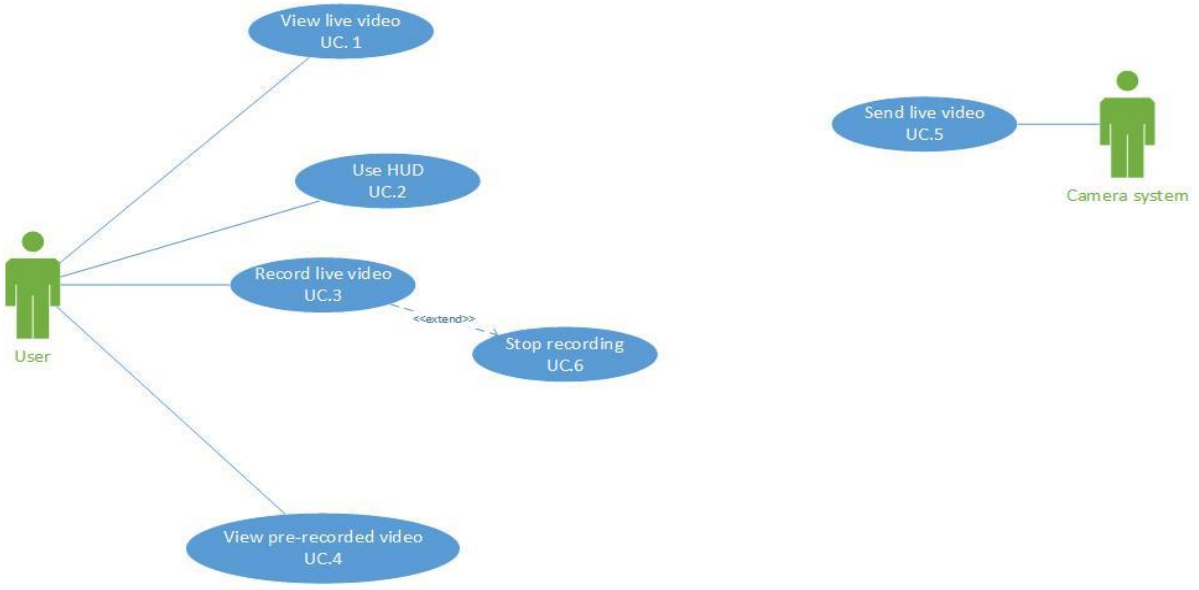

Figure 12: Use case diagram

The process view shows the dynamic view of the software architecture. It shows the interfaces and interaction between the software components in the sequence diagrams that corresponds to each use case.
We have used the process view and sequence diagrams help us find the objects and classes in the software. It is also important to show the interfaces and connections inside the software, and who interacts with whom of the software architectural blocks. The sequence diagrams expand on the use case diagrams with much more detail in how the program solves each use case. The code implementation can in some cases be slightly different than the diagrams, since the diagrams were created before the code as a help and guide for development. Read more on the implementation of the diagrams in to code in the TinyArgos Technical Solutions document.

8.1 View live video sequence diagram (UC.1)

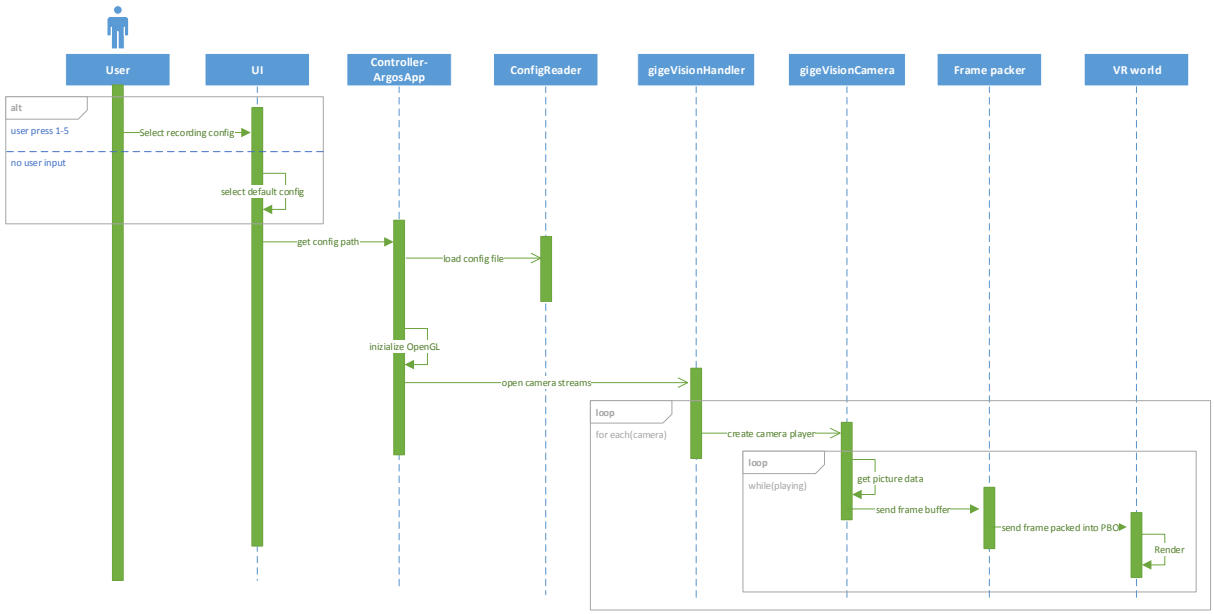

Figure 13: Sequence diagram for UC_1 "View live video"

When the user starts the program it will by default to the config.xml file in resources. If the user wants a different set up, it's possible to invoke that with a press of one of the buttons from 1 through 5 on the keyboard. The default or the chosen config file is parsed by the config reader, and the controller sets up the environment accordingly. Then the controller connects to the cameras as described in the config by creating a handler for the cameras. The handler creates a single camera object for each of the streams. The cameras gets data from the cameras, packs it into buffers and is sent to the virtual world to be displayed.

8.2 Choose and use HUD sequence diagram (UC.2)

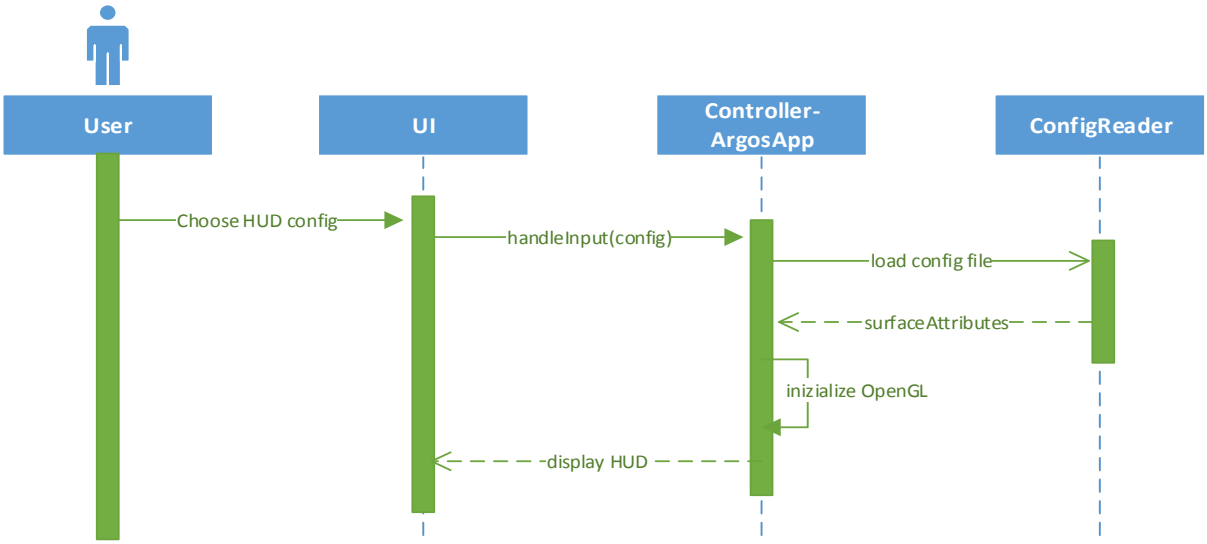

Figure 14: Sequence diagram for UC_2 "Use HUD"

The HUD is split from the normal config file, so any HUD can be used with any surface configuration. This means we reduce the number of configuration files, while maintaining flexibility for the user. The HUD configs are chosen by the user through the keyboard and captured by user interface. The controller sends the path of the config file to the config reader that parses the HUD config file and returns the surface attributes. Then the controller reinitializes OpenGL to set up the new surfaces in the virtual world, and is displayed to the user in the Oculus Rift.

8.3 Record and stop video sequence diagram (UC.3 / UC.6)

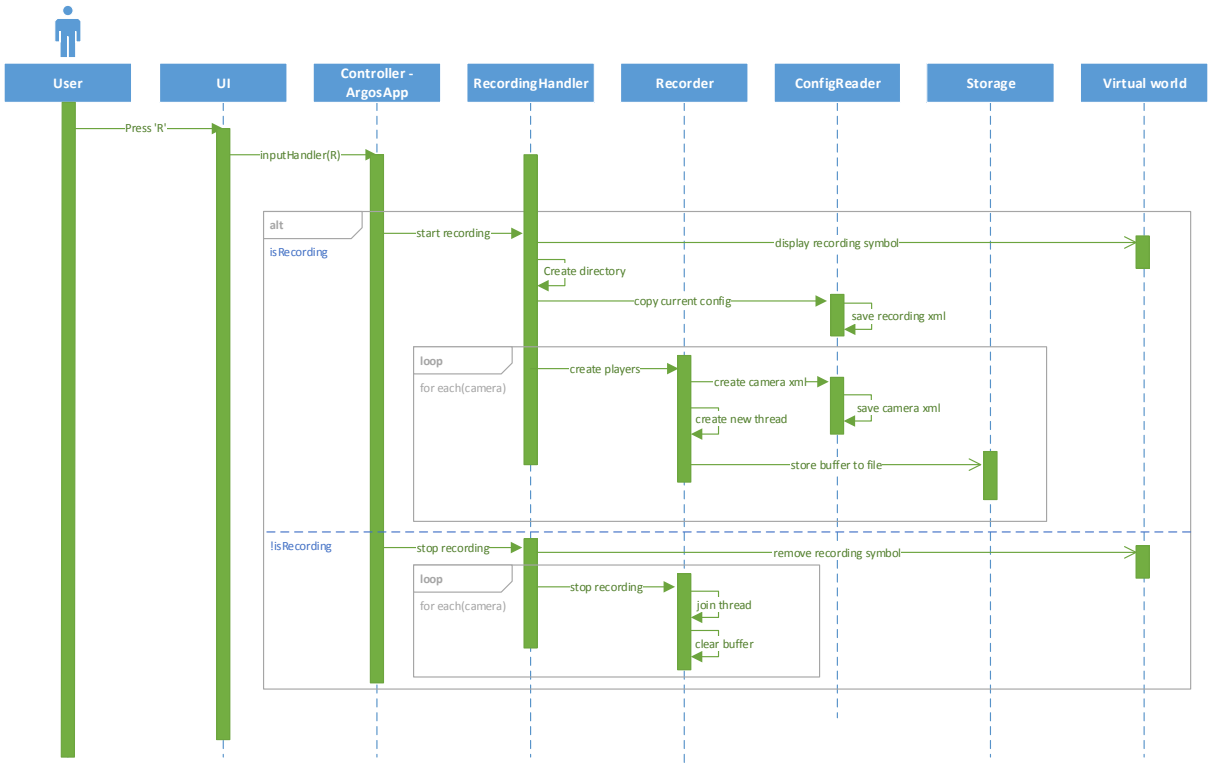

Figure 15: Sequence diagram for UC_3 "Record live video" and UC_6 "Stop recording"

The recording function is invoked by the user hitting R. Then the controller will check if a recording is active or not, and either start or stop the recording. If a recording is active the recording handler will let each recorder know to stop, where by each recorder will clear the buffers and join the thread back to the parent. Also the recording symbol in the HUD will be removed. Otherwise, if the user wants to start a recording, the "rec" symbol will be displayed. A new directory in the recording folder will be created with a unique name. The current configuration will be copied, slightly altered to reflect it isn't live and stored in the new recording folder. Then the recording handler will create a recorder for each camera, with an XML file with the camera settings and a thread to store the buffers to disk.

8.4 View pre-recorded video sequence diagram (UC.4)

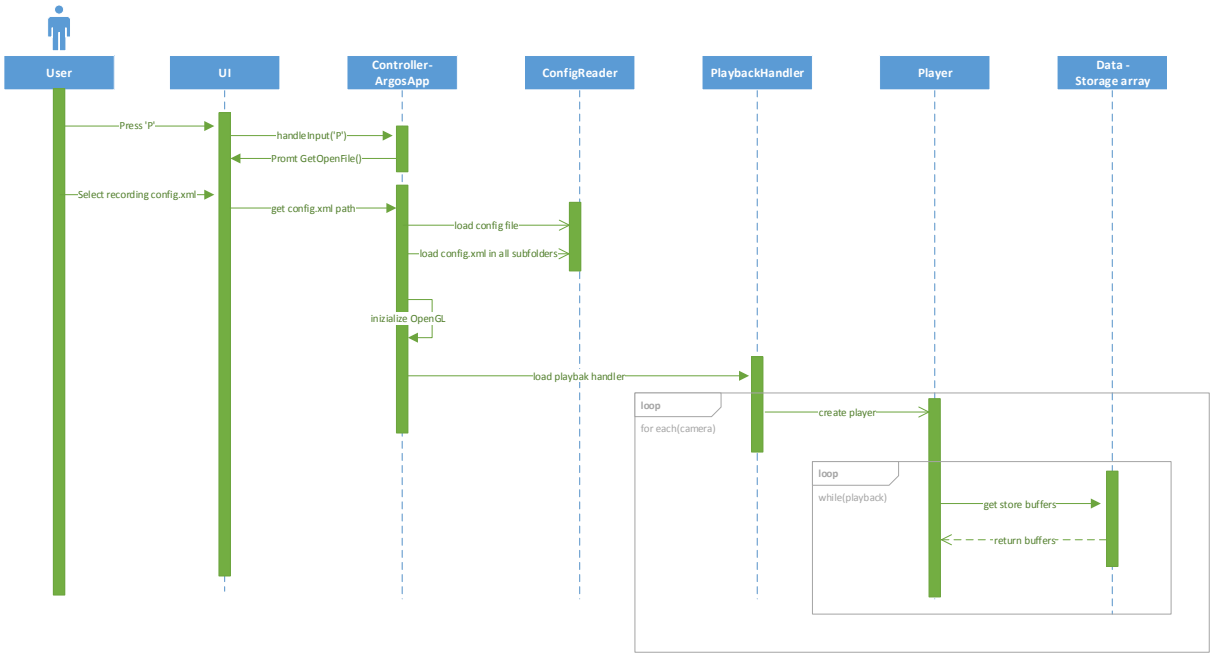

Figure 16: Sequence diagram for UC_4 "View pre-recorded video"

To choose a pre-recorded video to play in the TinyArgos software, we have a simple but effective keyboard interface. If the user hits the “p” button the UI triggers the controller to invoke the “getOpenFileName” function. This will show an open file dialog, and let the user choose the config.xml file in the correct recording folder. The controller will send the file path to be parsed, it will also search the folder for any sub folders, and all the config.xml files in the sub folder is parsed as well to set up the environment with the settings the cameras had at the time of capture. The OpenGL scene will also be set up with the same configuration that was stored when the video was recorded. Then the controller invokes the playback handler, this set up one player stream for each of the cameras that recorded, and it’s the players that gathers the buffers from the storage, and makes it available in the virtual world for the user.

Bibliography

[1] R. Juric, J. Kuljis and R. Paul, “Software architecutre style for interaoperable databases,” in *International Conference on Information Technology Interfaces*, Cavtat, Croatia, 2004.

[2] M. C. Oussalah, Software Architecture 1, Wiley, 2014.

[3] Google, “MVC Architecture,” Google Chrome, [Online]. Available: https://developer.chrome.com/apps/app_frameworks. [Accessed 20 May 2016].

[4] Oculus, “Documentation,” Oculus, [Online]. Available: https://developer.oculus.com/documentation/intro-vr/latest. [Accessed 20 May 2016].

[5] “TinyXML2,” 17 February 2016. [Online]. Available: https://github.com/leethomason/tinyxml2. [Accessed 27 April 2016].

[6] Cisco, “10 Gigabit Ethernet Technologies,” Cisco, [Online]. Available: http://www.cisco.com/c/en/us/solutions/data-center-virtualization/10-gigabit-ethernet-technologies/index.html. [Accessed 17 March 2016].

[7] GoPro, “Hero4 HDMI Output Informaiton,” GoPro, [Online]. Available: http://gopro.com/support/articles/hero4-hdmi-output-information. [Accessed 20 May 2016].

[8] “Best Processors April - 2016,” 08 April 2016. [Online]. Available: http://www.futuremark.com/hardware/cpu. [Accessed 27 April 2016].

[9] “Best Graphics Cards April - 2016,” 08 April 2016. [Online]. Available: http://www.futuremark.com/hardware/gpu. [Accessed 27 April 2016].

[10] “GeForce GTX TITAN X | Specifications | GeForce,” [Online]. Available: http://www.geforce.com/hardware/desktop-gpus/geforce-titan-x/specifications. [Accessed 27 April 2016].

[11] U. Pirzada, “WCCF Tech,” September 2015. [Online]. Available: http://wccftech.com/multi-gpu-nvidia-sli-and-crossfire-performance-value-comparison/. [Accessed 20 May 2016].

[12] “Brochures | SATA-IO,” May 2009. [Online]. Available: https://sata-io.org/sites/default/files/images/SATA-IO-English-Brochure-May-2009.pdf. [Accessed 27 April 2016].

[13] BackBlaze, “Hard Disk Drive Versus Solid State Drive: What's the Diff?,” Backblaze, 8 March 2016. [Online]. Available: https://www.backblaze.com/blog/hdd-versus-ssd-whats-the-diff/. [Accessed 20 May 2016].

[14] “Tutorialspoint,” 2015. [Online]. Available: http://www.tutorialspoint/software_architecture_design/software_architecture_design_tutorial.pdf. [Accessed 09 March 2016].

[15] “WikiBooks,” 2016. [Online]. Available: https://en.wikibooks.org/wiki/C%2B%2B_Programming/Programming_Languages/C%2B%2B/Code/Style_Conventions#References. [Accessed 09 March 2016].

[16] Microsoft Corp., “Microsoft Application Architecture Guide,” October 2009. [Online]. Available: http://msdn.microsoft.com/en-us/libary/ee658117.aspx.

[Accessed 21 April 2016].

[17] Carnegie Mellon University, “Software Engineering Institue - Software Architecture,” 2016. [Online]. Available: http://sei.cmu.edu/architecture/tools/. [Accessed 21 April 2016].

[18] G. Fairbanks, Just Enough Software Architecture, Marshall & Brainerd, 2010.

[19] P. Clemments, F. Backmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord and J. Stafford, Documenting Software Architectures: Views and Beyond, Addison-Wesley Professional, 2012.

[20] L. Bass, P. Clements and R. Kazman, Software Architecture in Practice, Addison-Wesley Professional, 2012.

[21] M. Engelsberger and T. Greiner, “Software Architecture for Cyber-Physical Control Systems with Flexible Application of the Software-as-a-Service and On-Premises Model,” IEEE, Pforzheim, 2015.

[22] P. Clements, R. Kazman and M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley Professional, 2001.

TinyArgos Technical Solutions 1.0

Created by: Mathias Havdal
20.05.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

Table of Contents

1. Document Overview

The purpose of the TinyArgos technical solution document is to give the reader an understanding of the key technical aspects of the TinyArgos software. After reading this document you should have a general idea about the inner workings of the software and the justifications for some of the technical solutions that have been used.

Describes

- key software components.
- technical details of software components.
- video stitching technique.
- coding style.

1.1 Document History

Version	Change	Date	Created by
0.1	Create document and move “Process View” and “Coding Style” sections from architecture notebook	20.05.2016	Mathias Havdal
0.2	Move sequence diagrams back to architecture notebook. Rename “Process View” section to “Software Components” and rewrite section intro. Rewrite section about controller component, change picture loader to picture handler.	20.05.2016	Mathias Havdal
0.3	Start rewriting “Virtual World” section in “Software Components” chapter	21.05.2016	Mathias Havdal
0.4	Finish rewriting “Virtual World” section. Start work on “Video Stitching” chapter.	21.05.2016	Mathias Havdal
0.5	Write subsections in “Video Stitching” chapter.	22.05.2016	Mathias Havdal
0.6	Fix section numbers, remove obsolete diagrams. Rewrite “Recorder” section. Write document overview. Add GigE Vision and lens technical documents to referenced documents. Fix bibliography.	22.05.2016	Mathias Havdal
0.7	Fix layout and typos.	22.05.2016	Morten J. Barbala
1.0	Final review	22.05.2016	Morten J. Barbala, Thomas Hansen Ingvild Damtjernhaug

1.2 Referenced Documents

Title	Document	Version
Glossary	doc-1113_glossary_2_0.docx	2.0
GigE Vision SDK	doc-21321_GigE_Vision_SDK_1_0.docx	1.0
Lenses	doc-21321_lenses_2_0.docx	2.0

1.3 List of Figures

2. Software Components

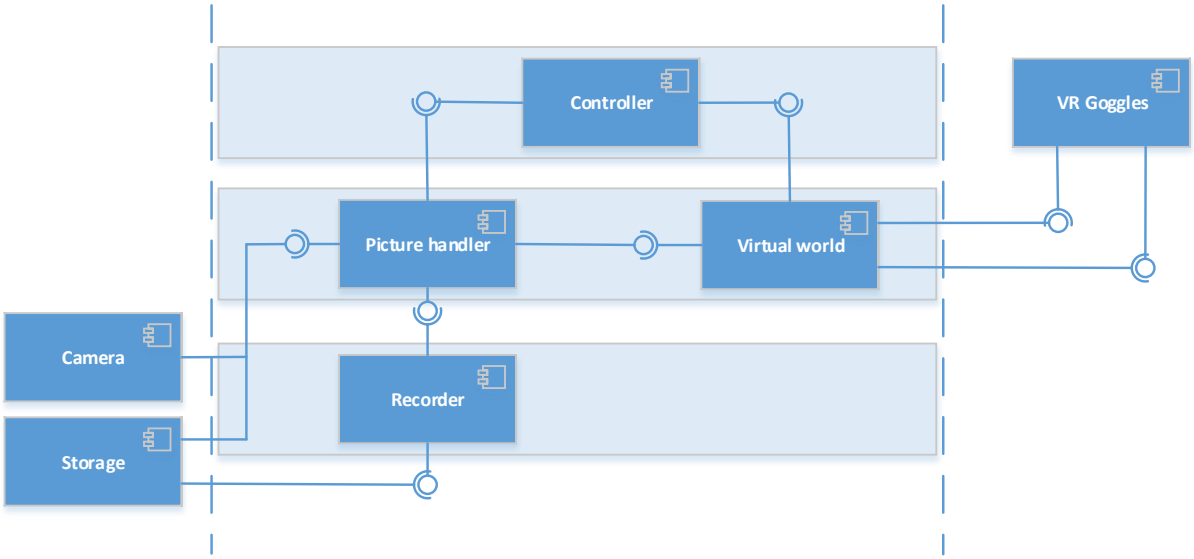

Figure 1: TinyArgos component diagram

This section gives an overview of the main components that make up the TinyArgos software. Full technical documentation, i.e. source code, is not public information and only available on DVDs supplied for assessment.

2.1 Controller

The controller in the TinyArgos software is the ArgosApp class. ArgosApp inherits from the RiftApp and RiftManager classes, which contain functionality specific to the Oculus Rift DK2 headset. The controller in our software has the task of managing the picture handler, recorder and virtual world based on inputs from the user.

One of the most common tasks the controller has to perform is switching config files. The config system is divided into two parts: One for video and one for the heads-up display (HUD). This gives our software the ability to display many different combinations of video and HUD while keeping the number of config files to a minimum.

When the user sends input telling the software to load a new config file, the controller will use either the ConfigReader or ConfigReaderHUD class to parse the file. Using the data from the config file, the controller will create new surfaces to be placed in the virtual world. Before the new surfaces are added, surfaces from the previous config are removed.

If the new config was for video, there are a few more steps that need to be carried out. We have two possible video sources: Recorded video from disk and live video from cameras. The controller will configure the picture handler according to user input. For live video, both the picture handler and recorder must be configured. For playback, however, the picture handler is configured and the HUD config associated with the recorded video is loaded.

When live video is running, the controller is responsible for starting and stopping the recorder. If a previously recorded video is being played, the controller manipulates the picture handler, pausing and skipping in the video. All of these actions are carried out according to user inputs.

2.2 Picture Handler

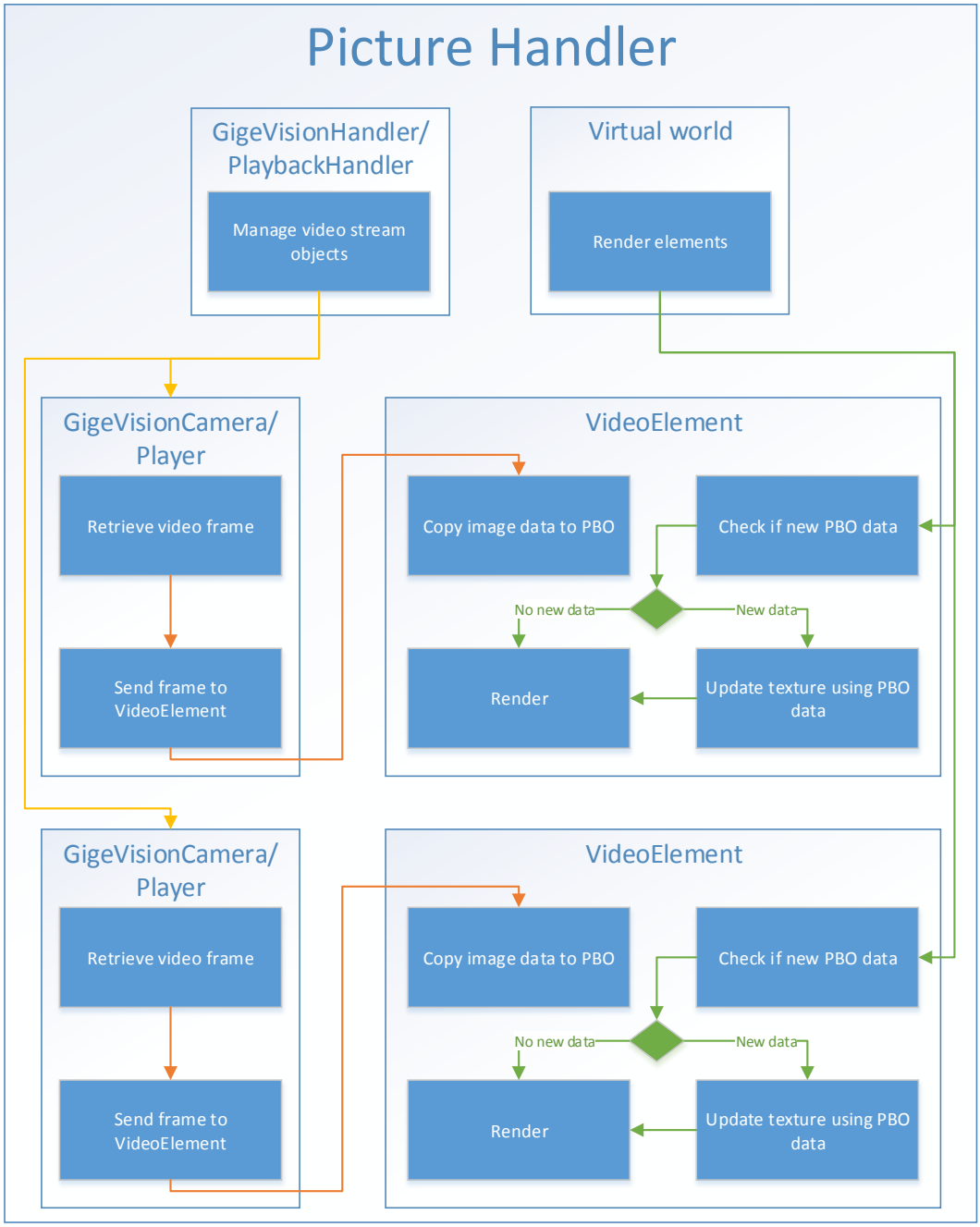

Figure 2: Picture Handler flowchart

The picture handler consists of multiple classes working together to retrieve video from a desired source and feed it into the virtual world. Because we have multiple types of video sources, we need to have a different set of classes for each one.

There are three classes that are used for each video type. The first is a video stream class, which has the task of retrieving the data for a single video stream of a specific type (live from a GigE Vision camera, or recording from disk). The second class is a video handler. It has the task of managing the video stream objects. This means creating them and telling them when to start and stop retrieving data.

The VideoElement class is the third class, and it acts as a common interface between the video stream classes (GigeVisionCamera/Player class) and the virtual world. It accepts raw video data, transferring it to a pixel buffer object (PBO). This PBO is later used as a data source in the virtual world, applying the video data to a surface the user can see in the VR goggles.

PBOs are critical to the performance of our software. They allow for asynchronous transfer of video data to OpenGL. Using multiple PBOs per video feed, we can render the previous video frame while transferring a new one simultaneously. This greatly reduces the time it takes to render a frame, increasing the framerate and thereby the smoothness and responsiveness of the user experience.

2.2.1 Live Video

For live video playback we use cameras that follow the GigE Vision standard (see GigE Vision SDK technical document). GigeVisionHandler is the name of the video handler class we use for live video. This class manages objects of the GigeVisionCamera class, which is the video stream class we use for live video. These two classes act as a wrapper for our implementation of the eBUS SDK, allowing us to abstract away any API specifics. This means that we could easily implement a different GigE Vision SDK down the line if necessary.

In the GigeVisionHandler class, cameras are detected using the eBUS API. For each camera that we want to connect to, a GigeVisionCamera object is created. All configuration specific to a single camera and video stream are carried out in the GigeVisionCamera class, again using the eBUS API.

For live video streaming, a thread is created in the GigeVisionCamera class. This thread interacts with the eBUS API, retrieving video frames from the camera in real-time. These video frames are then converted to a usable format and sent to an assigned VideoElement object, to be displayed to the user. Since a thread is created in each GigeVisionCamera object, multiple video streams can be processed in parallel. This gives our software good performance scaling on multicore processors when increasing the number of cameras in use.

2.2.2 Recorded Video

Recorded video is played from raw video data stored on disk. This data is accessed through the fstream C++ library, and copied into buffers used by the eBUS SDK to hold video frames. This is done because the raw data needs to be converted by the eBUS SDK to a format that can be rendered in the virtual world.

The video handler class used for playing recorded video is called PlaybackHandler. This class is responsible for finding all the video streams in the directory where the recording is stored. It also reads the metadata associated with each video stream. Based on this information, it will create the appropriate number of objects of the Player class.

The Player class is the video stream class used for playing recorded video. In this class, raw video data for a single video stream is retrieved and sent to the virtual world. To achieve smooth playback at the correct speed, two threads are used. The first thread has the job of preloading video frames and placing them in a render queue. This thread will aim to keep a certain number of frames in the render queue at all times, ensuring protection against fluctuations in disk read speed while keeping memory usage under control.

In the second thread, video frames in the render queue are sent to the virtual world at an appropriate rate. This is essential for the recorded video to appear in the same way it did when it was played live from a camera, as opposed to being slowed down or sped up. The frame timing is calculated from the metadata that is stored with the recorded video stream. To maintain this timing, the chrono C++ library has been used. This way we can ensure that playback performance is consistent, and that multiple video streams will stay synchronized.

Our software also supports pausing and skipping through recorded footage. This has also been implemented using metadata. Using the framerate, we can calculate the time elapsed at a certain video frame and how many frames to skip ahead to achieve a desired skip in time. Pausing works in a similar way. When the user presses the pause button, the calculated elapsed time is stored. When the user presses the play button, the time point is calculated back into a specific frame number for each video stream. This allows us to keep video streams with different framerates in sync, even when pausing and skipping.

2.3 Virtual World

The virtual world component has the task of rendering the image that appears on the display in the VR goggles. This image is composed from a set of surfaces placed in a virtual world. When rendering the image, the position and orientation of the VR goggles is taken into account. This means that when the users head moves, the surfaces in the rendered image move accordingly, giving the user the illusion of sitting in a real cinema.

Rendering an image that will be displayed in VR goggles comes with a few additional challenges that you do not have when rendering to a computer monitor. The most obvious one is that humans have two eyes. This means that the image you render actually has to consist of two parts: One for the left eye and one for the right. To convince the user’s mind, these two sub-images also need to have a slightly different perspective. This is done to imitate the distance between the user’s eyes. Effectively, this means that you need to render the virtual world twice for each image.

The other challenge you face when rendering to VR goggles, is that there are lenses between the display and the user's eyes. The lenses distort the image that is displayed on the screen. This means that the image that is sent to the display in the VR goggles needs to be rendered in such a way that it cancels out the distortion caused by the lenses.

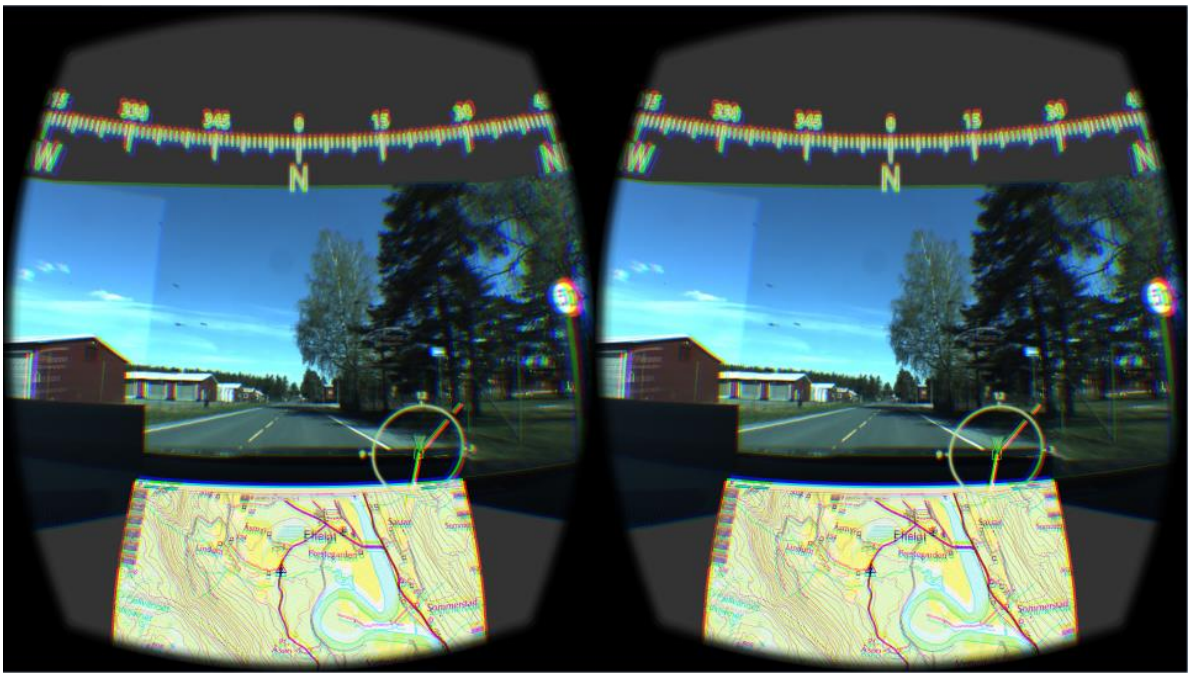

Figure 3: Screenshot of image displayed in VR goggles

There are several classes that have a role in the virtual world component. The RiftApp class handles the two aforementioned challenges specific to our VR goggles, rendering the appropriate image for both eyes while compensating for lens distortion. The actual rendering of each perspective happens in the GLScene class.

The GLScene class does most of the work configuring the OpenGL rendering environment, compiling the shader programs that are executed by the graphics processing unit (GPU). It also manages the surfaces in the virtual world, adding and removing them as the controller requests. When the GLScene receives a render request, it loops through all the surfaces calling a draw method. Surfaces are divided into two categories, local and world. Local surfaces will follow the user's head, while world surfaces will always stay in their fixed position in the virtual world.

There are many different classes that make up the surface system. The existing implementation when we started the project used a monolithic one-size-fits-all class named Surface. Due to the sheer number of unique virtual world objects that use this class in different ways, it has some serious maintainability issues. If you make changes to fix one virtual world object, you risk breaking 20 other virtual world objects in the process.

Because of this we decided to start from scratch and build our own system. An abstract class named Renderable has been created as a basis for all virtual world objects. Specific implementations inherit from this class, allowing us to keep unique

virtual world objects separate on a class level. This greatly improves maintainability and reduces the chance of unintentional breakage when implementing new features. It has also allowed us to make more specific optimizations for certain types of virtual world objects. A good example of this is the VideoElement class, which has a heavily optimized mechanism for loading and presenting video frames.

We have not removed the legacy Surface class due to the number of virtual world objects that still depend on it, but it should be phased out in the future. We have observed memory leaks originating from this class during testing, eventually causing TinyArgos to crash if too many objects of this type are added and removed.

2.4 Recorder

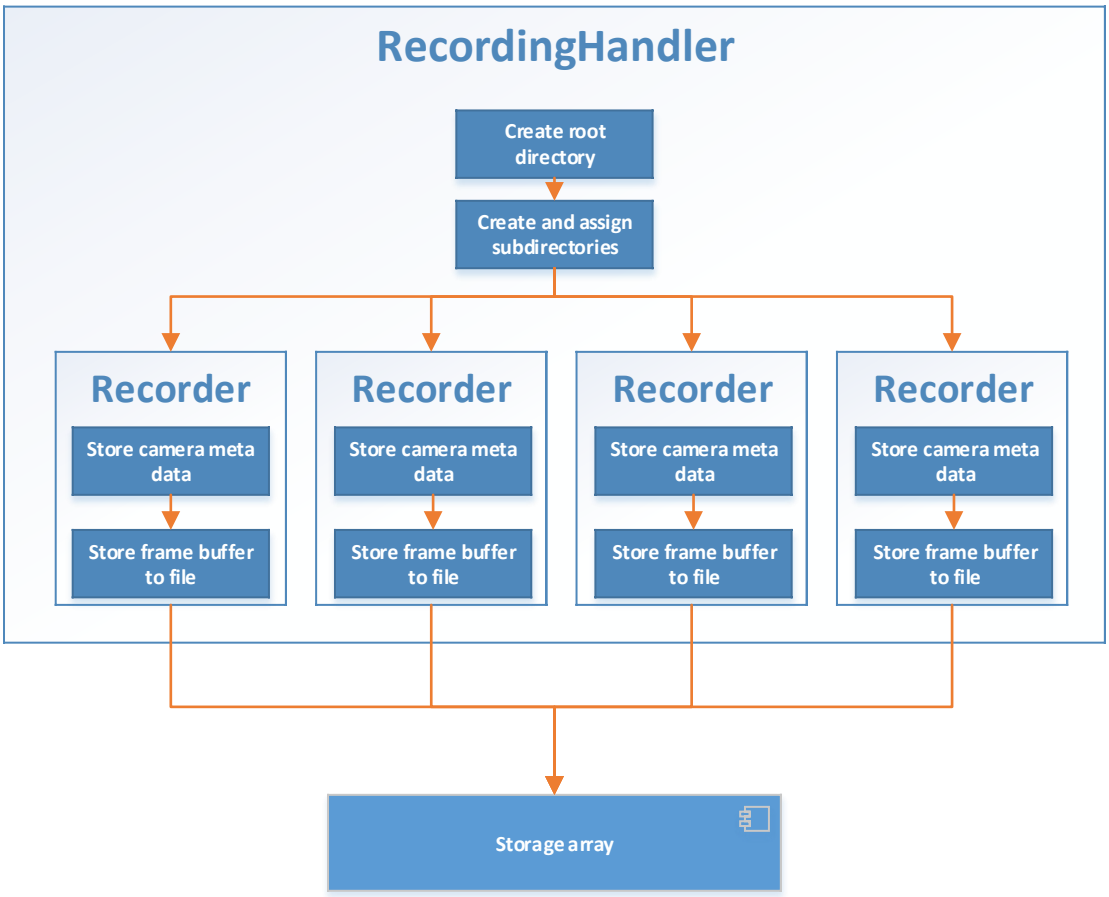

Figure 4: Recorder flowchart

The recorder component has one simple objective – store video from the cameras without interrupting the user experience. It is designed in a similar way to the picture handler. An object of the RecordingHandler class manages objects of the Recorder class.

The RecordingHandler class is responsible for creating the directory structure the recorded video streams are saved in. It creates a Recorder object for each video stream, and sets the appropriate subdirectory to store the video in. The handler tells the recorders when to start and stop. This ensures that when video is recorded from

multiple cameras, the individual recordings are of equal length and synchronized with each other.

To avoid interrupting the user experience, the Recorder class will create a copy of each video frame it has to store. This way, the actual writing to disk can happen in a separate thread, to avoid stalling the picture handler. To keep things as simple as possible, each video frame is stored in its own file. The files are given names that represent the sequence they were stored in. To keep the computational overhead to a minimum, no compression is done to the video frames before they are written to disk.

We want to make it possible to recreate the live experience when the recording is played back. To achieve this, the active config files for video and HUD are copied to the directory that is created by the recording handler. The individual recorders also create a metadata file, containing information about the resolution and framerate of each video stream. This makes it possible to present the video the same way it appeared when it was shown live.

3. Video Stitching

One of the key features of the TinyArgos software is the ability to stitch multiple video feeds into one continuous image. This section explains the technique we have used to achieve this and some of the pros and cons of this solution.

3.1 Inspiration

Latency is a critical aspect of our software, and this greatly limits our options for stitching video feeds together. We wish to avoid having to analyze the image data frame by frame, as this will take up valuable computation time.

The technique we have used was inspired by a typical enthusiast simulator setup (ex. flight simulators or driving simulators). To achieve increased immersion and awareness, it is common to use three monitors. Because of the limited viewing angles of cheaper computer monitors, the left and right side monitor are often angled towards the user. The problem with doing this is that the perspective of the image on these monitors becomes wrong.

To compensate for this issue, many modern simulators have the ability to adjust the perspective of the side monitors. This makes the image continuous across all three screens, despite the different angles.

3.2 Technique

The issue described in the previous section is about adapting the image to the physical placement of the screens. Our technique is basically the exact opposite of this. We receive images from the cameras that have a specific perspective, and we

have to place our “screens” in such a way that the combined image appears to be continuous.

There is an immediate problem with this approach however. The images rendered by a simulator have no distortion. Only the perspective differs. This is critical to making the images line up at the edges of the screens, so we first have to find a way to correct the distortion that is caused by the camera lenses.

The various types of distortion are discussed in the technical document for lenses. Our lenses have significant barrel distortion. This means that we need to stretch the image at the corners to compensate. To do this, we are changing the geometry of the “screens” we project the camera images on. Instead of making the “screen” flat, we give it a hemispherical shape curving toward the user. This gives the user the illusion of the corners being stretched out, which is the effect we are looking for. By fine tuning the parameters we use to get the hemispherical shape, we can completely remove the distortion caused by the lens. This means that a straight line in the real world will look straight on our “screen” as well. With the distortion issue out of the way, it is simply an issue of fine tuning the position of the “screens” to give the illusion of a continuous image from the cameras.

3.3 Pros and Cons

The most obvious strength of this solution is that it is computationally very cheap. Giving the “screens” a hemispherical shape increases the vertex count (the complexity of the geometry) slightly, but this has practically no performance impact on a high performance graphics card like the Titan X we are using.

This solution is also fairly simple to implement, since we don’t have to make an algorithm to analyze the contents of the images to stitch them together. The distortion is the biggest challenge, and beyond that it’s just a simple matter of adjusting parameters to fine tune the positioning of the “screens”.

A big drawback with this solution is that it is completely “dumb”. It depends on the camera rig being stable (the physical placement of the cameras is completely unchanging), and for optimal results it requires fine tuning by the user. If the camera rig were to go out of alignment there is no way to compensate on the fly, because the software has no knowledge of contents of the images and how they align.

Another issue is that this solution struggles with objects that are close to the cameras. This is a result of the cameras not having the exact same point of origin – they will always be placed above or next to one another in some fashion. At distance the issue is minimal, but at close range the difference in placement becomes apparent.

4. Coding Style

It is common knowledge in the software community that the use of a coding standard makes it easy to validate and understand code written by others, and will increase the maintainability of the code. Therefore we have made a set of coding rules that everyone should follow. [1]

4.1 Classes

Struct is only for objects carrying data where all data fields are used directly and not through method calls. If more functionality is required (e.g. methods), classes should be used.

- Avoid calling virtual methods in constructors
- Avoid implicit conversions
- Support copy and/or move only if it is required and useful for the class.

Declaration order:

- Private→protected→public
    - Typedefs and enums
    - Constants
    - Constructor
    - Destructor
    - Methods
    - Data members

4.2 Header Files

All header files must be self-contained, i.e. have header guards and include all other headers required. The header guards prevent multiple inclusions and use the following format:

PROJECT_PATH_FILE_H_

The name is based on the project source tree.

4.3 Functions

Parameter order is: First inputs (pass by value) and then outputs (pass by reference). New parameters are added at the end of their respective section.
In/out parameters can be placed in a separate section or together with related parameters.

If possible, create small and focused functions.

4.4 Naming

The name should be as descriptive as possible and immediately identify the entity. Understanding is more important than minimizing the horizontal space required and you should avoid using abbreviations unfamiliar to people outside the team.

Type names (classes and structs):

- Start with capital letter
- Every new word start with capital letter (UpperCamelCase)
- No underscores
- Example: ExampleClass

Variable and Function names:

- Start with lowercase letter
- Every new word start with capital letter (lowerCamelCase)
- No underscores
- Variable example: myLocalVariable
- Function example: myLocalFunction()

4.5 Documentation / Comments

Classes:
Explain what the purpose of the class is. This must be done in the header file.

Example:

```
/*
This class captures the picture data from the cameras.
Any inheritance
*/
```

Functions:
What the function does
Parameters
Return

Example:

```
/*
This function calculates 1+1
Param a, integer containing the first number.
Param b, float containing the second number.
Return sum, float containing the sum of the two numbers.
*/
```

Bibliography

[1] "WikiBooks," 2016. [Online]. Available: https://en.wikibooks.org/wiki/C%2B%2B_Programming/Programming_Languages/C%2B%2B/Code/Style_Conventions#References. [Accessed 09 March 2016].

Iteration Reports 2.0

Created by: Leiv Fredrik Berge
15.03.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

Table of contents

1. Document Overview

The purpose of the iteration reports is to keep track of the project progress and keep the development team informed. The iteration report contains milestones, objectives, work items, evaluation criteria and a review of each iteration. It contains estimated hours needed to complete each task. This should be read with the iteration burndown report. Work items are prioritized from 0 to 5, where 0 is highest priority and 5 lowest.

Describes

- the key milestones.
- objectives.
- work items with responsible person.
- evaluation criterias.
- details of the review in each iteration.

1.1 Document History

Version	Changes	Date	Created by
0.1	Formalized document	14.03.2016	Leiv Fredrik Berge
1.0	Reviewed and updated	15.03.2016	Ingvild Damtjernhaug
1.1	Add iteration 5	05.04.2016	Mathias Havdal
1.2	Iteration 4 review	07.04.2016	Ingvild Damtjernhaug
1.3	Changed format on name and date in document history	10.04.2016	Trond Egil Hammer
1.4	Updated i5a, i5b and i5c	14.04.2016	Leiv Fredrik Berge, Mathias Havdal
1.5	Added I6, updated i5b with review	15.04.2016	Leiv Fredrik Berge
1.6	Adjustment to i5c	21.04.2016	Ingvild Damtjernhaug
1.7	New document overview	29.04.2016	Leiv Fredrik Berge
1.8	I6 updated	12.05.2016	Leiv Fredrik Berge
1.9	Fixed layout. Corrections and clarifications.	16.05.2016	Morten J. Barbala
1.10	Pre final review	18.05.2016	Thomas Hansen
1.11	Formatting, proofreading, added headings, added I7	19.05.2016	Ingvild Damtjernhaug Leiv Fredrik Berge
2.0	Final review	20.05.2016	Trond Egil Hammer, Thomas Hansen Ingvild Damtjernhaug

1.2 References

Title	Document	Version
Project Plan	Argos_project_1_0.mpp	1.0
Burndown report	doc-112_Iteration_burndown_report_1_0.xlsx	1.0

1.3 List of Tables

2. Iteration I1: Project Start

Iteration I1, project start, belongs to the inception phase.

2.1 I1 Key Milestones

Title	Date
Iteration start	11.01.2016
Technical vision	14.01.2016
Iteration stop	20.01.2016

Table 1: Key milestones iteration I1

2.2 I1 High-Level Objectives

- Develop an initial project plan
- Develop a common technical vision
- Set up key development tools and environment

2.3 I1 Work Item Assignments

Task	Priority	Responsible	Estimated hours
Develop document standard	1	Ingvild Damtjernhaug	12
Initial project plan	0	Leiv Fredrik Berge	100
Develop technical vision	0	Trond Egil Hammer	48
Tailor the process	1	Leiv Fredrik Berge	30
Set up tools	2	Mathias Havdal	75
Verify tools and configurations	2	Morten J. Barbala	30
Deploy process	1	Leiv Fredrik Berge	20

Table 2: Work item assignments iteration I1

2.4 I1 Evaluation Criteria

- Stakeholder acceptance of technical vision
- Tools and configuration test passed

2.5 I1 Iteration Review

The technical vision document was approved by the stakeholder. The tools and configuration did not meet the requirements to the development process. We have Visual Studio 2012, but we need Visual Studio 2013 or 2015 for it to support the Oculus SDK and the OpenGL environment. The office space is adjusted to fit the project, and we have created the network and sever setup. This must be revisited in a later iteration, before development commences.

3. Iteration I2: Requirement Specification

Iteration I2, requirement specification, belongs to the inception phase.

3.1 I2 Key Milestones

I2 Requirement specifications

Title	Date
Iteration start	20.01.2016
Use cases	27.01.2016
System-wide requirements	27.01.2016
Presentation 1	09.02.2016
Iteration stop	09.02.2016

Table 3: Key milestones iteration I2

3.2 I2 High-Level Objectives

- Identify and outline requirements
- Detail use-case scenarios
- Detail system-wide requirements
- Create test cases
- Create first presentation

3.3 I2 Work Item Assignments

Task	Priority	Responsible	Estimated hours
Identify and outline requirements	0	Trond Egil Hammer	75
Detail use-case scenarios	0	Thomas Hansen	75
Details system-wide requirements	0	Thomas Hansen	100

Create test cases	1	Morten J. Barbala	40
Set up tools	2	Mathias Havdal	40
Verify tools and configurations	2	Morten J. Barbala	30
Risk review meeting	1	Leiv Fredrik Berge	12
Create first presentation	0	Ingvild Damtjernhaug	100
First presentation	0	Leiv Fredrik Berge	12

Table 4: Work item assignments iteration I2

3.4 I2 Evaluation Criteria

- Stakeholder acceptance of use cases
- Stakeholder acceptance of system wide requirements
- Successful response for the first presentation

3.5 I2 Iteration Review

We did not finish the setup of tools and environment from the first iteration. We adjusted, and configured the tools to our satisfaction in this iteration. We are now running Visual Studio 2015, with current SDKs. This iteration's main focus was the requirements modelling. We did get approval from our stakeholders on both use cases and system wide requirements, and received good feedback on the presentation, resulting in a successful iteration. We see that we use slightly more time on the presentation than expected, and a little less time on the requirements modelling than we planned for. We will keep an eye on this for the upcoming iteration, on whether or not we need to re-evaluate the plan, however the work tasks is performed to our satisfaction which is the most important aspect.

4. Iteration I3: Requirement & Architecture

Iteration I3, requirement and architecture, belongs to the elaboration phase.

4.1 I3 Key Milestones

I3 Requirement & architecture

Title	Date
Iteration start	09.02.2016
Refine architecture	01.03.2016
Develop solution increment	01.03.2016

Technical research on network solutions	22.02.2016
Technical research on lenses	22.02.2016
Iteration stop	22.02.2016

Table 5: Key milestones iteration I3

4.2 I3 High-Level Objectives

- Create a stable system architecture
- Create a stable software architecture
- Develop the solution process
- Decide on lenses for our camera rig
- Compile and run existing code

4.3 I3 Work Item Assignments

Task	Priority	Responsible	Estimated hours
Refine architecture	0	Thomas Hansen	200
Develop solution process diagrams	1	Mathias Havdal	40
Risk review meeting	1	Leiv Fredrik Berge	12
Refine risk document	0	Trond Egil Hammer	40
Update SDK	1	Mathias Havdal	20
Migrate VS13->VS15	1	Mathias Havdal	20
Run code base	1	Mathias Havdal	10
Research network solutions	1	Morten J. Barbala	40
Research lenses	0	Trond Egil Hammer	70
Technical document VR goggles	1	Ingvild Damtjernhaug	40

Table 6: Work item assignments iteration I3

4.4 I3 Evaluation Criteria

- Project group acceptance and consensus of architecture
- Successful compelling and running code
- Decide on lenses to use for the camera rig

4.5 I3 Iteration Review

The last iteration left us in a very comfortable spot with the requirements modelling. Only minor adjustments to the requirements were needed for our acceptance in this iteration. Also we did see the need for a couple additional tests to sufficiently prove the performance of our system hardware.
The code now compiles and work as it should. This did involve quite a bit of work we didn't anticipate. We knew we needed to migrate from VS13 to VS15. A little surprisingly to us, there was quite a bit of breaking changes, especially to the time.h. Oculus has also released two SDK versions after the 0.6.0.0 that TinyArgos used. We needed to perform the migration in two steps to make sure we updated all the functionality. All the other libraries also needed to be updated. We reached consensus on the pipe-filter architecture, with some adjustments. As we start work with integrating the cameras in the next iteration, we will keep working on refining the architecture. We had to dig a lot deeper into lenses than we expected. Apparently we are dealing with an issue that is very particular, so finding useful information has taken more time than we thought initially. We've really had to dig deep into different technical aspects of lenses. We have gotten some help by the optometrists at the college, in creating and understanding lens designs. We have found lenses we are comfortable ordering as soon as we get acceptance from the stakeholder in our solution

5. Iteration I4: Integrate Cameras

Iteration I4, integrate cameras, belongs to the construction phase.

5.1 I4 Key Milestones

I4 Integrate cameras

Title	Date
Iteration start	22.02.2016
Refine architecture	18.03.2016
Order lenses	05.03.2016
Implement GiGE SDK	07.03.2016
Receive picture data	18.03.2016
Second presentation	15.03.2016
Iteration stop	17.03.2016

Table 7: Key milestones iteration I4

5.2 I4 High-Level Objectives

- Refine architecture
- Implement GigE SDK
- Migrate code to 64 bit
- Order lenses
- Run code base with picture data from our cameras

5.3 I4 Work Item Assignments

Task	Priority	Responsible	Estimated hours
Refine architecture	0	Thomas Hansen	100
Choose GigE SDK	0	Mathias Havdal	20
Implement GigE SDK	0	Mathias Havdal	50
Order lenses	0	Trond Egil Hammer	30
Implement config XML parsing	1	Leiv Fredrik Berge	50
Create functions to receive picture data	0	Mathias Havdal	130
Create second presentation	1	Ingvild Damtjernhaug	100
Second presentation	0	Leiv Fredrik Berge	12

Table 8: Work item assignments iteration I4

5.4 I4 Evaluation Criteria

- Receive picture data from the camera rig
- Successful ordering of lenses
- Good feedback on the second presentation

5.5 I4 Iteration Review

In iteration 4 we managed to receive picture data from all the cameras, and successfully upgraded the code and libraries.
We ordered the lenses, and were afraid it would take a while before we actually got them. But it turned out well, the lenses were delivered quite fast, and they are now ready for testing.
We held the second presentation 15.03. The feedback on the technical parts of the presentation was good. A misunderstanding led us to believe that the focus for the second presentation were on the technical aspects. We should have had more focus on the process. We got some negative feedback on our documentation, as there were documents missing from the delivered CD. We are not allowed to connect our workstations to Internet, and because of that, we save documents at different places. Only the documents from our local server were added to the CD. We need to make sure this does not happen again.
There was also some inconsistency in our delivered documents. This will be fixed, and get high priority before the third and final presentation.

6. Iteration I5A: Merge Images

Iteration I5A, merge images, belongs to the construction phase.

6.1 I5A Key Milestones

Title	Date
Iteration start	04.04.2016
Get live video feed in Oculus headset	15.04.2016
Merge two images	22.04.2016
Merge all images	06.05.2016
Iteration stop	06.05.2016

Table 9: Key milestones iteration I5A

6.2 I5A High-Level Objectives

- Refine architecture
- Display live video feed in Oculus
- Merge video feeds into a continuous image
- Implement appropriate configuration files

6.3 I5A Work Item Assignments

Task	Priority	Responsible	Estimated hours
Refine architecture	0	Leiv Fredrik Berge	50
Transfer video data to OpenGL render medium	0	Mathias Havdal	100
Create and position OpenGL geometry to render video on	0	Thomas Hansen	100
Solution for merge areas	1	Leiv Fredrik Berge	50
Review config system	0	Ingvild Damtjernhaug	100

Table 10: Work item assignments iteration I5A

6.4 I5A Evaluation Criteria

- Successfully display live video in Oculus
- Merge video feeds seamlessly
- Should be able to configure the system using only config files

6.5 I5A Iteration Adjustments

Date	Reason	Responsible	Adjusted hours
08.04.2016	Smoother implementation of GigE Vision than anticipated.	Leiv Fredrik Berge	-75

Table 11: Iteration adjustments iteration I5A

6.6 I5A Iteration Review

We have done a lot of work with the software architecture. Refining and updating all aspects of the software architecture to incorporate the changes and upgrades in the code base. We've created sequence diagrams for the key functions and flowcharts for each module.
We encountered a major issue with the video stream buffers we had to deal with. As soon as we deleted a buffer, the entire program crashed. And if we can't delete buffers, it will keep filling up in memory and eventually crash. We did a lot of investigation, and found out that the EBUS SDK seems to have a bug in the 64 bit version of the SDK. So we had to migrate the code back to 32 bit, as the 32 bit EBUS

SDK worked perfectly. 32 bit is completely fine for this project, but it should be upgraded to 64 bit when Pleora releases an update to EBUS.
We have also rewritten the entire configuration system. We wanted the software to be more scalable and configurable, so we moved as much as possible out to the XML files instead of hardcoding it. The config files allows us to very easily move the system onto different hardware, share configuration setups or recreate the exact setup from a recording to a playback scenario. Along the way we've done major clean-up of allocation of functionality, fixed memory leaks and other minor bugs.
We added support for PBO. This is a more clever way of packing the frame buffers, and should yield a performance increase and better stability.
We've also rearranged the surfaces on which the videos are rendered on in a new way. Now we have full freedom to place the surfaces in all three directions in the configuration file. And we have distortion correction functionality. With this we place the images on a curved surface to correct for the distortion the lenses add to the video.
We've created a prototype for the camera rig. We needed to have a rig, in order to test the distortion correction and the stitching functionality. This is quite crude, but we have designed a more permanent rig, that will either be 3D printed or manufactured in wood.

7. Iteration I5B: Record Video Stream

Iteration I5B, record video stream, belongs to construction phase.

7.1 I5B Key Milestones

Title	Date
Iteration start	04.04.2016
Add controls for recording	08.04.2016
Record video streams to file	15.04.2016
Iteration stop	15.04.2016

Table 12: Key milestones iteration IB5

7.2 I5B High-Level Objectives

- Should be able to start and stop recording
- Recording status should be visible to user
- Should be able to record video to file from all cameras in parallel

7.3 I5B Work Item Assignments

Task	Priority	Responsible	Estimated hours
Add interface to control recording	0	Trond Egil Hammer	40
Add HUD notification for recording status	0	Morten J. Barbala	75
Implement system for recording video to file	0	Trond Egil Hammer	150

Table 13: Work item assignments iteration I5B

7.4 I5B Evaluation Criteria

- User is able to control recording start/stop
- Confirm that video file is playable
- Recording status should be visible in the Oculus headset

7.5 I5B Iteration Adjustments

Date	Reason	Responsible	Adjusted hours
08.04.2016	The architecture allowed for a very clean integration of recording that required less rewrite of code base than anticipated.	Leiv Fredrik Berge	-50
13.04.2016	Less performance optimization required than anticipated.	Leiv Fredrik Berge	-50

Table 14: Iteration adjustments iteration I5B

7.6 I5B Iteration Review

As we went to work on this iteration we didn't know what to expect. This is a totally new functionality, and as far as our research is concerned, not a very common action to perform with our setup. However we did have a general idea; we wanted to write the buffers to disk with as little overhead as possible. We considered the to use functionality from the Ebus SDK to use a mpg container, and drop the video files into that, however we instead went with a more basic approach of writing the buffers raw to file. The main reasons is that this requires very little overhead, the file sizes are large but manageable and it makes playback implementation very smooth, as we basically can just read the buffers as if they came from the camera system. As work progressed in this iteration we realized that our idea would actually work, probably even better than we anticipated, so we were able to slash some hours of the expected workload. The performance, which faster than expected, also meant we did

not have to spend as much time optimizing the speed of recording as initially planned. This is in great part thanks to the planning and the problems we solved in advance, like setting up our super-fast SSD array. Trond Egil and Morten has been lead in this iteration with design help from Mathias and Leiv Fredrik. Working in parallel with I5a has worked out great for us, as it has engaged the entire team in useful coding task throughout the entirety of this iteration. The repository manager, Mercurial (Hg), has been key for this to be achieved. Hg allows us to develop multiple branches at the same time, and then integrate into the code base when new functionality is stable. The evaluation criteria are considered to be met in all accounts. No tasks from this iteration is expected to slide into I5c, however the playback and recording as quite interconnected, so there is slight chance that the playback module will reveal aspects from the recorder that needs to be added.

8. Iteration I5C: Video Playback

Iteration I5C, video playback, belongs to construction phase.

8.1 I5C Key Milestones

Title	Date
Iteration start	18.04.2016
Read and create buffer from raw files created by recorder	20.04.2016
Display buffer in renderer	22.04.2016
Time rendering appropriately (correct framerate, etc.)	26.04.2016
Implement pausing, skipping forward/backward	28.04.2016
Implement configuration system with metadata for video feeds	04.05.2016
Iteration stop	04.05.2016

Table 15: Key milestones iteration I5C

8.2 I5C High-Level Objectives

- Should be able to play video created by the recorder
- Playback framerate should be equal to recording framerate
- Should be possible to pause, skip forwards and backwards in video by fixed amounts of time
- Playback system should be able to configure surfaces as they were recorded
- Playback system should be able to determine the correct framerate for video playback

8.3 I5C Work Item Assignments

Task	Priority	Responsible	Estimated hours
Add functionality for creating a buffer from raw data on disk	0	Morten J. Barbala	40
Add functionality for displaying buffers in renderer with correct timing and framerate	0	Mathias Havdal	80
Add functionality for controlling playback	1	Morten J. Barbala	60
Add functionality for parsing config files specific to the recording that is being played	1	Leiv Fredrik Berge	60

Table 16: Work item assignments I5C

8.4 I5C Evaluation Criteria

- Video playback has a correct framerate (video is not sped up or slowed down)
- Video playback is consistent (no stuttering or small variations in framerate)
- User is able to pause video and skip forwards/backwards

8.5 I5C Iteration Adjustments

Date	Reason	Responsible	Adjusted hours
21.04.2016	Successful demo of recording functionality at 21.04. Seems likely that the integration against earlier work is smoother than expected, due to the architecture and planning.	Leiv Fredrik Berge	-40
02.05.2016	The work as progressed nicely and faster than planned.	Leiv Fredrik Berge	-50

Table 17: Iteration adjustments iterations I5C

8.6 I5C Iteration Review

This iteration was very closely linked to I5b. Basically it was very hard for us to really test the recording functions before the playback was implemented, and vice versa. So when we started the iteration we were not sure how much trouble we were going

to run into. And there were some issues, but in general our architecture, idea and implementation worked out great. We did however have to rewrite some of the recorder and playback when we implemented the PBOs in I5a.
We also were able to avoid a lot of problems in advance. Our initial calculations of required disk writing and reading speed turned out to be pretty much spot on. And our solution with the three SSD RAID0 array worked perfectly.
We expected timing and quantization to be a potential issue, but after some performance adjustments and optimizations, the timing is very good. We also found a clever way of quickly and exactly store metadata about the video stream and configuration, and reloading it in the player.
We did discover that some of the old code that is quite unpleasant, giving us a slight memory leak when swapping configuration files. The issue is buried pretty deep in a part of the program that needs to be refined. Fixing the issue will require significant effort, so the better solution is probably to rewrite the entire module. This will streamline the code and yield a more stable solution.

9. Iteration I6: Add Markers and Information

Iteration I6, add markers and information, belongs to construction phase.

9.1 I6 Key Milestones

Title	Date
Iteration start	05.05.2016
Clean up HUD generation	12.05.2016
Generate test HUDs	14.05.2016
Implement system tests	15.05.2016
Create systems guide	15.05.2016
Finalize documentation	16.05.2016
Iteration stop	20.05.2016

Table 18: Key milestones iteration I6

9.2 I6 High-Level Objectives

- Allow for text and image objects to be placed over the video stream
- Allow for an interface to gather information from other systems to generate HUD objects
- Perform system tests
- Finalize documentation

9.3 I6 Work Item Assignments

Task	Priority	Responsible	Estimated hours
Clean up HUD objects	0	Thomas Hansen	60
Generate test HUD objects	1	Ingvild Damtjernhaug	20
Perform system tests	0	Morten J. Barbala	60
Create systems guide	1	Mathias Havdal	30
Finalize code documentation	0	Leiv Fredrik Berge	75

Table 19 Work item assignments iteration I6

9.4 I6 Evaluation Criteria

- HUD objects is visible in the Oculus
- System passes systems tests successfully
- Documentation receives positive feedback at the final presentation
- System receives positive feedback at final presentation
- System is accepted by the stakeholder
- Documentation is accepted by the stakeholder

9.5 I6 Iteration Review

We have integrated new functionalities into the code base along the way, so there was no huge integration work that was left for the final development iteration. The major development task that was left was to improve the HUD system that was left from the previous group. We did find there was quite a few problems with that integration, but decided to leave much in place and rather incrementally phase out the old system. The new HUD system doesn't leak memory as the old one did, fixes some issues with static objects and is much less complex to work with. We also integrated some of the features we started on in I5a to split the configuration system of the HUD objects and the video objects. The system outperformed our expectations greatly in our internal systems acceptance test. The latency is unnoticeably low and the seams aren't annoying while driving. There are, however, some issues with the Oculus: The tracker expects to be stationary, so while turning, the tracker and the gyroscope in the Oculus gets a little out of whack. This results in the video feed snapping into place every so often to realign the Oculus with the tracker. The problem was not severe enough to hinder the usage of the system, but should be improved before the system is deployed in a real world situation. All in all we are very

pleased with the progress of this iteration and the entire project. We also received great feedback from the project owner and stakeholder in our systems acceptance test.

10. Iteration I7: Delivery

Iteration I7, delivery, belongs to transition phase.

10.1 I7 Key Milestones

Title	Date
Iteration start	21.05.2016
Documentation deadline	23.05.2016
Final presentation	27.05.2016
Project delivery	01.06.2016
Final demonstration	10.06.2016

Table 20: Key milestones iteration I7

10.2 I7 High-Level Objectives

- Finalize all documentation
- Finalize all code
- Perform the final presentation
- Deliver the project to the project owner

10.3 I7 Work Item Assignments

Task	Priority	Responsible	Estimated hours
Finalize documentation	0	Ingvild Damtjernhaug	50
Finalize final presentation	0	Leiv Fredrik Berge	50
Perform final presentation	0	Leiv Fredrik Berge	10
Deliver the project	0	Leiv Fredrik Berge	30

Table 21 Work item assignments iteration I7

10.4 I7 Evaluation Criteria

- Receive positive feedback and acceptance from project owner on project delivery
- Receive positive feedback on the final presentation

Evaluation 1.0

Created by: Leiv Fredrik Berge
04.05.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

KONGSBERG

Table of Contents

1. Document Overview

The purpose of the evaluation document is to evaluate the project plan, iterations and project model. It also shows the challenges we have faced during the development process. The document details the hours we have spent individually and as a group in the development iterations. Finally, it includes each team member's evaluation of the project.

Describes:

- our experience with the project.
- our experience project plan.
- our experience with iteration plan.
- our experience with the project model.
- the challenges we have faced.
- hours spent.
- each group members evaluation of the project.

1.1 Document History

Version	Change	Date	Created by
0.1	First version	04.05.2016	Leiv Fredrik Berge
0.2	Added time usage	06.05.2016	Leiv Fredrik Berge
0.3	Added individual hours spent	09.05.2016	Leiv Fredrik Berge
0.4	Added technical difficulty and acknowledgements	11.05.2016	Leiv Fredrik Berge
0.5	Fixed heading and table of contents. Corrections and clarifications	16.05.2016	Morten J. Barbala
0.6	Added planning of the project	16.05.2016	Leiv Fredrik Berge
0.7	Added reflections	20.05.2016	Ingvild Damtjernhaug
0.8	Added reflections	21.05.2016	Mathias Havdal
1.0	Final review	21.05.2016	Thomas Hansen Trond Egil Hammer

1.2 Referenced Documents

Title	Document	Version
Project plan	doc-1112_project_plan_2_0.docx	2.0
Glossary	doc-1113_glossary_2_0.docx	2.0

1.3 List of Figures

2. Planning of the Project

We formed the group in the fall of 2015 and started looking for a suitable project. We were familiar with Project Argos, so we knew it fitted well with our group constellation. The initial planning started in the end of 2015, with the proper initiation in January of 2016. The first week of 2016 was spent researching project processes and dividing areas of responsibility. We chose a very flat hierarchy and informal structure, as we all knew each other well and have worked well together in earlier projects.

2.1 Project Plan

We created an initial plan in Microsoft Project with a Gantt diagram, activities and tasks. We set up the tools and guidelines for the project, with how to deal with meetings, communication, work registration and usage of key tools and programs. This has been our basis to make sure the project progressed and keep us on track. We printed out the entire Gantt diagram and hung it up in our office space along with a calendar to mark off important milestones and events. The plan has been revised in each risk review meeting to make sure we are on track and do minor adjustments to ensure we reach our targets. From the Project file we exported the task list, work breakdown structure (WBS). We used the WBS as the basis for the registration of our hours, enabling us to create statistics over how we spend the hours. We used a Google Form to register hours of work, which went into a Google Sheet.

2.2 Iteration Plan

The top level project plan has been divided into seven iterations. The iteration helped us to break down the project into smaller, more manageable chunks of work. This made it a lot easier to estimate the hours needed to complete the work and to keep track of the progress. We also used an Excel sheet that detailed the work done in each iteration according to registered hours from the Google Sheet. This was the basis for the iteration burndown report. The burndown made it very clear to everyone how the work progressed. We also made major improvements to work done early in the process by revisiting it in later iterations. The iterations also made it possible for us to make sure everybody had well-defined tasks and responsibilities.

2.3 Task Level plan

On the day to day, task level, we used a Kanban board to organize the tasks at hand. We broke down the objectives from the iteration into small items. This made it so that everyone had a clear understanding of the task at hand and coming up shortly. We discussed these tasks at our stand up meetings and updated the iteration burndown and project plan as task went into development, testing, integration and done lanes at the Kanban board. The clear and visual board was a great tool for us to stay focused on the key tasks and keep everyone in the loop.

2.4 Project Process

The project followed a project process called OpenUP. This is a process derived from unified process with some agile principles attached. We found it to be very useful in organizing the project, especially to get us started with project. However, the types of artefacts that the model specifies did not comply 100% with our needs and the stakeholders' expectations. We found this to be true when we tried to follow the guidelines too rigorously. We therefore ended up with modifying some of the artefacts, for example grouping some documents together into one. Overall we are very satisfied with the decision to use OpenUP. The Eclipse Foundations OpenUP wiki has been a great resource for us to make sure we are driving the project forward in a manner that complies with the OpenUP process. We have definitely seen the benefits of an iterative project process, and also to have different phases with different focus areas. OpenUP has also been free enough for us to make the adjustments we needed to make it fit our needs and comply with the requirements from the project owner and the college. The division into different roles has been a great tool to ensure that everybody have clear responsibilities and task, utilizing the project team as efficiently as possible.

2.5 Meetings

Internally we have used stand up meetings as our primary meeting form. In this quite informal way we have been able to communicate and work very well together as we all share the office space. This has negated the need for many formal meetings, as most issues have been solved informally. The only formal internal meetings we had were five scheduled risk review meetings at the end of iterations. In these meetings we have looked at the project progress with the iteration burndown reports and review the risk analysis document.

We have also had meetings with Alexander Gosling from KDS roughly every two weeks, depending on the need. These meetings have been used to clarify aspects of the project and to agree on major decisions. The meetings have usually lasted 30 minutes and have been very important for us to know we are moving in the right direction. Internal supervisor meetings have been held regularly every two weeks as well, with more frequency right before milestones and deadlines. These meetings have focus primarily on documentation and presentations, and have been very useful for us when producing documentation.

3. Time Usage

The early estimates of the project planned for 600 hours of work for each member, totaling 3600 hours. As the project progressed, we have detailed the plan and broken it down to smaller tasks. This has increased the precision of the plan, and lowered the time usage to about 3200 hours, about 500 hours per team member. The 3600 hours was based on estimates and expectations from the college, however as we did not have a pre-project the hours estimated is a bit high. Because we had success with our project plan and the development, we were able to reach all the milestones and project goals within the slightly reduced timeframe.

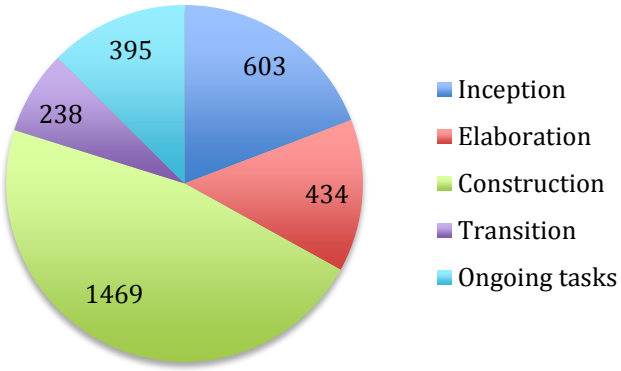

Figure 1: Estimated hours in phases

We have spent the bulk of the time in the construction phase, where all the development has taken place. Our parallel iteration strategy, with merge image development working in parallel with recording and playback, made it easier to utilize the entire team efficiently. It enabled us work mostly in pairs on the development tasks, dividing the task into subtasks and increasing the development velocity. We did overestimate the time needed for most of the development tasks, and underestimated the time needed for the documentation work in the construction phase. These two factors did, however, cancel each other out, resulting in that the plan has been quite accurate both in hours and dates. We have not had any major task slide from an iteration into the next one.

3.1 Time by Construction Iterations

In the early iteration of the construction phase the emphasis is on design. The fact that we spent a lot of time on design early on meant we had to spend less time on it later. This approach also made integration of the new functionalities in each iteration much smoother than we expected due to the thorough design work. The testing portion of the work increases as the iteration progresses, so we spent more time testing the system when more of the functionalities were implemented. This is how we planned and expected the focus to shift along the development process, and we are very happy to see that we were able to execute the plan successfully.

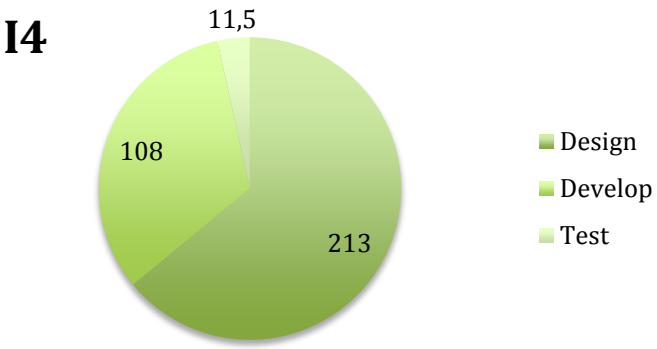

Figure 2: Hours spent in iteration I4 divided by top level tasks

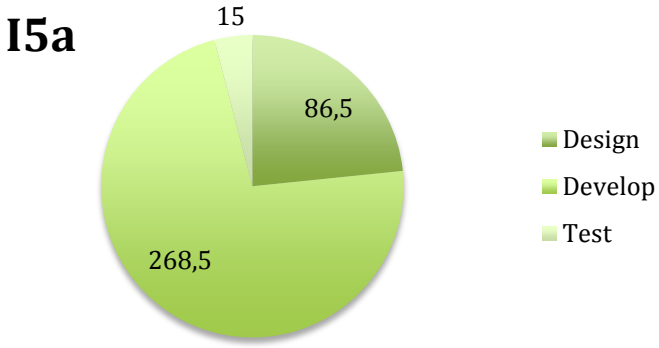

Figure 3: Hours spent in iteration I5a divided by top level tasks

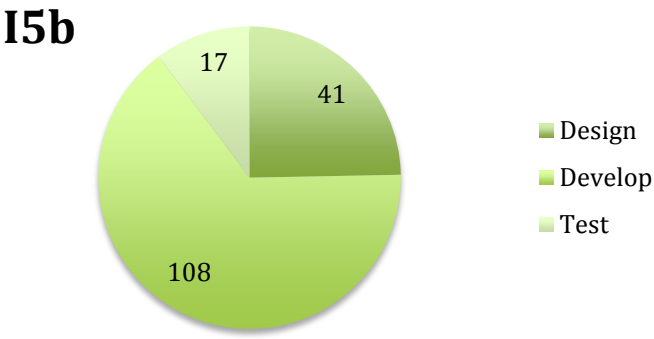

Figure 4: Hours spent in iteration I5b divided by top level tasks

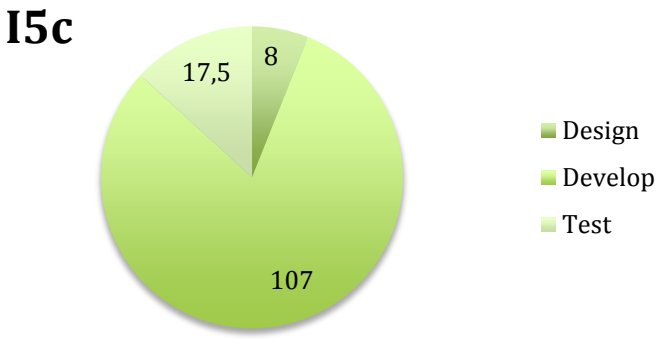

Figure 5: Hours spent in iteration I5c divided by top level tasks

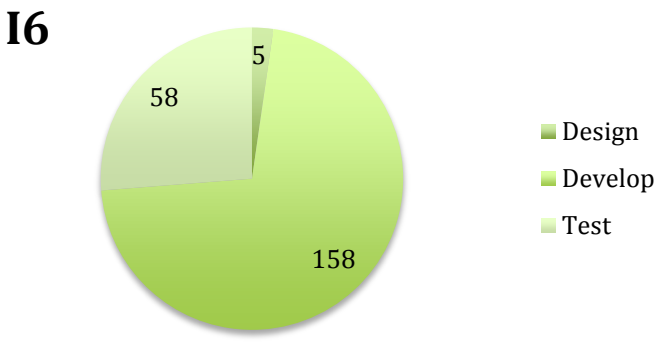

Figure 6: Hours spent in iteration I6 divided by top level tasks

3.2 Time by Group Members

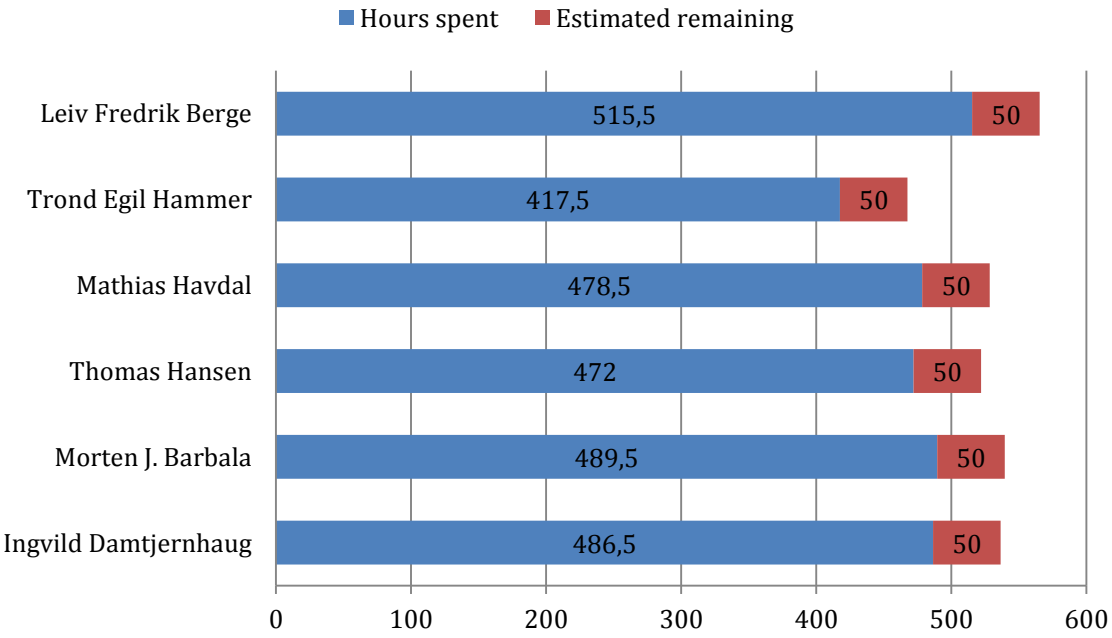

Figure 7: Actual hours spent by each group member

We have worked closely together in our shared office space. This has allowed us to spend less time with formal meetings as we have had constant communication when issues arise. Trond Egil got twins in the middle of the project, so he has some fewer hours than the rest of the project members. Our plan took this into account and Trond Egil had his most important task as analyst early in the project and as far as possible had tasks which allowed him to work from home. In the diagrams below you can see how each team member's hours breaks down into the major elements of the project.

Each team member's hours are broken down for each major group of tasks. As expected there are some differences that mostly boils down to the different roles each member has had. Meeting and administrative contains stakholder and supervisor meetings, risk review meetings , work with the web page, general research and administrative tasks that did not fit anywhere else. Presentations contains all the work creating, rehearsing and performing the presentations. Project plan includes the work directly related to the development of the project plan and iteration plan and management. The development part contains all the development, design and testing of the software and system. The hours are current for final delivery of documentation and some more hours in presentation is expected to occur before the project ends.

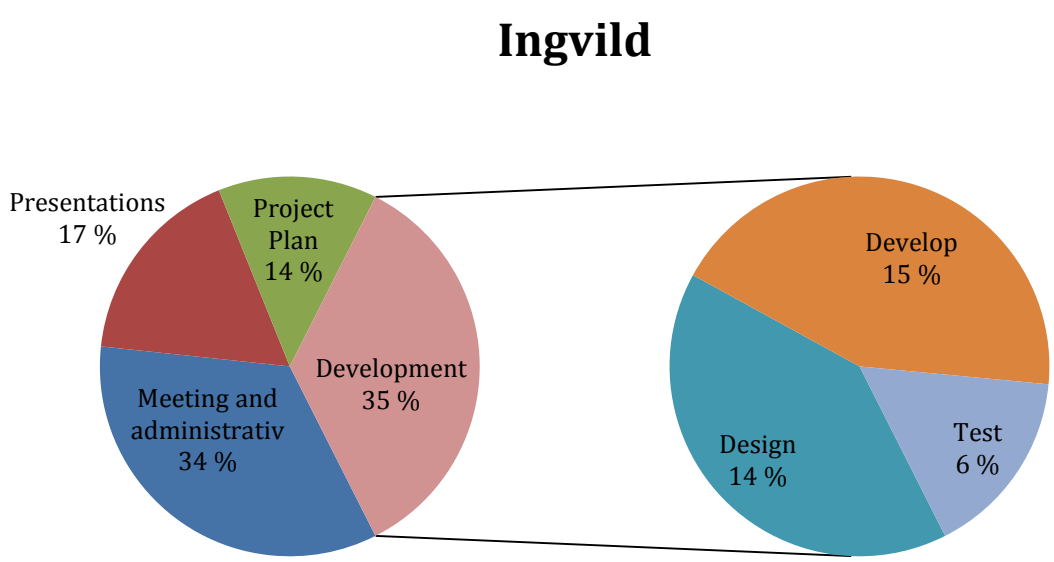

Figure 8: Ingvild's hours

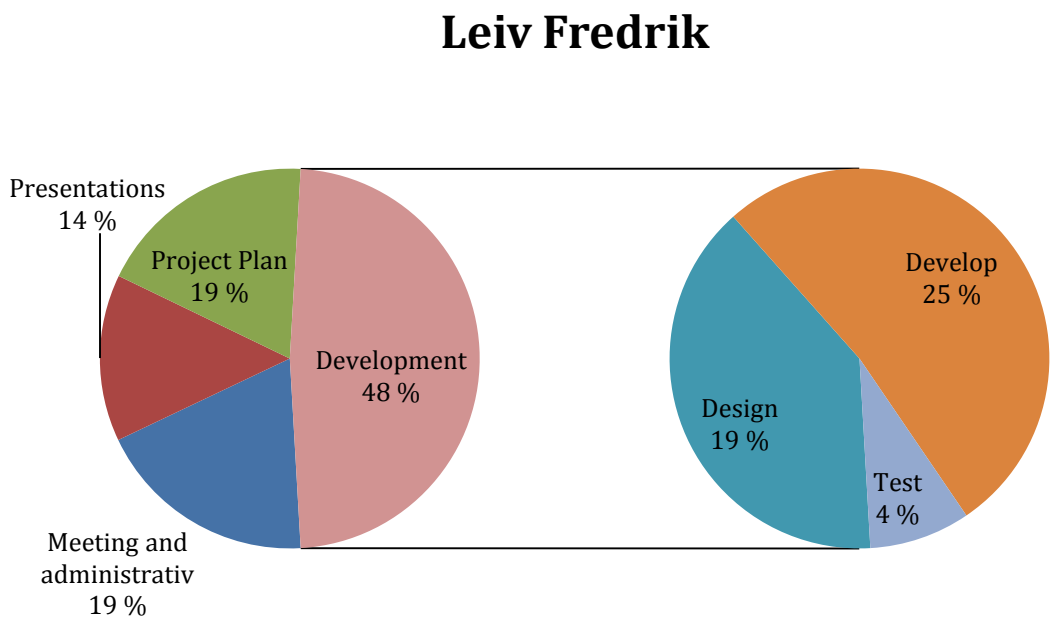

Figure 9: Leiv Fredrik's hours

Thomas

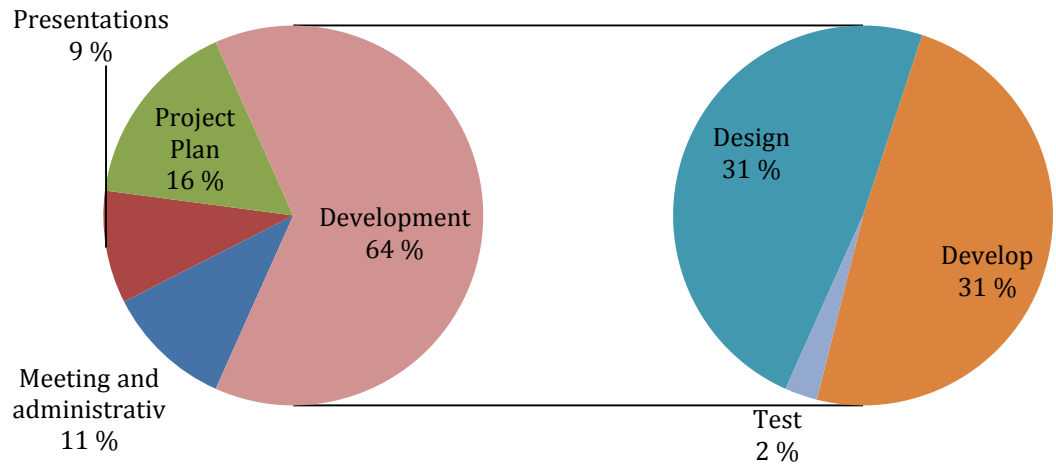

Figure 10: Thomas' hours

Morten

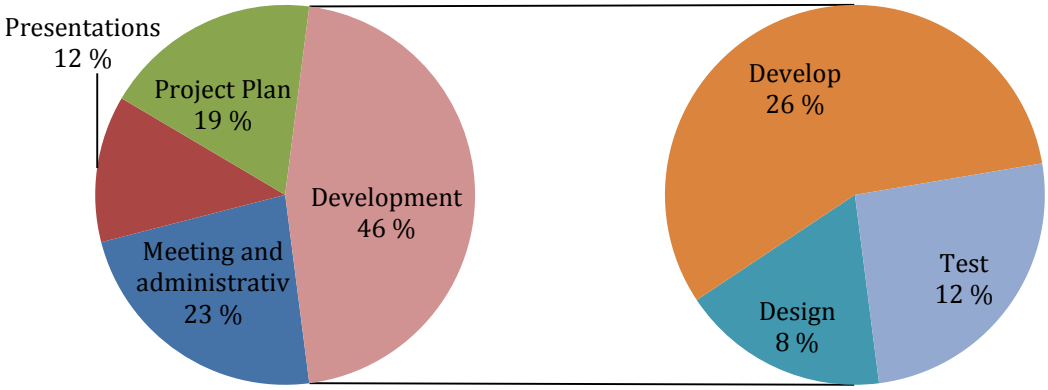

Figure 11: Morten's hours

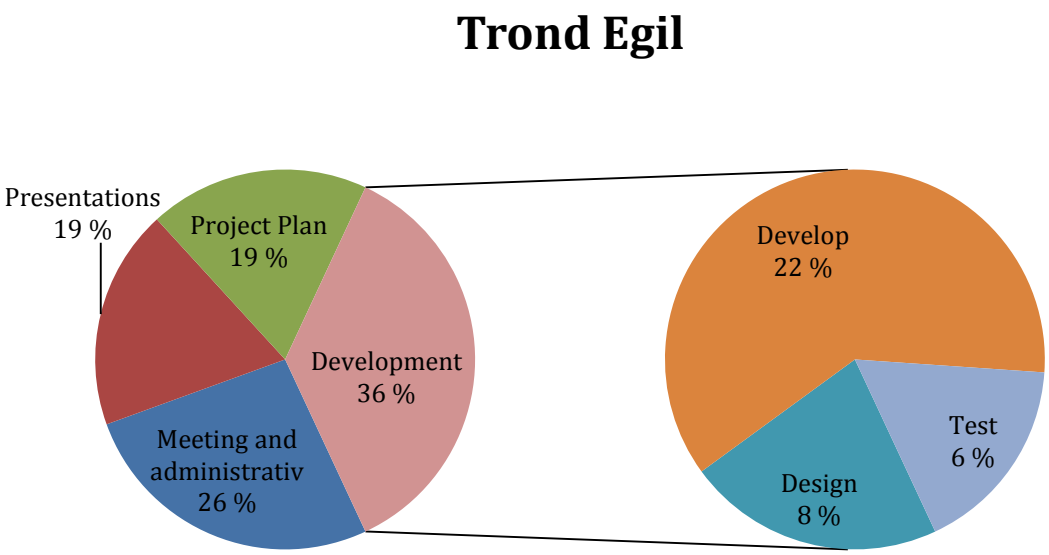

Figure 12: Trond Egil's hours

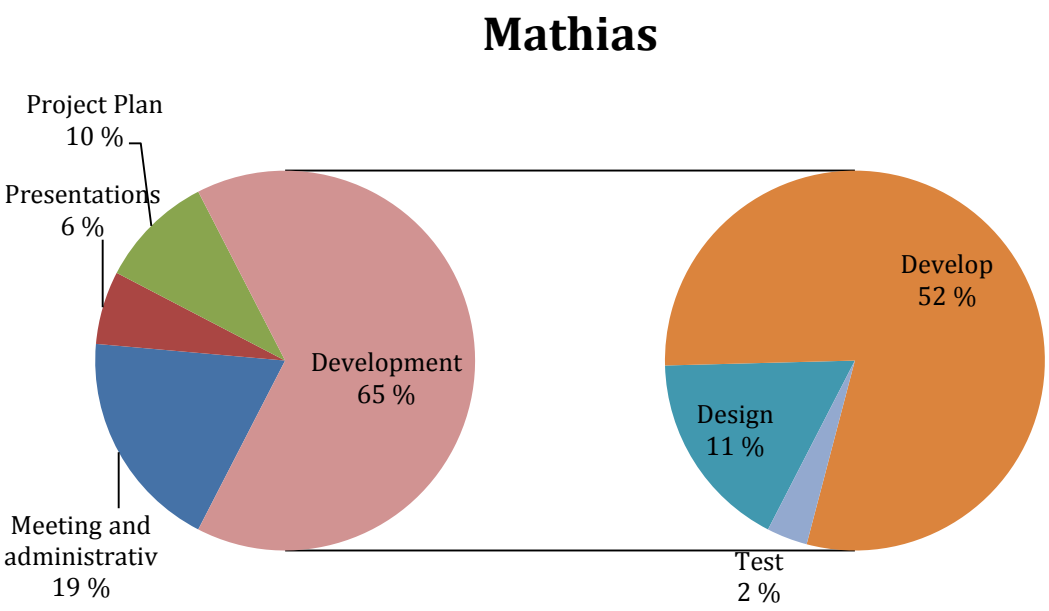

Figure 13: Mathias' hours

4. Project Challenges

We inherited a code base that was quite decently documented with comments in the code. However, it was deficient in regards to modelling and outside documentation. This meant it was hard for us to see what the earlier group had thought. In addition to the code, we also inherited a lot of hardware without specifications of what the components was intended to do. All this resulted in us having to redo a lot of the work that probably already had been done, but not documented. One example is the cameras we use in Project Argos. We had the cameras, but only one lens and no documentation on the lenses. This turned out to be a very complex topic. The way we use the cameras is very different from the normal use case of camera systems like ours. There are a huge number of factors that influences how a lens and camera will perform, and even though University College of Southeast Norway (HSN) has a great deal of expertise with optics, we still had a very complex job of dealing with the camera system and lenses.

The camera system produces an enormous amount of data. Each camera has a data rate of 128 MB/s, totaling more than 1 GB of data every two seconds from the camera rig. This puts a huge load on the infrastructure of the computer and the network. With the selection of components from the previous group we were limited in our ways of solving the issue. A number of adjustments to the camera setup and a network configuration with custom network drivers were required, but we eventually optimized the network performance to roughly 90%. However, not only is the network impacted by the data stream, but normal hard drives are nowhere near fast enough to handle 512 MB/s write speeds. Even normal SSDs will have trouble with that kind of a load. We continuously write to disk, but it’s not a single file, so that degrades the performance compared to the advertised speed of SSDs. We solved this by creating a three SSD array in RAID0 to increase the performance significantly.

In the middle of April, one of our cameras failed. We immediately contacted the vendor in Norway, and sent the camera in. The camera had to be sent to the manufacturer in Germany for repair, and this meant we had to reevaluate our solution for the final presentation and prepare for a presentation with only three cameras. Luckily, we did not have to alter too much of the code base, since we had already created a solution that could be customized with many different numbers of cameras and configurations. On Thursday 19.05.2016 we got back the broken camera from repair. So we will prepare the final presentation with all four cameras.

The eBUS SDK from Pleora we use to import the video frame buffers from the cameras crashed every time we deleted the reference to an old buffer. This consequently failed in 64-bit mode, but we did get it to work by creating a new 32-bit project and importing the old files. It was critical that we were able to fix this as the software would leak memory excessively and crash. The problem seems to be a bug in the SDK itself.

The Oculus Rift DK2 is a development kit without a stable and well documented development guide. The old code base used Oculus SDK 0.6.0.0 while the current SDK is 0.8.0.0, and the migration guides were not complete. We had to migrate 0.6 to 0.7 and then to 0.8. Along with this we also upgraded from Visual Studio 2013 to

2015 for support with the latest libraries. This should make the migration to Oculus Rift CV1 and SDK 1.3 much smoother for future groups.

5. Technical Assessment

We did not know what to expect when we started the development process. This was pretty much unchartered territory for KDS and the team. None of us had much experience with cameras, OpenGL or VR before working with Project Argos. We knew that there were going to be challenges along the way, especially with the camera integration, and VR and Oculus are still very immature. There are not that much documentation and guides to work with, so we had to spend time on learning the tools and OpenGL. The system is at the edge of what is possible to do with current hardware in regards to CPU usage, network throughput and disk read/write speeds.

The entire development process has been very challenging. The software we have created is quite advanced and complex, but our plans and ideas have pretty much all worked. Even when faced with unexpected bugs and performance issues our design and software architecture has proved stable and robust enough to handle the issues. During testing, the system has outperformed our and our stakeholder's expectations in terms of latency, efficiency, stability and functionality. We were even able to perform system acceptance test outside the lab in a vehicle, far exceeding our expectations when the project started.

6. Individual Reports

6.1 Mathias Havdal

I had the role of being the lead developer on this project. As one of the more experienced programmers in the group, this was very fitting. One of my responsibilities was planning development iterations. This proved to be a real challenge, because I had to figure out all the steps necessary to satisfy the iteration goals. Not only that, but I had to put the steps in the correct order and allocate an appropriate amount of time for each one. Fortunately my planning skills improved throughout the project, and none of my plans were overly optimistic. This meant that we were positively surprised to be ahead of schedule, and not the other way around.

As lead developer I also spent a lot of time writing code. This project had a lot of exciting technical challenges, and together with the rest of the team I was able to come up with technical solutions that worked very well. Since the codebase was my main responsibility, I had a hand in every software component that was developed.

One of the new challenges for me was learning to lead a team of developers. We all had different levels of programming experience, and I had to find fitting tasks to delegate to each group member. I also did a lot of work coordinating tasks being worked on in parallel. One of the most difficult things in a software project is to achieve increased productivity by adding more developers, without compromising on the stability and functionality of the software.

6.2 Morten J. Barbala

My role in the project was tester. It was my responsibility to design, implement and perform tests to make sure we verified and documented fulfillment of requirements for the system. I also had to observe tests and analyze results to discover problems and rooms for improvement. Working on project of this size, I quickly realized the importance of testing, and it was very satisfying to see the number of successful test logs grow as the system improved and became complete.

The main technical aspect I was responsible for in Project Argos was the playback module. I have gotten to solve difficult software challenges and experience programming in a project setting, producing code both alone and in pairs or groups. I had to learn about the Windows API for handling files and directories, both writing to and reading from disk. I needed to find effective ways of handling binary data of the large size that the cameras produced.

The bachelor project has been my absolute favorite time at HSN. We have gotten to use our programming and modeling knowledge to create an excellent system, but more importantly, we have created a very tight-knit team. Working close with people for hours every day could be very hard, especially since this is the most important project of our education so far, but I never felt tired or burned out. I could not have found better teammates.

6.3 Thomas Hansen

Project Argos was a scary project at the beginning, there was so much to do and so little time. The uncertainty was high, but the excitement was even higher. And my competent team gave me the confidence I needed.

As a last year computer science student I have been given the tools to manage any software challenge, and during this project there were a few challenges. Some of the challenges I had have been, understanding the code from the previous group and getting up in the morning. I had to learn a lot about computer graphics and software architecture while working on the project. I mostly worked on the architecture and a replacement for the graphical engine in the software and the coherent documents for these. As the designated architect of the project I have also been responsible for the abstract parts of the system and making technical decisions that can limit the design and implementation of the system.

The execution and implementation of this project have given me an insight in professional teamwork that I can take with me to my professional career. And of course I have learned the pleasures and terrors of working with others over a long time.

6.4 Trond Egil Hammer

To work on this project has been very fun and educational. I have learned how to work in a team and with a project over time. The team has worked very well together throughout this project. My role in this project has been as an analyst. As an analyst, I have been responsible for the customer and stakeholders needs as well as develop technical vision, system wide-requirements and use cases. Identify and outline the requirements has been one of my most important tasks. To ensure that the customer's needs are represented by the requirements. To perform these tasks I have used a good amount of knowledge from my time at HSN. I have also needed to learn about lenses and lens technology and been responsible for updating the webpage and developing the recording module. Most of the tasks that I were responsible for were finished early in the project, because my wife and I were expecting twins in April/May. The team knew this before we started this project, so I were lucky to get most of my tasks early in the process.
It has been a very fun experience to take part in this project. The assignment has been exciting and challenging. I am very proud of what the team has achieved. At the beginning of the project I didn't expect that we were able to drive a car with our software.

6.5 Ingvild Damtjernhaug

I was excited and a little frightened before we started this bachelor project; did the team have enough competence to reach the project goals? What would I be able to contribute with in this project? I thought the project sounded very interesting, with virtual reality and cutting edge technology, but I also knew this was going to be difficult. It was not sure we would be able to reach the goals.

I soon released it would be several topics to familiarize myself with, like VR and OpenGL.
My role as document manager have given me the opportunity to use my organizing skills, and my ability to keep the team on track and focused. I have been responsible for the configuration reader that reads the different XML config files. There have been times during the project where I felt overwhelmed, because of the high level of difficulty. But through a lot of work and support from good team members, I managed to solve the problems and gain both knowledge and self-esteem.
It has been nice to experience how the things I have learned during my time at HSN, merge together and become a whole. The subjects have all together given me a solid foundation, and the bachelor project has given me a change to use the knowledge I have gained.

The team-work could not have been better, and together as a team, we have managed to solve one problem at a time, overcoming challenge after challenge. I have mixed feelings about the fact that the project now comes to an end. I am relieved that we have reached our goals, and a little sad because I no longer will work together with my friends and fellow team members. But I'm most of all proud of what we have accomplished, proud of the team's effort and good work, and proud of myself for finishing three years at HSN in a good way. I now have better self-esteem and feel more prepared to enter work in «the real world».

6.6 Leiv Fredrik Berge

I'm proud over our system and project. We exceeded everyone's expectations and I'm very happy in the way we worked with Project Argos. As project manager I felt a lot of responsibility, both the entire team has impressed me with their skills and work ethics. This has made my job of driving the project forward much more pleasant. I think the most important thing I've done in Project Argos were to keep the big picture in mind and keeping track of the overall progress, to make sure we reached all our milestones.

It's been very exciting to work with new and emerging technology. I consider myself very lucky to have gotten the opportunity to work with such an interesting project. It has also been very challenging. I did not have any prior experience with OpenGL and VR. It's been a steep learning curve, but it's been nice to set the programming and modeling knowledge I've gained over the last couple of years out in a real world project.

7. Acknowledgements

We would like to extend our gratitude to the following people for their help and guidance.

- **Karoline Moholth** as internal sensor
- **Alexander Gosling** as external sensor and supervisor
- **Radmila Juric** as internal supervisor
- **Erik Torp** as project owner
- **Ellen Svarverud** as technical guide for visual perception
- **Fagskolen Tinius Olsen** for providing an electric car for the demo

Future Work 1.0

Created by: Leiv Fredrik Berge
05.05.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

Table of Contents

1. Document Overview

The purpose of the future work document is to show our proposal for future work needed in the system. It also documents the work that has been started, but not finished in the bachelor project.

Describes

- potential new functions in the system.
- potential improvements in the existing functionality.
- unfinished functionality.

1.1 Document History

Version	Change	Date	Created by
0.1	First version	05.05.2016	Leiv Fredrik Berge
0.2	Added text to 2.1,2.2,2.3	06.05.2016	Trond Egil Hammer
0.3	Added text to 4.1	09.05.2016	Thomas Hansen
0.4	Started on text 2.4 VR goggles	13.05.2016	Trond Egil Hammer
0.5	Added config creator, VR goggles	19.05.2016	Leiv Fredrik Berge
0.6	Corrections and clarifications, updated section 2-4, document history	20.05.2016	Ingvild Damtjernhaug
1.0	Final review	20.05.2016	Leiv Fredrik Berge Trond Egil Hammer Ingvild Damtjernhaug

1.2 Referenced Documents

Title	Document	Version
Glossary	doc-1113_glossary_2_0.xlsx	2.0

2. New Functionality

Project Argos is still in an early phase of development. It is expected that multiple project teams will add more functionality to the system. At the end of the bachelor project of 2016 all of the basic functionality for real-time virtual reality is present and working in the TinyArgos software.

2.1 Camera Rig

So far in the development process software has been the main area of focus. The camera rig we have created give the developers a stable platform to mount the cameras while in the lab. It is not meant to withstand the punishment it would receive mounted on a vehicle.

2.2 Interaction with other Systems

As of now the TinyArgos software do not interact with any outside systems. For future development there could be created interfaces that exchange data between multiple Project Argos equipped vehicles. Even other systems from Kongsberg can be integrated to provide data to the operator of a Project Argos system. For example an unmanned aerial vehicle (UAV), like LocalHawk, can provide information that can be displayed in the heads up display (HUD) in the Oculus Rift. Together with integration of sensors and other systems, it can give a better overview and situational awareness for the crew. Project Argos could merge data from the vehicle, internal and external sources. Information from these sub systems could be presented in real time on the screen. This could include moveable target indicators, terrain analysis and estimation of threats, reported with distance and other notifications.

2.3 Extra Cameras

By adding more cameras to the application it can ease the driving of the vehicle and give a better overview for the driver. When driving into small passages or reversing the vehicle, additional cameras can be helpful to get a better view of the surroundings. For example a camera that can show the edges of the vehicle or a camera inside the vehicle.

2.4 Virtual Reality Goggles

During our field test of Project Argos we discovered a drawback in the Oculus Rift. The positional tracker that tracks the movement of the Oculus Rift Dk2 is designed to be stationary, so it can correct the drift of the accelerometer and gyroscope in the headset. In our case, the tracker is placed inside a moving vehicle. When the tracker is placed on a moving ground, it fails to track the correct head movements. A discrepancy between the tracker and the headset then occurs. This results in the Oculus realigning itself in distinct jumps while the car is turning, instead of a smooth

rotational movement. To correct for this the tracker must probably also have an accelerometer and gyroscope to correct for the motion of the car, so it can act stationary in relation to the headset. Newer versions of the Oculus Rift or other head-mounted displays can potentially solve this problem.

2.5 Configuration Creator

The setup of the Oculus view is handled in the configuration files. These files contain all the information needed for distortion correction and placement of the video streams. However the creation of the XML files is not very intuitive, and could be made a lot easier with an application that would let you preview the settings, and then write the configuration file for you.

2.6 Passenger View

As of now the software displays the operator view in the Oculus, and a copy of that view on a monitor. The view in the monitor shows two barrel shaped images. This is not a pleasant way for a passenger to see the outside. So to improve the passenger experience multiple VR-headsets can be implemented or a monitor with a continuous, user controllable image.

3. Improvements to Existing Functionality

The third party software should be upgraded as they are updated with new releases. Also new hardware should continuously be evaluated to increase the TinyArgos performance. We have found some parts of the code inherited from the first group that needs some work. This is not critical errors, but rather parts of the code that should be improved to increase the reliability, stability and performance of the system. The following sections address those parts.

3.1 Improve the old Surface Objects

From TinyArgos 1.0 there was a system for creating HUD objects. The way they are created is quite convoluted and leaks memory when they are loaded into the scene. This means that when TinyArgos 2.0 enables multiple HUD configurations to be loaded during one session of use, the application will leak memory and eventually crash. In TinyArgos 2.0 there is a new way of creating HUD objects, but there is still support for older objects. For further development all the old classes should be replaced, and support for the older system should be broken.

3.2 Vulkan Graphics API

During the project the Khronos group released their new graphics application programming interface (API) Vulkan. Vulkan is the next generation of APIs created

for today's graphics cards. Vulkan supports multicore central processing units (CPU) and therefore can generate the work done by the graphics processing unit (GPU) in parallel using many CPU cores. This can possibly reduce the time it takes to send the texture data to the graphics card for rendering. This makes Vulkan a good candidate for consideration to be implemented into the graphics engine. This also requires some changes outside the virtual world component in the renderable class in the function that render the object.

4. Unfinished Functionality

This section specifies the work that has been started, but not considered stable or finished enough to be included in the default branch in the code base.

4.1 New Graphics Engine

The project is highly dependent on the highest possible throughput and a potential bottleneck that we could do something about was the graphics engine. We started to work on a new graphics engine with a goal that it would be more effective, more understandable and easier to continue to work on and refine at a later stage.

The vision was to create a simplified game engine that could accept geometrical shapes (surfaces) and textures for the graphics engine to render. The geometrical data would be created outside the engine and the textures would be fed by the cameras to the shapes that have to show video. The engine should also handle input from the user and VR googles.

The shapes needed for the scene can be loaded into a vector and added or removed to and from a rendering queue as needed. This gives us greater dynamic control over the scene as shapes can be added and removed when needed without reloading the whole scene. The code is written as understandable as possible so it should be easy to pick up and continue to work on it.

There is some work before the engine can be implemented into the working solution.

- The engine need some kind of VR system API implemented for use with a VR headset.
- The engine needs an input system that can handle the movement of the VR system and other inputs from the user. At the moment the input handling is done in the main class and it is just for the debug movement.
- The fragmentation and vertex shaders need to be changed a bit to handle the textures.

Technical Documentation
Network Solutions 2.0

Created by: Mathias Havdal
16.02.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

HSN

KONGSBERG

Table of Contents

1. Document Overview

The purpose of the network solution document is to give the reader a clear understanding of the challenges associated with the Local Area Network (LAN) that will be used in the Argos system. After reading this you should know the reasoning for the choice of network setup. A number of possible solutions are detailed and compared with one another in a Pugh matrix diagram.

Describes

- the challenges that the LAN is presenting to our system.
- possible network solutions
- the hardware and software already provided by Kongsberg Defence Systems
- the selection of network solutions.
- the comparison criteria used in the Pugh matrix
- a Pugh matrix that compares the solutions presented
- a conclusion with reasoning

1.1 Document History

Version	Changes	Date	Created by
0.1	Create document, add first draft of problem and possible solutions	12.02.2016	Mathias Havdal
0.2	Remove parts about DHCP	16.02.2016	Mathias Havdal
0.3	Corrections, rewriting	16.02.2016	Morten J. Barbala
0.4	Added content into correct template	16.02.2016	Trond Egil Hammer
0.5	Revised document layout. Added introduction, overview, pugh matrix	17.02.2016	Mathias Havdal
1.0	Added conclusion, descriptions and references	08.03.2016	Leiv Fredrik Berge
1.1	Changed format on name and date in document history	10.04.2016	Trond Egil Hammer
1.2	Fix layout, minor corrections	26.04.2016	Morten J. Barbala
1.3	Change reference style to IEEE 2006	28.04.2016	Morten J. Barbala
1.4	Minor changes	09.05.2016	Thomas Hansen
1.5	Minor changes	09.05.2016	Trond Egil Hammer
1.6	Deleted introduction, formatted front page, rewritten document overview	11.05.2016	Ingvild Damtjernhaug
1.7	Fix some issues and add a few comments. Made lots of improvements to writing (fix poor English).	12.05.2016	Mathias Havdal
1.8	Improve writing and technical correctness in some places	13.05.2016	Mathias Havdal
1.9	Add description of comparison	16.05.2016	Mathias Havdal

	criteria importance and rating scale. Rewrite conclusion with improved English.		
1.10	Fixed front page, headings and layout. Corrections and clarifications.	19.05.2016	Morten J. Barbala
2.0	Final review	20.05.2016	Trond Egil Hammer, Morten J. Barbala

1.2 Referenced Documents

Title	Document	Version
Glossary	doc-1113_glossary_2_0.docx	2.0

1.3 List of Tables

2. Challenges

Our system will use four cameras, each of which have the potential to saturate a 1Gbit/s link. The computer that runs the TinyArgos 2.0 software will need to receive data from all cameras simultaneously, without delay or loss. This means that the computer must have a theoretical receive rate of at least 4Gbit/s. There are numerous different combinations of software and hardware that can be used to solve this challenge. This document will cover the most relevant options.

3. Possible Solutions

Through research and analysing the current hardware and software on the system, we have found three solutions that we believe to be suited for transferring the live video from the Ethernet cameras to the Argos PC. The first two rely on the hardware that was provided for us at the start of the project, and the third explores the possibility of upgrading the network switch and network interface card for faster speeds and a more stable connection.

3.1 Existing Hardware/Software

Project Argos uses the hardware and software that was provided by Kongsberg Defence Systems (KDS) for the project. The latest drivers from Intel supports network interface card (NIC) teaming, however it does require a bit of configuration to enable the Intel driver to override the standard driver from Microsoft. NIC teaming is the term used for grouping several physical connections into one logical port. Under ideal circumstances this will lead to a significant increase in throughput.

With this solution each camera will saturate 80-90% of the available bandwidth on each physical link. If one of the physical ports starts slowing down or fails, the remaining ports will not have enough free capacity to offload the failing port. This means that the user experience will be interrupted by video stuttering and corruption.

Computer	**OS:**	Windows 7 Enterprise Service Pack 1
	NIC:	Intel Pro/1000 PT Quad Port LP (4x1Gbit)
Switch	**Make/Model:**	Unknown
	Ports:	24x1Gbit
	Capabilities:	Link aggregation, Power over Ethernet (PoE)

Pros	• No additional cost • Theoretical receive rate is 4Gbit/s
Cons	• Windows 7 does not have native support for NIC teaming/link aggregation

	○ Driver must have support ▪ Link aggregation between different NIC models likely not possible • Peak receive rate for a single connection is 1Gbit/s ○ 4Gbit/s only with ideal load balancing ○ Likely limited to four cameras even if system theoretically can support more

3.2 Proposal 1: Change OS

The second solution relies on swapping the operating system from Windows 7 to Windows Server 2008 or 2012. The server OS supports more control of the network cards and ports, which should make the network configuration smoother and perhaps more stable than Windows 7. This solution has the same limitation as the previous one, where if one port fails, there will not be enough bandwidth available to keep all four video streams running smoothly.

Computer	**OS:**	Windows Server 2008/2012
	NIC:	Intel Pro/1000 PT Quad Port LP (4x1Gbit)
Switch	**Make/Model:**	Unknown
	Ports:	24x1Gbit
	Capabilities:	Link aggregation, Power over Ethernet (PoE)

Pros	• Windows Server natively supports NIC teaming/link aggregation ○ Can connect to cameras over single virtual interface, with automatic load-balancing over the 4 NIC ports ○ System level aggregation rather than driver level ▪ Easier to integrate more cameras • Theoretical receive rate is 4Gbit/s (with existing hardware/software)
Cons	• Windows Server is unsuited for our software ○ Not intended to be used for 3D rendering and VR, or desktop applications in general ○ Will require significant configuration ○ No longer needed when network hardware is upgraded, possible waste of time and effort in a wider scope • Peak receive rate for a single connection is 1Gbit/s ○ 4Gbit/s only with ideal load balancing • Additional cost: Windows Server license

3.3 Proposal 2: Change Switch and NIC

The third solution is to upgrade the network hardware. Instead of using multiple 1 Gbit ports to connect the computer to the switch, we can use a single or dual port 10 Gbit NIC. This solution guarantees that we will achieve at least 4 Gbit throughput, and requires minimal software configuration. The obvious downside with this solution is the cost. 10 Gbit ports are only available on high-end networking hardware, and both the switch and the NIC will need to be replaced. Colfax, a leading global provider of computing systems [1], has a mid-range dual port 10 Gbit NIC priced at 395 USD [2], and the switch is likely to be even more expensive. Not only does the switch need to have one (preferably two) 10 Gbit ports, but it also needs to have at least 8 ports (preferably 16) with Power over Ethernet (PoE) capability and 1 Gbit speed for the cameras.

Computer	**OS:**	Windows 7
	NIC:	10Gbit
Switch	**Make/Model:**	(To be determined)
	Ports:	1x10Gbit, (8-16)x1Gbit
	Capabilities:	Power over Ethernet (PoE)

Pros	• Guaranteed receive rate of at least 4Gbit/s
	  ○ Connect to cameras over single interface
	• Minimal software configuration required
	  ○ Can keep current OS
	  ○ No need to configure NIC teaming/link aggregation

Cons	• Likely high cost

4. Pugh Matrix Comparison

The three possible solutions will be compared in a Pugh matrix. The rating and importance of each criterion is given on a scale of 1-3. The score is calculated by multiplying the rating with the importance of the criteria. Higher is better.

4.1 Comparison Criteria

Criteria	Importance	Rating
Ease of configuration	2 – This criterion represents the time and effort needed to get ideal performance from this hardware/software configuration. It has been given a medium importance rating, because time spent configuring hardware and software takes away	1 = Significant OS/network configuration 2 = Some OS/network configuration 3 = Minimal configuration (driver install at most)

	from time that could have been invested elsewhere.	
Scalability	2 – Scalability represents how many software/hardware changes would be needed to support more than the four cameras we intend to use. This bachelor project is part of a long term student project at Kongsberg Defence Systems (KDS) and for that reason this criteria has been given a medium importance rating.	1 = Significant software/hardware changes required to use more than four cameras 2 = Some software/hardware changes required to use more than four cameras 3 = No software/hardware changes required to use more than four cameras
Cost	3 – High-end network hardware can be very costly. For that reason, this criterion has been given the highest importance rating.	1 = Significant additional cost (likely more than 1000 USD) 2 = Limited additional cost (likely less than 1000 USD) 3 = No additional cost

Table 1: Comparison criteria for Pugh matrix

4.2 Pugh Matrix Proposals

		Options					
		Existing hardware		Proposal 1		Proposal 2	
Criteria	**Importance**	**Rating**	**Score**	**Rating**	**Score**	**Rating**	**Score**
Ease of configuration	2	2	4	1	2	3	6
Scalability	2	1	2	2	4	3	6
Cost	3	3	9	2	6	1	3
	Total		15		12		15

Table 2: Pugh matrix comparison

5. Conclusion

As we see from the Pugh matrix, the existing hardware/software and proposal 2 (upgrade network hardware) are the highest scoring solutions. If we look closely at the scores we can see that the network hardware upgrade is better on all criteria other than cost, but we intend to use the existing hardware for this project. In theory, we should not have any issues with bandwidth starvation, and only in the worst case scenario we might have to compromise slightly on the bandwidth settings for each of the cameras. This makes it hard to justify the increased cost for the hardware detailed in proposal 2. Proposal 2 is first and foremost about increasing headroom and scalability, which is not strictly necessary in the scope of this bachelor project. If the number of cameras is increased in the continuation after the bachelor project, proposal 2 should be reconsidered.

Bibliography

[1] "Colfax Direct," Colfax, [Online]. Available: http://www.colfaxdirect.com/store/pc/home.asp. [Accessed 08 05 2016].

[2] "Colfax Direct SFN7002F Dual-Port," Colfax, [Online]. Available: http://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=2131&idcategory=6. [Accessed 16 05 2016].

[3] Intel, "Intel PRO/1000 PT Quad Port Low Profile Server Adapter," Intel, 2016. [Online]. Available: http://www.intel.com/content/www/us/en/ethernet-products/gigabit-server-adapters/pro-1000-pt-qp.html. [Accessed 08 03 2016].

[4] Microsoft Corp, "Network adapter teaming and server clustering," Microsoft, 02 March 2010. [Online]. Available: http://support.microsoft.com/en-us/kb/254101. [Accessed 08 March 2016].

Technical Documentation
GigE Vision SDK 1.0

Created by: Morten J. Barbala
29.04.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

HSN

KONGSBERG

Table of Contents

1. Document Overview

The technical document for GigE Vision software development kit (SDK) gives the reader an overview of the GigE Vision standard and SDKs we could use to interface with our GigE Vision compliant cameras. The SDKs will be compared using a Pugh matrix, evaluating their strengths and weaknesses relative to our needs. After reading this document, the reader should know which SDK has been selected and why.

Describes

- overview of the GigE Vision standard.
- different SDK candidates.
- the comparison criteria used in the Pugh matrix.
- how the SDKs compare based on our needs.
- a conclusion with reasoning.

1.1 Document History

Version	Changes	Date	Created by
0.1	Create document, write about GigE Vision, eBUS, Active GigE and VIMBA	29.04.2016	Morten J. Barbala
0.2	Start adding Pugh matrix comparison criteria	11.05.2016	Mathias Havdal
0.3	Add Pugh matrix with importance and rating descriptions. Add conclusion.	12.05.2016	Mathias Havdal
0.4	Fixed layout. Minor corrections	13.05.2016	Morten J. Barbala
1.0	Final review	20.05.2016	Trond Egil Hammer, Thomas Hansen, Morten J. Barbala

1.2 Referenced Documents

Title	**Document**	**Version**
Glossary	doc-1113_glossary_2_0.docx	2.0

1.3 List of Tables

2. GigE Vision

GigE Vision is an interface standard for high-performance video transfer and device control over Ethernet networks. It was ratified in May 2006 and the latest revision, 2.0 in November 2011 [1], included control for non-streaming devices, support for transfer speeds with 10 gigabit Ethernet, and link aggregation. GigE Vision 2.0 also makes it possible to transmit compressed images like JPEG and H.264, and has improved accuracy in synchronization of systems with multiple cameras.

The GigE Vision standard uses GenICam [2] to describe the features available on the cameras in the form of XML description files. GenICam is a programming interface used for all types of cameras, and serves as a base for “plug and play” handling of devices and cameras with any interface technology (GigE, USB3, Camera Link etc.). Both GigE Vision and GenICam are protected standards and have committees, which only allow vendors to brand compliant products.

3. GigE Vision SDK

An SDK exposes the underlying application program interface (API) to easier utilize the functionalities of the GigE Vision standard. There are a number of manufactures that create their own SDKs. This means that multiple SDKs for the same GigE Vision system can have different functionalities and quality. The choice of SDK will impact our systems performance and features.

3.1 eBUS SDK

The eBUS SDK is developed by Pleora Technologies for use in high-performance digital video systems [3]. It enables developers to create GigE Vision standard-compliant video applications for both Windows and Linux through a modular architecture.

3.2 Active GigE

Active GigE is a GigE vision driver as well as hardware-independent SDK. Any GigE Vision compliant device works out-of-the box with a wide selection of integrated development environments (IDE) natively supported. Using an ActiveGigE object, the application can instantly support multiple GigE Vision cameras, even live video directly in Microsoft Word or PowerPoint. [4]

3.3 Vimba

Vimba is a vendor-specific SDK designed for Allied Vision cameras. It is, however, platform-independent, i.e. works on any operating system, and has APIs for C, C++ and .NET. It is based on GenICam and comes with license for all Allied Vision cameras free of charge.

4. Pugh Matrix Comparison

The possible solutions will be compared in a Pugh Matrix. The rating and importance of each criterion is given on a scale of 1-3. The score is calculated by multiplying the rating with the importance of the criteria. Higher is better.

4.1 Comparison Criteria

Criteria	Importance	Rating
Documentation	3 – Good documentation is essential when working with an unfamiliar SDK	1 = No quickstart guide/limited API function documentation 2 = Quickstart guide or full API function documentation 3 = Quickstart guide AND full API function documentation
Camera Support	3 – Having support for a wide range of cameras is important to ensure the longevity of the software	1 = Only cameras from this vendor 2 = Limited selection from other vendors 3 = Full GigE Vision support
Cost	1 – The cost of the SDK is not a significant obstacle for our project owner	1 = More than 1000 NOK and/or recurring fee 2 = Affordable one time purchase (less than 1000 NOK) 3 = Free
OS Support	2 – Our target platform is Windows 7, but having support for other platforms gives more options in the future	1 = Windows 7 support 2 = Support for newer Windows versions 3 = Full Windows and Linux support
Demo Performance	2 – If the SDK has a demo, it can give us an idea about the performance we can expect when implementing it in our own software.	1 = No demo available/demo performs poorly 2 = Demo performs well after configuration adjustments 3 = Demo performs well out of the box

Table 1: Comparison criteria for Pugh matrix

4.2 GigE Vision SDK Pugh Matrix

		Options					
		eBUS		Vimba		Active GigE	
Criteria	**Importance**	**Rating**	**Score**	**Rating**	**Score**	**Rating**	**Score**
Documentation	3	3	9	3	9	3	9
Camera Support	3	3	9	1	3	3	9
Cost	1	2	2	3	3	1	1
OS Support	2	3	6	3	6	2	4
Demo Performance	2	2	4	3	6	1	2
	Total	30		27		25	

Table 2: Pugh Matrix comparison

5. Conclusion

The results from the Pugh matrix show that the eBUS SDK is the best option. All three SDKs have good documentation, but the eBUS SDK wins on having full GigE Vision support and good OS support (several versions of Windows and multiple Linux distros). We will be developing for Windows 7, but using an SDK with support for a broad range of operating systems will provide more options in the future.

The eBUS SDK also has a demo that performed well after adjusting the maximum transmission unit (MTU) on the network interface we were using to connect to the cameras. This gives us a clear indication that eBUS can provide the performance we need if we were to implement it in our own software. Vimba performed well with no configuration changes. This is perhaps a result of Allied Vision making optimizations specific to our cameras (made by Allied Vision). Active GigE did not run.

As for cost, the eBUS SDK uses a per-machine licensing system. A permanent license for the computer that will run our software would cost 850 SEK. The Active GigE SDK is significantly more expensive, with a single developer license costing nearly 500 USD [5]. Vimba on the other hand, requires no license purchase. This is because we are using Allied Vision cameras. Obviously at 850 SEK for the eBUS license, cost is not enough to swing our decision in favour of Vimba.

In conclusion, we believe the eBUS SDK to be the best option, so that is what we intend to use.

Bibliography

[1] “GigE Vision Main Page,” Vision Technology, [Online]. Available: http://visiononline.org/vision-standards-details.cfm. [Accessed 29 April 2016].

[2] “GenICam - EMVA,” [Online]. Available: http://www.emva.org/standards-technology/genicam/. [Accessed 29 April 2016].

[3] “eBUS SDK,” [Online]. Available: http://www.pleora.com/our-products/ebus-sdk. [Accessed 29 April 2016].

[4] “GigE Vision camera SDK,” [Online]. Available: http://www.ab-soft.com/activegige.php. [Accessed 29 April 2016].

[5] “A&B software,” [Online]. Available: http://www.ab-soft.com/activegigestore.php. [Accessed 12 05 2016].

Technical Documentation
Virtual Reality Goggles 3.0

Created by: Ingvild Damtjernhaug
16.02.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

KONGSBERG

Table of Contents

1. Document Overview

The technical document for the virtual reality goggles (VR goggles) contains the technical documentations on different types of VR goggles, and justification for the choice of Oculus Rift Development Kit 2 (DK2). This document serves also as help for further project groups, and shows how we were thinking according to the information and solutions available. This document will cover the most relevant options for Project Argos.

Describes

- briefly the background, why this document is needed.
- possible alternatives.
- the reasoning for the choice of Oculus Rift DK2.
- the comparison criteria used in the Pugh matrix.
- a Pugh Matrix that compares the alternatives presented.
- a conclusion with reasoning.

1.1 Document History

Version	Changes	Date	Created by
0.1	First version	16.02.2016	Ingvild Damtjernhaug
0.2	Added Pugh Matrix	20.02.2016	Ingvild Damtjernhaug
0.3	Minor fixes	03.03.2016	Leiv Fredrik Berge and Ingvild Damtjernhaug
1.0	Printed	08.03.2016	Ingvild Damtjernhaug
1.1	Added description and references	08.03.2016	Leiv Fredrik Berge
2.0	Added conclusion	11.03.2016	Ingvild Damtjernhaug
2.1	Changed format on name and date in document history	10.04.2016	Trond Egil Hammer
2.2	Fixed layout, corrections and clarifications	26.04.2016	Morten J. Barbala
2.3	Updated references and document overview, deleted introduction, added pictures, formatted front page	05.05.2016	Ingvild Damtjernhaug
2.4	Added Pugh matrix comparison criteria	16.05.2016	Leiv Fredrik Berge
2.5	Updated doc.overview, proofreading with corrections	19.05.2016	Ingvild Damtjerhaug
3.0	Final review	20.05.2016	Thomas Hansen Morten J. Barbala Ingvild Damtjernhaug

1.2 Referenced Documents

Title	Document	Version
Requirements Document	doc-1213_requirements_2_0	2.0
Glossary	doc-1113_glossary_2_0	2.0

1.3 List of Figures

1.4 List of Tables

2. Background

The VR goggles are a key component in Project Argos (see the Requirements document). By “VR goggles” we mean a virtual reality head-mounted display, where your head movements are tracked in a three-dimensional world. We inherited certain design decisions from an early stage of Project Argos. When the project started up during the summer of 2015, the Oculus Rift DK 2 was the only actual option. Oculus Rift DK2 was selected as every other solution either seemed to be either incomplete or unproven. The team working on Project Argos summer 2015 assumed better goggles might be on the market in 2016.

3. Alternatives

At the time being, several companies are said to be releasing their VR goggles early in 2016. There are already several virtual reality goggles on the market, but many of them are not relevant for our project as we need high-end goggles. Release dates are also an issue. Mid-2016 seems to be the time where many companies release their first consumer version of VR goggles. The following are the goggles we consider to be most relevant.

3.1 Oculus Rift DK2

Oculus Rift is developed by Palmer Luckey, founded via Kickstarter and later bought by Facebook. Oculus Rift DK2 was released in July 2014. DK2 is the successor of the DK1, both of which are development kits for very early adopters, content creators and developers. DK2 is not finalized hardware, but should give developers a platform to create their applications for the first consumer version (CV1) when that is released.

Figure 1: Oculus Rift DK2

Display: OLED panel
Resolution: 1920 X 1080 HD (960 x 1080 display for each eye)
Refresh rate: 75 Hz
Field of view: 100 degrees
Price: not relevant
Consumer Release: July 2014.

Additional information: Does not support 360 degrees tracking. DK2 is not a consumer version, but meant for developers to create VR programs in. [1]

3.2 Oculus Rift CV1

The Oculus Rift CV1 is the consumer version that ships in March 2016. This is the finalized hardware with the lessons learned from DK1 and DK2.

Figure 2: Oculus Rift CV1

Display: OLED panel
Resolution: 2160 x 1200
Refresh rate: 90 Hz.
Field of view: 110 degrees
Tracking area: not relevant
Price: 499-599$
Consumer Release: March 28th 2016.

Additional information: Separate lenses, that are adjustable by a dial on the bottom of the device, so they will fit any user (accommodates a wide range of inter pupillary distances). It gives 360 degrees tracking by LEDs in the rear of the goggles, meaning full 360 degrees perspective for the users. The CV1 has more accurate and improved head tracking than DK2. [2]

3.3 Sony PlayStation VR

Sony's VR goggles are based on the project Morpheus. These VR goggles are meant to work on the PlayStation platform, with quite severe hardware limitations over the Rift and the Vive that is PC-based.

Figure 3: Sony Playstation VR

Display: OLED panel
Resolution: 1920 x 1080
Refresh rate: 90 Hz (interpolated)
Field of view: Approximately 110 degrees
Tracking area: not relevant
Price: TBA
Consumer Release: 2016
Additional information: Works only with PlayStation. [3] [4]

3.4 HTC Vive

Vive is HTC's VR goggles. These are developed with Valve, which might indicate that it will support Linux-based operating systems, but this is not confirmed at the time being.

Figure 4: HTC Vive

Display: OLED
Resolution: 2160 x 1200
Refresh rate: 90 Hz
Field of view: 110 degrees
Tracking area: 15 x 15 feet
Price: 799$
Consumer Release: April 2016
Additional information: Tracks motion in space, but this is not relevant sitting inside a car. [2] [5]

4. Pugh Matrix Comparison

The possible solutions will be compared in a Pugh matrix. The rating and importance of each criterion is given on a scale of 1-3. The score is calculated by multiplying the rating with the importance of the criteria. Higher is better.

4.1 Comparison Criteria

Criteria	Importance	Rating
Resolution	2 – A high resolution is essential to create an immersive virtual world	1 = < 1080p 2 = 1080p 3 = > 1080p
Framerate	2 – The framerate must be high to reduce chances of motion sickness and a fluid experience	1 = < 70 FPS 2 = 70 – 80 FPS 3 = > 80 FPS
Availability	3 – Available for purchase	1 = Available summer of 2016 2 = Available spring of 2016 3 = In possession
SDK availability	3 – Available SDK for download	1 = Available summer of 2016 2 = Available spring of 2016 3 = In possession
Cost	1 – The cost is not significant	1 = > $700 2 = $500 - $700 3 = < $500

Table 1: Comparison criteria for Pugh matrix

4.2 VR Goggles Pugh Matrix

		Options							
		Oculus Rift DK2		Oculus Rift CV1		HTC Vive Pre		Sony PS VR	
Criteria	**Weight**	**Rating**	**Score**	**Rating**	**Score**	**Rating**	**Score**	**Rating**	**Score**
Resolution	2	2	4	3	6	3	6	2	4
Framerate	2	1	2	3	6	3	6	1	2
Availability	3	3	9	2	6	2	6	1	3
SDK availability	3	3	9	2	6	1	3	1	3
Cost	1	3	3	2	2	1	1	-	-
	Total		27		26		22		12

Table 2: Pugh matrix comparison

The comparison is primarily based on information available at the time of writing. This is still in an early development stage, and the Vive, Sony VR and Rift CV1 are yet to be launched at the time of making the decision. The values in the Pugh matrix, and consequently the result, will change as the first consumer version of the goggles hit the market.

5. Conclusion

The reason why Oculus Rift DK2 is the winner comes down to availability. Based on the results from the Pugh matrix comparison in section 3, we have decided to stick to DK2. We believe it will satisfy our needs for this part of the project as both the software and the hardware are available and stable. CV1 and HTC Vive look promising and this decision should be revisited by the next Argos team after the launch of CV1 and Vive. The code should in theory support CV1 with an upgrade to the latest SDK. Oculus provides migration guides from SDK version to SDK version, so the upgrade should be manageable.

Bibliography

[1] Rift Info, "Oculus Rift Specs - DK1 vs DK2," Rift Info, 06 01 2016. [Online]. Available: http://riftinfo.com/oculus-rift/specs-dk1-vs-dk2-comparison. [Accessed 08 03 2016].

[2] Digital Trends, "Spec Comparison," Digital Trends, 29 02 2016. [Online]. Available: http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive. [Accessed 08 03 2016].

[3] Digital Trends, "Oculus Rift vs Playstation VR Spec comparison," Digital Trends, 06 01 2016. [Online]. Available: http://www.digitaltrends.com/computing/oculus-rift-vs-playstation-vr. [Accessed 08 03 2016].

[4] "PlaystationVR," [Online]. Available: http://www.playstation.com. [Accessed 23 March 2016].

[5] "HTC Vive Pre," [Online]. Available: http://htcvive.com. [Accessed 24 March 2016].

Technical Documentation
Graphics API 1.0

Author: Thomas Hansen
31.04.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

HSN

KONGSBERG

Table of Contents

1. Document Overview

The purpose of the technical documentation on graphics application programming interface (API) is to show an overview of different graphics APIs. Project Argos needs to use a graphics API in its virtual world module to render images to the virtual reality goggles and to the screen.

Describes

- the context.
- what graphics APIs are.
- three alternatives for graphics APIs.
- a conclusion based on the information provided.

1.1 Document History

Version	Changes	Date	Author
0.1	Document started	31.04.2016	Thomas Hansen
0.2	Revised	05.05.2016	Thomas Hansen
0.3	Updated document overview	18.05.2016	Trond Egil Hammer
0.4	Changed 4.1 to new standard, removed 1.3 references	18.05.2016	Thomas Hansen
0.5	Updated document overview, revised section 2, added section 4, added comments, formatted,	18.05.2016	Ingvild Damtjernhaug
0.6	Added conclusion, rewritten OpenGL, Direct3D and Vulkan	18.05.2016	Leiv Fredrik Berge
0.7	Fixed references and more content	20.05.2016	Thomas Hansen
1.0	Final review	20.05.2016	Trond Egil Hammer, Morten J. Barbala, Thomas Hansen Ingvild Damtjernhaug

1.2 Referenced Documents

Title	Document	Version
Motion Sickness	doc-21321_motion_sickness_1_0.docx	1.0
Glossary	doc-1113_glossary_2_0.docx	2.0

1.3 List of Figures

2. Context

TinyArgos, the software in Project Argos, creates a virtual world that is displayed in the Oculus Rift. This needs to be done as efficiently as possible to reduce the HMD latency and the glass-to-glass latency, see Motion Sickness for more details. The graphics card (GPU) is the most efficient way to do the same operation on large amounts of similar data, which is what TinyArgos does with rendering and transformation of the video. In order to control the use of the GPU in computation we need an API to expose the functions to our application.

3. Graphics API

The virtual world module in the TinyArgos architecture relies on a graphics API to expose the functions we need to render the live video and surfaces. The graphics API functions as a layer between our application and the operating system, making it more efficient to utilize the hardware, in our case the GPU. There are different graphics APIs available with different strengths and weaknesses. However, not all fit our need for the type of application we are developing

The graphics API exposes hardware functionality. The developers can use a high level language to write code that takes advantage of the GPUs features. The API translates the high level input from the application to low level code for the GPU driver, which turns the code into machine instructions the GPU can execute. This means that the graphics driver and graphics API must be optimized for each other to get maximum performance of the system. [1]

4. Different Types of Graphics API

On the marked there are multiple graphics APIs that could work with TinyArgos. It needs to support virtual reality, the Oculus SDK, be very efficient and preferably be multiplatform. The main contenders in the graphics API marked are Direct3D by Microsoft, OpenGL by Silicon Graphics and Vulkan by Khronos group. All of these could potentially work with TinyArgos.

4.1 OpenGL

OpenGL (Open Graphics Library) is a 2D and 3D graphics API that was released in 1992 by Silicon Graphics and is now curated by Khronos Group. Khronos is an independent non-profit consortium of leading media centric companies founded in January 2000. [2] OpenGL is an open standard, so it is free for anyone to use and implement. The latest release of OpenGL is 4.5 from August 2014. [3] The library can be used with any operating system and with any programming language. The API is extremely versatile, and will run on most systems. This makes it easier to change operating system or programming language for TinyArgos if needed. OpenGL was released in 1992 and therefore was the first graphics API available for developers. [3]

[4] Since 1992 OpenGL has become the industry's most widely used API. [5] The first version of TinyArgos was built with OpenGL as its graphical API.

The OpenGL rendering pipeline is the sequence of steps that OpenGL takes when rendering objects. [6] It is this pipeline that together with the graphics driver determines how effective the API is.

The OpenGL pipeline:

- Vertex specification [7]
  - The application sets up an ordered lists of vertices that are sent into the pipeline when a render command is issued [7]
- Vertex shader
  - The vertex shader performs basic processing of each individual vertex. [7] Like the transformation of the vertex in the OpenGL world. [6]
- Tessellation
  - Determines the amount of tessellation to apply on a primitive. [8]
- Geometry shader
  - Creates the primitives from points.
- Transform feedback
  - Allows for holding data for later use [7]
- Clipping
  - This cuts the primitives that lies on the border of the viewing area into several primitives. And removes the primitives outside the view area. [7]
- Face culling
  - Culling is the process of excluding the primitive faces that do not face the window from rendering. [7]
- Rasterization
  - This is the process of turning the primitives into fragments containing a screen position and [7]
- Fragment shader
  - Here the fragments are given colour and are sent to the next stage. [7]
- Per-Sample Operations
  - Each fragment from the fragment shader is then put through a series of tests before being written to a frame buffer. [7]

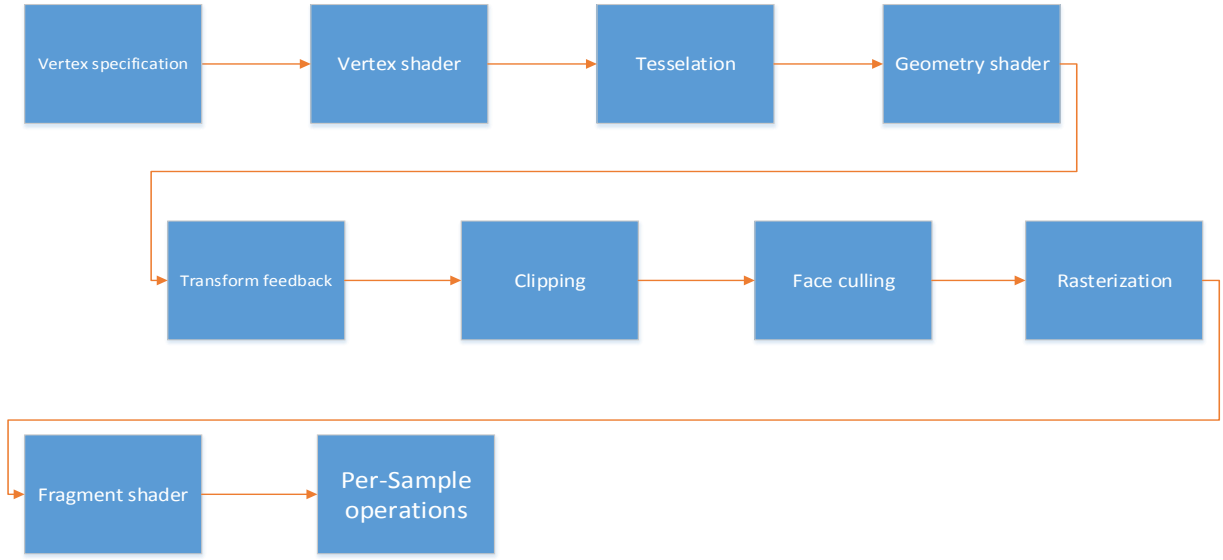

Figure 1: The OpenGL pipeline

4.2 Direct3D

Direct3D is the graphical API in the DirectX collection by Microsoft. Driect3D first appeared in 1996 in DirectX 2.0. [4] The latest version Direct3D 12 (DX12) was released July of 2015 together with Windows 10. [9] The DirectX suite only supports the operating system Microsoft Windows, making a port of TinyArgos to a another platform at some point much harder. It does support a number of programming languages, but it is not as versatile as OpenGL. Direct3D 12 reduces the load on the CPU, improves the multithreading scaling and allows for a lower level of hardware abstraction than DX11 did. [10] Direct3D is an advanced graphics API it requires a fine level of tuning and significant graphics expertise. Because it is designed to make full use of multi-threading, a considerable amount of programing skills on memory level are also required. [10] Direct3D 12 represents a significant departure from the Direct3D 11 programming model. It lets you get closer to hardware than ever before, but the trade-off is that you are responsible for more tasks yourself. [11]

The Direct3D rendering pipeline looks somewhat like the OpenGL pipeline but is a bit different. And again, the pipeline works together with the graphics driver to determine how efficient it is. [12]

The Direct3D pipeline:

- Input assembler
  - Supplies vertex data to the pipeline [12]
- Vertex shader
  - The vertex shader performs basic processing of each individual vertex. Like the transformation of the vertex in the OpenGL world [12]
- Hull shader
  - A programmable stage that generate patches [12]
- Tessellation
  - Uses GPU to calculate a more detailed surface from patches [12]

- Domain shader
  - A programmable shader stage that calculates the vertex position that corresponds to each domain sample. [12]
- Geometry shader
  - Creates primitives from points. [12]
- Rasterizer
  - The rasterizer clips primitives prepare primitives for the pixel shader, and determines how to invoke the pixel shader. [12]
- Pixel shader
  - Generates per-pixel data such as colour [12]
- Output merger
  - Combines the output data with the contents of the render target. [12]

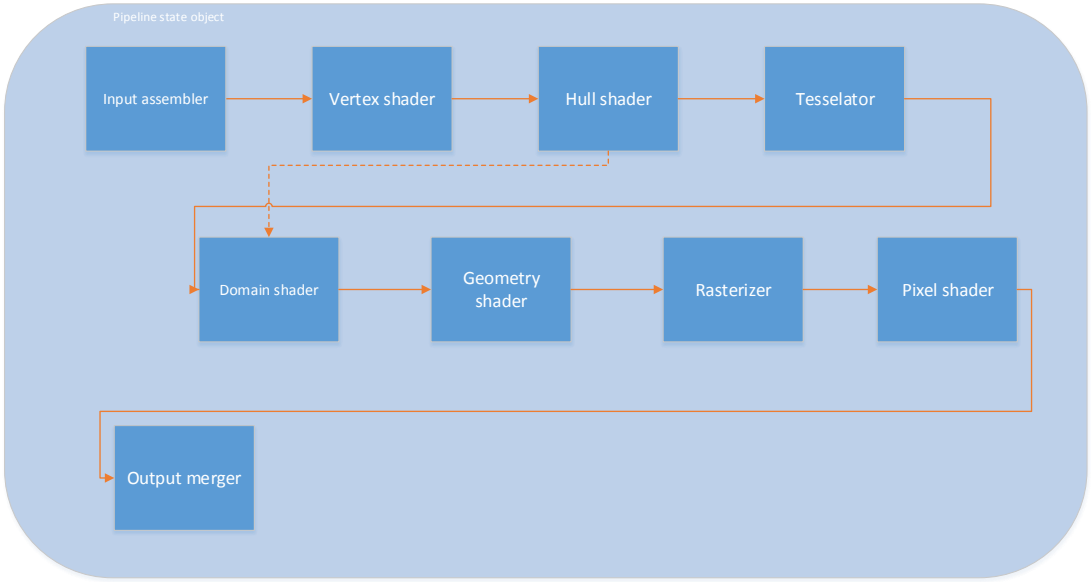

Figure 2: The direct3D 12 pipeline

Direct3D 12 introduces the pipeline state object (PSO). This object stores the pipeline state in a PSO instead of across a large number of high-level objects.

4.3 Vulkan

Vulkan is created by Khronos Group, the same group that curates OpenGL. Initially Vulkan were marketed as the next generation of OpenGL. [13] The 1.0 version was released in February of 2016 [14], with 1.0.13 as the latest release as of 18.05.2016. The advantage Vulkan has over OpenGL is the ability to generate GPU work in parallel using several central processing units (CPU) cores. Along with other improvements, like reduced driver overhead, Vulkan has less CPU usage than OpenGL. This can make Vulkan very useful for games, simulations and CPU bound applications. [15]

If the GPU workload dominates the rendering time in the application, [16] Vulkan might not offer any speed up to the software. On the other hand, if the application is sensitive to missed frames or micro stuttering, Vulkan's design allows you to control explicitly when expensive operations happens during rendering. [16] If the

applications work load is spread across multiple threads, Vulkan can improve the situation. [16]

The Vulkan pipeline:

- Input assembler
  - Assembles vertices to form geometric primitives [17]
- Vertex shader
  - Computes position and other attributes for each vertex [17]
- Tessellation shader
  - Calculates a more detailed surface [17]
- Geometry shader
  - Generates primitives [17]
- Primitive assembler
  - The primitives are clipped to a clip volume (primitives outside the view are excluded) [17]
- Rasterization
  - Produces a series of framebuffer addresses and values using a two-dimensional description of a point, line or triangle. [17]
- Fragment shader
  - Performs operations on individual fragments before they alter the framebuffer. [17]

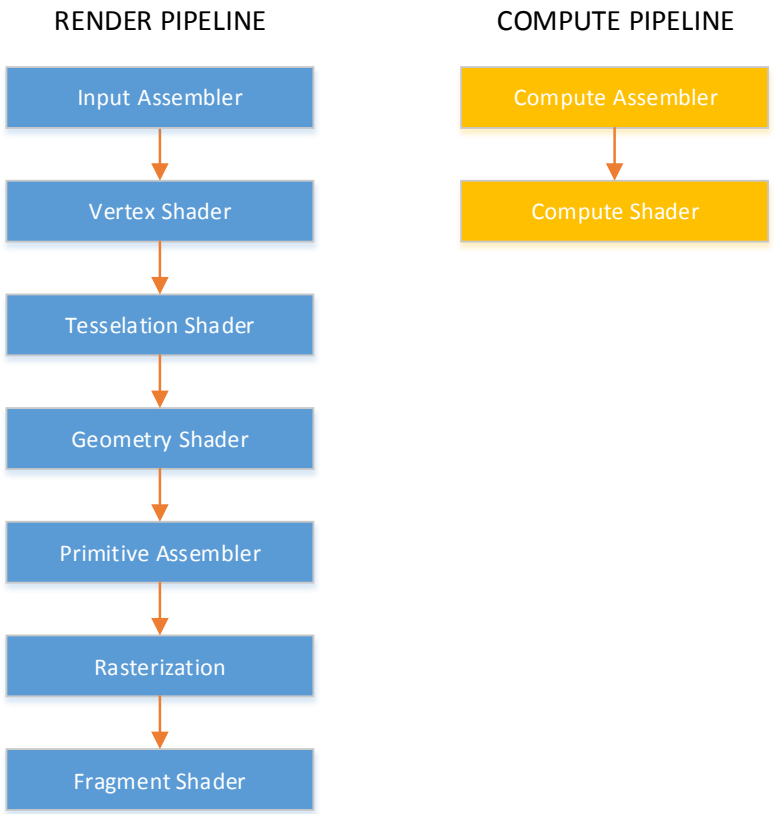

Figure 3: The Vulkan pipeline

The compute pipeline is a separate pipeline from the graphics pipeline, which operates on one-, two- or three dimensional workgroups which can read and write to a buffer and image. [17]

5. Conclusion

It is hard to quantify the differences in the different graphical APIs, much will depend on other factors in the system than just the API. E.g. Nvidia has yet to upgrade their drivers to fully utilize the potential performance increase in DX12, and often perform better with DX11, while AMD cards can have significantly better performance. [18]

With the load that TinyArgos puts on the system, it is hard to say what kind of performance can be expected with the various API and hardware combinations. The PC that will run the TinyArgos software has an Nvidia Titan X graphics card, making DX12 the least favoured option, with poor DX12 driver support and no other operating system support in the API.

Both DX12 and Vulkan are quite immature as driver and operating system support is not all there yet. However this is likely to change at some point, making both DX12 and Vulkan more appealing options. Especially the superior support for multithread scaling and lower driver overhead can potentially increase the performance and stability of TinyArgos.

Given OpenGL's excellent track record as a multiplatform and good performing graphics API, we find it to be the better choice for TinyArgos at this point in time. It might be worth revisiting this decision as more benchmarks and information about Vulkan becomes available, as it might do a slightly better job at lowering the glass-to-glass latency.

Bibliography

[1] L.-Y. Wei, “http://graphics.stanford.edu,” [Online]. Available: http://graphics.stanford.edu/~liyiwei/courses/GPU/paper/paper.pdf. [Accessed 05 May 2016].

[2] “Khronos,” Khronos, 01 2000. [Online]. Available: http://www.khronos.org/about. [Accessed 19 05 2016].

[3] “OpenGL,” 05 04 2016. [Online]. Available: http://www.opengl.org/wiki/History_of_OpenGL. [Accessed 19 05 2016].

[4] W. F. Engel, Direct 3D Game Programming 2nd Edition, Boston: Premier Press, 2003.

[5] Khronos, “OpenGL,” Khronos, [Online]. Available: http://www.opengl.org/about. [Accessed 19 05 2016].

[6] G. Sellers, R. S. Wright jr. and N. Haemel, OpenGL SuperBible Sixth Edition, Crawfordsville: RR Donnelley, 2013.

[7] “OpenGL,” OpenGL, [Online]. Available: http://www.opengl.org/wiki/Rendering_Pipeline_Overview. [Accessed 19 05 2016].

[8] “OpenGL,” [Online]. Available: http://www.opengl.org/wiki/Tesselation. [Accessed 19 05 2016].

[9] B. Langley, “Microsoft developer,” Microsoft, 29 07 2015. [Online]. Available: http://blogs.msdn.microsoft.com/directx/2015/07/29/windows-10-and-directx-12-released/. [Accessed 19 05 2016].

[10] “microsoft.com,” Microsoft, 2015. [Online]. Available: http://msdn.microsoft,com/en-us/library/windows/desktop/dn899288(v=vs.85).aspx. [Accessed 19 05 2016].

[11] “Microsoft.com,” Microsoft, [Online]. Available: http://msdn.microsoft,com/en-us/library/windows/desktop/dn899288(v=vs.85).aspx. [Accessed 19 05 2016].

[12] “microsoft.com,” Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/hardware/ff569022(v=vs.85).aspx. [Accessed 19 05 2016].

[13] “amd.com,” AMD, 2016. [Online]. Available: support.amd.com/en-us/kb-articles/Pages/AMD_Radeon_Software_Crimson_Edition_16.3.aspx. [Accessed 19 05 2016].

[14] Khronos, “Khronos.org,” 2016. [Online]. Available: http://www.khronos.org/registry/vulkan/specs/1.0/pdf/vkspec.pdf. [Accessed 19 05 2016].

[15] “Khronos,” Khronos, [Online]. Available: https://www.khronos.org/news/press/khronos-releases-vulkan-1-0-specification. [Accessed 05 May 2016].

[16] “nvidia.com,” nvidia, [Online]. Available: http://developer.nvidia.com/transitioning-opengl-vulkan. [Accessed 19 05 2016].

[17] “Khronos.org,” Khronos, [Online]. Available: http://www.khronos.org/registry/vulkan/1.0/xhtml/vkspec.html. [Accessed 20 05 2016].

[18] M. Walton, “Arstechnica,” 19 February 2016. [Online]. Available: http://arstechnica.com/gaming/2016/02/vulkan-benchmarks-a-boost-for-amd-and-nvidia-but-theres-work-to-be-done/. [Accessed 18 May 2016].

Technical Documentation
Architecture Style 1.0

Created by: Thomas Hansen
08.04.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

KONGSBERG

Table of Contents

1. Document Overview

The technical documentation on architecture style presents the architectural challenge in the project and briefly explains several architectural styles as possible solutions. It also lists some pros and cons so that the reader can better understand the justification for our choice: A mixture of Pipe-filter style and component based architecture.

Describes

- why we need to have the right software style.
- the different styles that are suitable for our software system.
- the comparison criteria used in the Pugh matrix.
- a Pugh matrix that compares the alternatives presented.
- a conclusion with reasoning.

1.1 Document History

Version	Changes	Date	Author
0.1	Document started	08.04.2016	Thomas Hansen
0.2	Changed format on name and date in document history	10.04.2016	Trond Egil Hammer
0.3	Continue working on document	11.04.2016	Thomas Hansen
0.4	Corrections and clarifications	25.04.2016	Morten J. Barbala
0.5	Rewriting, references	28.04.2016	Morten J. Barbala
0.6	Several small changes	11.05.2016	Thomas Hansen
0.7	Added CBA and our own style to comparison.	15.05.2016	Thomas Hansen
0.8	Changed 4.1 to new standard and final cosmetic changes	18.05.2016	Thomas Hansen
0.9	Fixed front page and headings. Added glossary to referenced documents. Rewriting, corrections and clarifications.	19.05.2016	Morten J. Barbala
1.0	Final review	20.05.2015	Trond Egil Hammer, Morten J. Barbala, Thomas Hansen Ingvild Damtjernhaug

1.2 Referenced Documents

Title	Document	Version
Glossary	doc-1113_glossary_2_0.docx	2.0
Requirements document	doc-1213_requirements_2_0.docx	2.0

1.3 List of Figures

1.4 List of Tables

2. Architectural Challenge

The software architecture describes the building blocks through: (i) The nature of the architectural components, i.e. the nature of their computation, (ii) the way they interact with other components when composing a system and (iii) constraints on the way this composition is done. [1] Architecture is the structural map or blueprint of the software. [2] An architectural style describes a set of components, which perform a required function and the interfaces between these components. [2]

The main function of the Argos software system is getting data from one place to another (from cameras to VR goggles). The software must have a maximum of 75ms of glass (camera) to glass (Oculus) latency. This means that the architecture needs to focus on how data traverses through the system. The software also needs to be configurable and have an architectural style that allows user interactions with the software. There are several architectural styles that are all about data flow, and we will take a brief look at some of them. The selected styles described in this document are not the only relevant architecture styles, but the selected styles have a larger focus on flow of data, which is what needed in Project Argos.

3. Possible Solutions

Described in the next sections are the possible solutions for Project Argos. We focused on the highest priority: throughput. See requirement document for more information. We have also looked at other solutions like component based architecture to satisfy the need for interactivity.

3.1 Data Flow Architecture

In data flow architecture the software is seen as series of transformation on data from a series of inputs and outputs. The data and operations are independent of each other. The data enters the system and flows through the different modules until it reaches a final destination, for instance a screen or data storage. The connections between the modules can be implemented as input/output stream, buffers or as pipes. The data in the system can flow in a cycle, with a linear path or in a tree structure. This type of architecture is suitable for applications where the data need to go through well-defined data transformations or computations. [2]

3.1.1 Batch Sequential Architecture

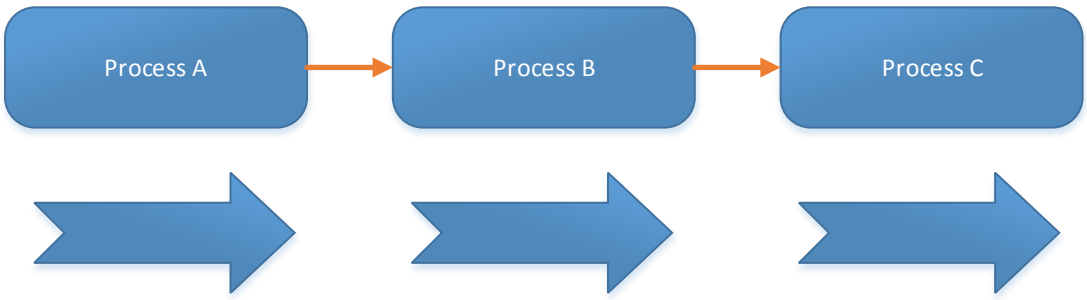

Figure 1: Batch sequential architecture.

Figure 1 shows a classical data processing model. In the batch sequential pattern, the data is carried in batches from module to module. The data cannot proceed before the module has finished all the computations on the data in the batch. This is a very simple style and is easy to understand, but it does not provide interactive interfaces and has a high latency and low throughput. Typical applications that uses this pattern is banking and utility billing softwares. [3]

3.1.2 Pipe and Filter Architecture

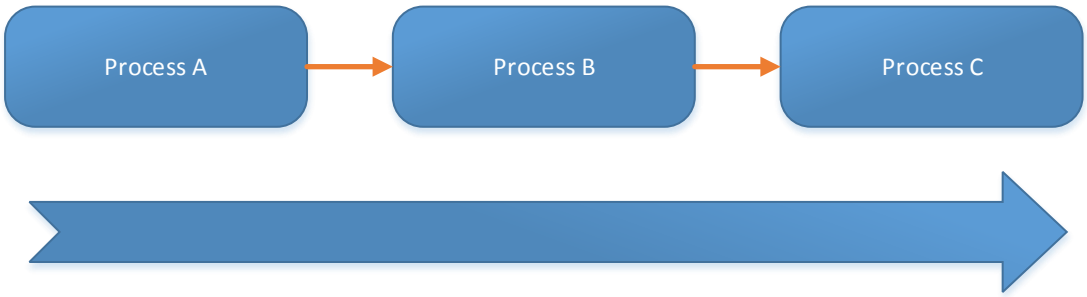

Figure 2: Pipe-filter architectural style

The pipe filter pattern looks a lot like batch sequential pattern, but is quite different. From figure 2 you can see that the data is sent between the modules as a stream and has a first in first out (FIFO) system. [2] This allows data to continuously flow through the modules without having to wait for data to finish computing. [2] The whole system is decomposed into data source, filters, pipes, and data sinks. The filters do the computation, i.e. transform the data from an input into what is needed as input in the next filter [2], and can be either passive or active. The active filters work with passive pipes and push and pull the data itself. Passive filters work in the opposite way and have active pipes push and pull the data. The filters must provide a control mechanism for the interface. [2]

The pipe and filter pattern is specialized for applications that needs a high throughput. [2] It provides reusability if two adjacent modules “agree” on the data being transmitted, which simplifies maintenance and modification (filters can be

added or replaced), and has low coupling between filters. [3] Furthermore, pipe filter pattern supports both sequential and parallel execution. [2] The pattern is easy to understand as pipes do not have states: They simply carry binary data or a stream of characters between filters. [2]

The pipe filter style looks a lot like the batch sequential pattern, and if you are not careful while programming, it is easy to end up with a semi batch sequential pattern. If you have one place in your code where you have to wait for another part to finish, then you are processing in batches. [3] Therefore the pattern is not suitable for dynamic interactions. [2]

3.1.3 Process Control Architecture

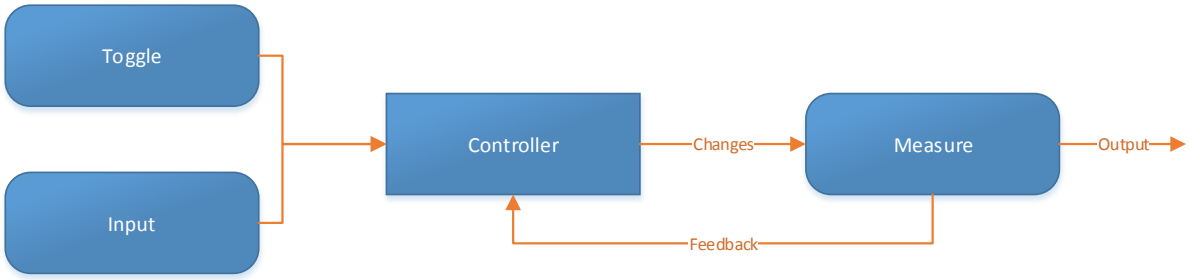

Figure 3: Process control architecture

I think that a better name for this architecture pattern would be feedback control system. The idea is that you have a controlled variable that is measured and a set of process variables that are manipulated. The measurement of the control variable is used to change one or more of the process variables so that the controlled variable is as close as it can get to a set point. Typical systems that use this architecture are car cruise control or air conditioning systems. [3]

3.2 Component Based Architecture

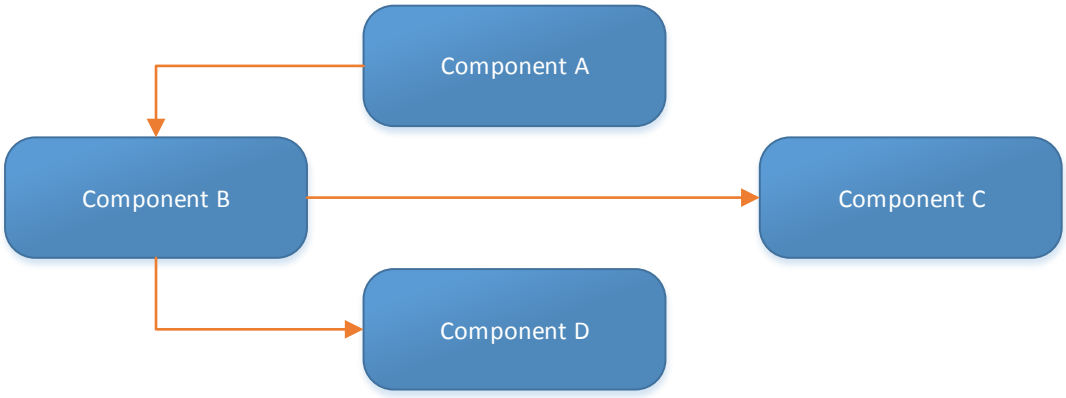

Figure 4: Component based architecture

Component based architecture describes the software system as a set of components that communicate with the help of interfaces. [3] The goal is to compose a system of easily interchangeable pieces that are loosely coupled by interfaces. [2]

Having an architecture like this also makes the software more maintainable as each component can be changed or upgraded by itself without affecting the rest of the software.

3.3 Our own Architecture

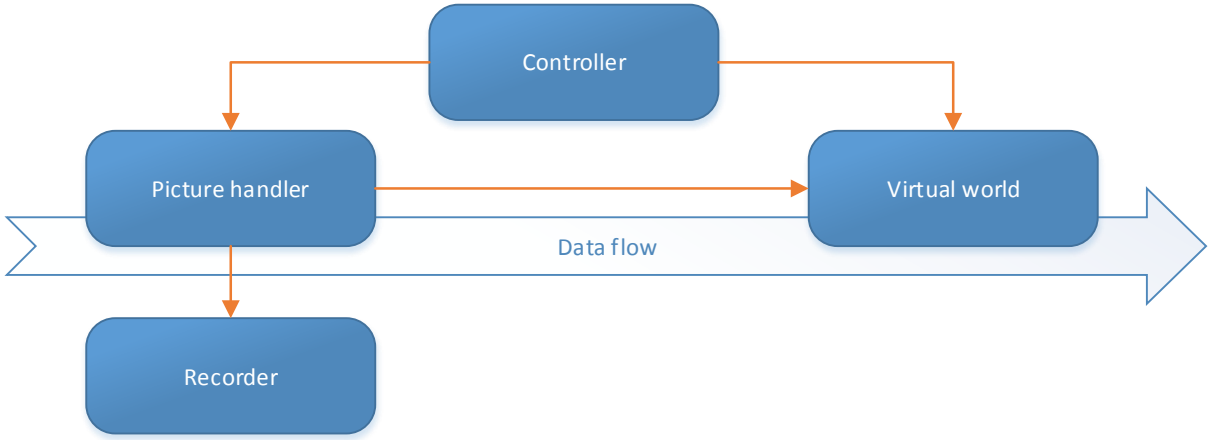

Figure 5: Our own architecture

A solution is to make our own architecture with the best from both worlds. This way we could keep the throughput from the pipe filter and the flexibility from component based architecture. The pipe filter style has similar construction as component based architecture, but in just one direction. We can use this and create an upgraded pipe filter pattern, with the dataflow in one direction and other modules that are outside of the main pipe and can work as valves that control the dataflow.

4. Pugh Matrix Comparison

The four architectural styles will be compared in a Pugh matrix diagram. The rating and importance of the criteria is given on a scale from 1-3. The score is calculated by multiplying the rating with the importance of the criteria. Higher is better.

4.1 Comparison Criteria

Criteria	Importance	Rating
Throughput	3 – Throughput represents how fast it takes for the data to travel from glass to glass.	1 = The data stops and waits for other processes to finish before it continues. 2 = The data have some bottlenecks in the software. 3 = The do not have unnecessary bottlenecks.

Modifiable	2 – It is easy to modify or replace the modules or mechanisms inside the modules, without having to make big changes in other modules.	1 = Not modifiable without making changes to the whole system 2 = Some changes have to be made in other parts of the software when changing some code. 3 = Changes in the software can be confined inside the components
Simple	3 – The time to learn or understand the style should take as little time as possible	1 = Very complicated, takes very long to understand. 2 = Some time needed to learn and understand the style. 3 = The style is easily understandable and takes no time to comprehend.

Table 1: Pugh matrix criteria

4.2 Pugh Matrix

		Batch sequential		Pipe-filter		Process control		CBA		Our own	
Criteria	**Importance**	**Rating**	**Score**	**Rating**	**Score**	**Rating**	**Score**	**Rating**	**Score**	**Rating**	**Score**
Throughput	3	1	3	3	9	1	3	2	6	3	9
Modifiable	2	3	6	2	4	2	4	3	6	3	6
Simple	1	3	3	3	3	2	2	3	3	2	2
	Total		**12**		**16**		**9**		**15**		**17**

Table 2: Pugh matrix dataflow architecture

5. Conclusion

As we can see from the Pugh matrix the best choice is if we make our own style out of pipe filter and component based style. This style has high scores on all criteria, only losing a point because of complexity. Pipe filter and component based style actually get the second and third place, respectively, and only lack modifiability and throughput. Batch sequential and process control style lack throughput and process control and are the least suitable for the software system. Our choice is therefore to make our own architectural style because this will give us the throughput and the possibility to interact with the data going through the system. Using a component based style will also make it easier to make changes to the software later.

Bibliography

[1] R. Juric, J. Kuljis and R. Paul, “Software architecutre style for interaoperable databases,” in *International Conference on Information Technology Interfaces*, Cavtat, Croatia, 2004.

[2] “Tutorialspoint,” 2015. [Online]. Available: http://www.tutorialspoint/software_architecture_design/software_architecture_design_tutorial.pdf. [Accessed 09 March 2016].

[3] A. Bijlsma, B. Heerendr, E. Roubtsovair and S. Stuurman, “Software Architecture,” [Online]. Available: ftacademy.org/sites/ftacademy.org/files/materials/fta-m11-doft_arch-pre.pdf. [Accessed 08 04 2016].

Technical Documentation
Lenses 2.0

Created by: Leiv Fredrik Berge
16.02.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

KONGSBERG

Table of Contents

1. Document Overview

The purpose of the technical documentation on lenses is to give the reader a clear understanding of the system solutions and lens technology we have considered in Project Argos. After reading this you should know the reasoning for our choice of lenses and solution.

Describes

- the camera system.
- the cameras used in Project Argos.
- lens technology.
- different setups and solutions.
- the comparison criteria used in the Pugh matrix.
- a conclusion with reasoning.

1.1 Document History

Version	Changes	Date	Created by
0.1	Created document	16.02.2016	Leiv Fredrik Berge
0.2	Spellcheck and added references	18.02.2016	Trond Egil Hammer
0.3	Added Pugh Matrix and Kowa lenses	19.02.2016	Leiv Fredrik Berge
0.4	Added proposal 3	24.02.2016	Leiv Fredrik Berge
0.5	Added specs on Kowa lens in the text	02.03.2016	Trond Egil Hammer
0.6	Added lens technology and bibliography	08.03.2016	Trond Egil Hammer
1.0	Added conclusion	08.03.2016	Leiv Fredrik Berge
1.1	Changed format on name and date in document history	10.04.2016	Trond Egil Hammer
1.2	Fix layout, corrections and clarifications	25.04.2016	Morten J. Barbala
1.3	Rewriting, corrections and clarifications. Additions to introduction, document overview, camera systems and lens system solutions	28.04.2016	Morten J. Barbala
1.4	Added figures to lens technology	04.04.2016	Trond Egil Hammer
1.5	Rewriting	06.05.2016	Trond Egil Hammer
1.6	Deleted introduction, formatted front page, rewritten document overview, corrections and clarifications	11.05.2016	Ingvild Damtjernhaug
1.7	Fixed layout, headings, table of contents and paragraph spacing	13.05.2016	Morten J. Barbala
1.8	Added Pugh matrix criteria table	16.05.2016	Leiv Fredrik Berge
1.9	Wrote intro to lens technology. Rewriting, corrections and	19.05.2016	Morten J. Barbala

	clarifications		
2.0	Final review	20.05.2016	Morten J. Barbala Thomas Hansen Ingvild Damtjernhaug

1.2 Referred Documents

Title	Document	Version
Glossary	doc-1113_glossary_2_0.docx	2.0

1.3 List of Figures

1.4 List of Tables

2. Camera System

The cameras available to us in Project Argos are Mako G223C PoE surveillance cameras. The Argos system has strict requirements in regards to response time, which is a key aspect regarding lens decision. We want to run our cameras with the minimum frame rate, 49.5 FPS, and this reduces the exposure time compared to a normal camera setup. [1] In turn, this requires a lens that lets as much light as possible through to the camera sensor. [1] However, this needs to be balanced with the range of focus because reducing the blender and letting more light into the sensor will reduce the depth of view. [1] It is important to remember that the camera system will be mounted on a vehicle so manual adjustment of the lenses will not be possible when the system is in use. Fixed focus lenses are therefore preferable to reduce the complexity and risk of system failure. The camera uses the standard C-mount lens system and a 2/3” image sensor size. This is also important as a lens will have a different view on differently sized sensors. [1]

The angle of view is also an important issue. Our solutions and the view in the VR-goggles are heavily impacted by the viewing angle of the lens. A wide-angle fisheye lens will provide a large field of view, but will distort the picture and create a vignette frame of black around the image. A narrow angle will have less distortion and no vignetting, but will require more cameras to cover the needed field of view to use the system. [1] Adding additional cameras will also provide a heavier load on the network and the rest of the system. [1] The narrow angle solution will ultimately require more stitching to have a continuous image in VR while the wide angle solution will require more distortion handling. In the future, tests should be run on different types of camera- and lens setups to see which solution is optimal in terms of resource use, video quality and frame rate.

3. Lens Technology

Lens technology is a complex subject. Constraints and requirements for real-time and virtual reality systems can increase this complexity even further. Terminologies and technical aspects are unfamiliar to us as computer engineering students, and this makes researching lens technology challenging and time consuming. Luckily, we have been able to talk with optometry experts at the university college to help with the research. This section will cover the factors that we believe to be the most relevant for this project.

3.1 Focal Length

Focal length is a calculation of an optical distance from where the light originates to the digital sensor when an object is in focus. [2] The focal length tells us how much of the scene is captured, the angle of view and the magnification. [2] With a long focal length the angle of view is smaller and the magnification will be higher, and vice versa. [2]

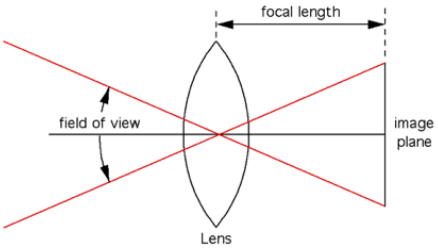

Figure 1: Focal length [3]

3.2 Aperture

In optics, an aperture is a hole in the lens where the light travels. It is designed similar to the human eyes. [3] The pupil in the human eye is a good analogy for aperture in photography. In a room with limited light the pupil in human eye is large and more light enters the retina. When the pupil is large the aperture will also be high and let more light through the lens. In a room with many light sources, however, the pupil will be very small and this equals small aperture letting through less light. [3] The aperture controls the depth of field: A large aperture will show both foreground and background in focus while a small aperture shows the foreground in focus and the background blurry. [3]

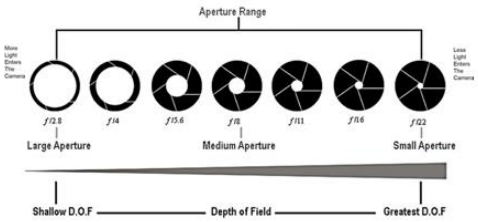

Figure 2: Aperture [2]

3.3 Vignetting

Vignetting means darkening of image corners when compared to the center of the picture. Vignetting is common in optics and photography. [4] Some lenses or external tools like filters and lens hoods can cause vignetting. Vignetting can also be applied with different post-processing software like Photoshop. [4] There are different types of vignetting: Optical, pixel and mechanical. [4] This section will cover optical vignetting since this is what we think will be most relevant and appear with the selected lenses.

Optical vignetting occurs naturally in all lenses. It depends on the design and construction of the lens and can be very strong on some lenses and barely perceivable on others. [4] On most modern lenses it will still be visible, especially on fixed lenses with large apertures. [4] There are two causes for optical vignetting: Light blocking and light travel distance. Firstly, light can be partially blocked by the lens barrel, i.e. the peripheral light rays are partially blocked at extreme wide angles and there will be less light at the edges of the picture. [4] This effect is most visible at

larger apertures. Secondly, when the light travels through the lens the light rays at the periphery of the lens will have to travel longer than the light rays in the center. This is more noticeable with wide-angle lenses. [4] In this case the cosine fourth law of illumination falloff kicks in. [4] The law states that the light reduction is proportional with the fourth power of cosine to the angle between the periphery light ray and the optical axis. Light rays that are further away from optical axis need to travel longer and therefore more vignetting will be visible. [4]

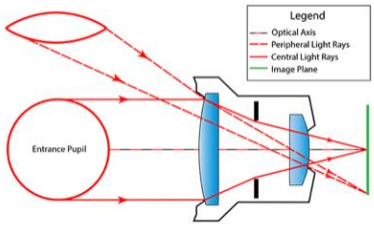

Figure 3: Optical vignetting

3.4 Distortion

In a picture that is taken with a very wide angle lens, you will notice that the picture is bended, a kind of deformation of the image. The lens deforms and bends the straight lines and makes them curvy. This is caused by the design of the lens or the position of the camera relative to the subject. [5] There are three known types of optical distortion: Barrel, pincushion and mustache. [5] For our study the barrel distortion is the most relevant. Barrel distortion occurs when the field of view of the lens is wider than the size of the image sensor. The picture is squeezed to fit, and the result of this is that straight lines are curved at the edges of the picture. The lines in the center are more or less straight. However the further the lines are from the image center the more curved they are. [5]

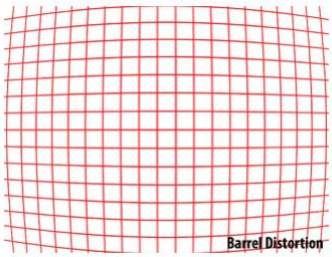

Figure 4: Distortion

4. Lens System Solutions

There are two main solutions we have looked into. They both share the same camera rig dictated by available hardware from earlier project teams, but the lenses create two distinctly different experiences. Since this is a bachelor project, we neither have the time nor resources to run tests and experiment to find the best solution, and have chosen what we think to be the optimal one according to criteria described in section 5.1.

4.1 Proposal 1: Three front-facing, one rear

The first proposed solution is to use three cameras to cover over 200 degrees in front of the driver, and use one fisheye camera to capture 180 degrees behind the vehicle. The data from the three front-facing cameras can then be merged into one single image, and leave the last camera to act as a rear-view mirror. This solution will not have much distortion and virtually no vignetting, but two seams will be present in front of the driver and need to be handled carefully. Using lenses with about 80 degrees of horizontal view, the stich will appear roughly in the area where the A-pillar normally is placed in a typical family car.

Distortion will be a key factor as the task of merging the images becomes harder and harder as the distortion rate grows. The described solution will cover 360 degrees of view, but not in a single continuous image, and the view in the vertical angle will be limited. This solution should, however, be possible to extend with more cameras to either cover more vertical space with cameras above or below the existing, or use more cameras to cover more horizontal space. The low level of distortion should translate into a much easier job of merging the images together. The way that three cameras cover the key area for driving also leaves room available in the Oculus viewable sphere to put specialized camera views or other useful information.

Lens	Angle of view	Focal length	Distortion (tv)	Aperture	Focus range	Price	Availability
Tamron M23FM06 [6]	74.3 x 58.4	6mm	-2.1%	F/1.8	0.1m - ∞	$500	Not in stock
Kowa LM6JC [7]	81.9 x 61.2	6mm	-10.7%	F/1.4	0.1m - ∞	1000 NOK	In stock
Kowa LM8JC [8]	64.2 x 47.7	8mm	-6.2%	F/1.4	0.1m - ∞	1000 NOK	In stock

Table 1: Overview of lenses we have considered

4.2 Proposal 2: Four fisheye lenses

Proposal 2 is to cover about 180 degrees with each camera, and use four to cover 360 degrees field of view. Then each image will be almost spherical and quite distorted with a visible vignetting frame. With this solution, merging the images together to a more or less seamless image will most likely require lots of computational power to fix the distortion. For this type of solution is it probably better to leave each sphere as four separate images rendered in the Oculus Rift. This will provide the user with much greater vertical viewing angle, and no seams will be visible for the driver in a normal front facing driving position. It's also not entirely clear how this system can be extended with more cameras as two cameras alone cover 360x140 degrees of view. The vignetting also makes it hard to stich the images together to a continuous image.

Lens	Angle of view	Focal length	Distortion (tv)	Aperture	Focus range	Price	Availability
Fujinon FE185C O086H A-1 [9]	185 x 140	N/A	N/A	F/1.8	0.2m - ∞	$800	Available

Table 2: Overview of fisheye lens

4.3 Proposal 3: Produce our own lenses

A third alternative is to build our own lens system. This would be a very simple solution, which as an added bonus could be serviceable in the field if a lens somehow is broken. The basic idea is to create a small hole in front of the camera so the light entering the lens would be focused towards the image sensor. The angle of view would be decided by the size of the hole and the distance from the hole to the sensor. The solution would be very affordable, potentially user serviceable and should give a very low level of optical distortion. Another advantage is how easy it would be to test different types of aperture and angle of views.

The camera system shares the setup with proposal 1: Three front-facing and one rear-facing camera. The lens casings could be 3D printed to fit with the C-mount, and different size holes from 1 mm up to 5 mm could be drilled in the center of the lens. We could also add a hard eye lens of +1 that could be fixed to the print to focus the light even more. It might also be possible to create a rectangular lens, which might be more suitable for the live-merging we want.

4.3.1 Testing Proposal 3

Our initial testing of proposal 3 has not been promising. On our first test we attached our 1mm lens to the outside of the camera C-mount. Using ideal exposure time to keep a frame rate of 49.5 FPS, we didn't have any light in the image, and as we increased exposure time the image started to appear, but very blurry. A larger aperture opening improved the light performance slightly, but didn't improve the blurry image. In later testing we have tried to mount 1mm to 2mm lenses closer to the image sensor inside the C-mount. This performed about the same in regards to exposure time and light sensitivity, but degraded the color performance and the image is still unusably blurry. If we can't greatly improve the focus of the image, this will not be a viable solution, and therefore this solution is not included in the selection area until we reach a usable version of this proposal. However, this approach is still worth pursuing in the future to allow a user to create an emergency solution in the field and as a means to test a variety of different field of view combinations.

5. Pugh Matrix Comparison of Proposals

The three possible solutions will be compared on a Pugh matrix. The rating and importance of each criterion is given on a scale of 1-3. The score is calculated by multiplying the rating with the importance of the criterion. Higher is better. The proposal relies on the best lenses in each category. If another lens is picked the matrix needs to be updated to reflect the differences in the optical performance of the selected lens.

5.1 Comparison Criteria

Total horizontal viewing angle	3 - The combined view of all cameras in the horizontal space.	1 = < 150 degrees 2 = 150 – 180 degrees 3 = >180 degrees
Vertical viewing angle	2 - This refers to the angle of view in the vertical space. More vertical space will provide the operator with a more lifelike view and more awareness.	1 = < 50 degrees 2 = 50-60 degrees 3 = < 60 degrees
Distortion	3 - Distortion is how much the cameras alter the reality to capture the information to the image sensor. This is a combination of the optical properties of the lens and the image sensor itself. More distortion will give an unnatural picture, and require more computational time to restore the distorted image and merge the images to a continuous picture.	1 = > 20% 2 = 10 – 20% 3 = < 10%

Light sensitivity / Field of depth	1 - A short exposure time as a result of a high frame rate means we need to capture as much light as we can while the shutter is open. However a low aperture also means short field of depth, so this must be balanced. Field of depth is also a factor as less aperture gives less depth of field.	1 = > F/1.8 2 = F/1.6 – F/1.8 3 = < F/1.6
Extendibility	1 - How easy it is to extend the system with more cameras to provide the system operator with additional information.	1 = Complex to stich in more cameras 2 = Acceptable to implement more cameras. 3 = Easy to implement more cameras
Cost	How expensive is the solution.	1 = > 10 000 NOK 2 = 5 000 – 10 000 NOK 3 = < 5 000 NOK

Table 3: Pugh matrix criteria

5.2 Pugh Matrix Proposals

		Options			
		Proposal 1		Proposal 2	
Criteria	**Importance**	**Rating**	**Score**	**Rating**	**Score**
Total horizontal viewing angle	3	2	6	3	3
Vertical viewing angle	2	2	4	3	6
Distortion	3	3	9	1	3
Light sensitivity / Field of depth	2	3	6	1	1
Extendibility	1	2	2	1	1
Cost	2	2	4	1	2
	Total		**31**		**16**

Table 4: Pugh matrix on lens setup

Our recommendation is to choose proposal 1, which both scores highest in the Pugh matrix comparison and in our analysis. This in our opinion the optimal solution that provides an image that is more comfortable for the user and more flexible for the developers.

6. Conclusion

Lenses are highly complex, and all the variables create different challenges, solutions and user experiences. We have decided to go for the solution with three front-facing lenses and on rear-facing fisheye lens. For the fish eye lens the Fujinon FE185C086HA-1 is the lens of choice, and for the front facing we decided to go with the Kowa LM6JC. This solution will cover well over 360 degrees of view, even if we choose to crop the image of the front facing cameras. We believe this solution will be the most effective and give the most pleasant experience for the user. If the live merging and distortion correction is faster and smoother than expected, an all fisheye solution will provide the user with more vertical field of view and this can be research further in future development.

Bibliography

[1] "Mesa public schools," [Online]. Available: www.mpsaz.org/rmhs/sewqright/photography-resource/files/camera_exposure_chart2.pdf. [Accessed 19 05 2016].

[2] Nikon Usa, "Nikon Usa," [Online]. Available: www.nikonusa.com/en/learn-and-explore/article/g3cu6020/understanding-focal-length.html. [Accessed 08 March 2016].

[3] Nikon Usa, "Nikon Usa," [Online]. Available: www.nikonusa.com/en/learn-and-explore/article/g3cu601r/understanding-maximum-aperture.html. [Accessed 08 March 2016].

[4] Photographylife, "Photographylife," [Online]. Available: https://photographylife.com/what-is-distortion. [Accessed 8 March 2016].

[5] Photographylife, "Photographylife," [Online]. Available: https://photographylife.com/what-is-vignetting. [Accessed 08 March 2016].

[6] Tamron, "Tamron Lenses," Tamron, [Online]. Available: http://www.tamron.biz/en/data/fa/catalog/fa_e.pdf. [Accessed 16 March 2016].

[7] Kowa Europe, "Kowa Lenses," Kowa, [Online]. Available: http://www.kowa-europe.com/lenses/en/LM6JC.1309.php. [Accessed 16 March 2016].

[8] Kowa Europe, "Kowa Lenses," [Online]. Available: http://www.kowa-europe.com/lenses/en/LM8JC.1496.php. [Accessed 16 March 2016].

[9] Fujifilm, "Fujinon CCTV Lens," Fujifilm, [Online]. Available: http://www.fujifilm.com/products/optical_devices/pdf/cctv/fa/fisheye/fe185c086ha-1.pdf. [Accessed 16 March 2016].

Research Document
Motion Sickness 1.0

Created by: Ingvild Damtjernhaug
19.02.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

HSN

KONGSBERG

Table of Contents

1. Document Overview

The research document on “Motion Sickness” takes a closer look at factors that contributes to motion sickness and why this is important in Project Argos.

Describes

- the context, why it applies to Project Argos.
- what motion sickness is and what symptoms that may occur.
- different types of motion sickness.
- the relationship between motion sickness and virtual reality.
- factors contributing to motion sickness.
- a conclusion.

1.1 Document History

Version	Changes	Date	Created by
0.1	Document created	19.02.2016	Ingvild Damtjernhaug
0.2	Formatted	10.04.2016	Trond Egil Hammer
0.3	Added sections about factors and experience, references, formatted front page	09.05.2016	Ingvild Damtjernhaug
0.4	Added more on sections about factors contributing to motion sickness, document overview, conclusion and pictures	18.05.2016	Ingvild Damtjernhaug
0.5	Restructuring of paragraphs, corrections and clarifications	19.05.2016	Morten J. Barbala
1.0	Final review	20.05.2016	Trond Egil Hammer Ingvild Damtjernhaug Morten J. Barbala

1.2 References

Title	Document	Version
Tec.Doc. Lenses	doc-21321_lenses_2_0.docx	2.0
Tec.Doc. VR Goggles	doc-21321_VR_goggles_3_0.docx	3.0
Glossary	doc-1113_glossary_2_0.docx	2.0
Requirements	doc-1213_requirements_2_0.docx	2.0

1.3 List of Figures

2. Context

Project Argos is a system that makes it possible to drive a vehicle by seeing through virtual reality goggles (VR goggles). Many people experience motion sickness when using VR goggles, so it is important to know which factors that cause motion sickness, both at this stage and for further development. It must be possible to wear the goggles without experiencing motion sickness; this is one of the main requirements of the system.

Our hypothesis is that delay is the most important factor, but there are probably more factors that impact the experience. The project team needs to know how to reduce the chances of a user experiencing motion sickness as there is no way for a user to escape the discomfort of motion sickness. Therefore it is important to understand its causes and implement strategies to minimize the risk for it to occur.

3. What is motion sickness?

Motion sickness is also known as kinetosis or travel sickness. It is a condition in which a disagreement exists between visually perceived movement and the vestibular system's sense of movement. In other words, you can experience motion sickness when the eyes tell the body you are at rest, but the balance organ in your inner ear says you are moving, or vice versa. The visual system is the eyes and the perception of the visual impressions. The vestibular system is located in the inner ear, and is the system that deals with balance and coordinating movement with balance. The proprioceptive senses gives the human body the ability to sense stimuli regarding position, motion and equilibrium.

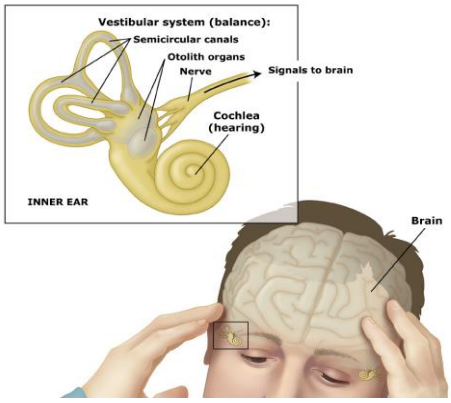

Figure 1: Vestibular system, inner ear

Dizziness, fatigue and nausea are the most common symptoms, but other syndromes and illnesses can also be associated with motion sickness. Some people can experience headache, cold sweat, vomiting, shallow breathing and/or extreme tiredness related to motion sickness. Which symptoms and how strong the symptoms will be, differs from person to person. Some people never experience motion sickness. The scientists are not sure why some people seem to steer clear of the symptoms related to motion sickness and others do not. [1]

There are different hypothesis for the cause of motion sickness. The most common is that it functions as a defense mechanism against neurotoxins. The area postrema in the brain is responsible for inducing vomiting when poison is detected, and for resolving conflicts between vison and balance. The discordance between what the eye sees and what the brain senses, will make the brain believe that the person is hallucinating and conclude that the hallucinating is due to poison ingestion. The body responds by vomiting, to clear the assumed toxin. [2]

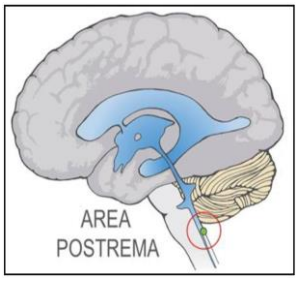

Figure 2: Area postrema

3.1 Types of Motion Sickness

Depending of the cause, motion sickness can also be referred to as sea-sickness, car-sickness, simulation-sickness or air-sickness. They are all really the same, caused by motion sickness, and can be divided into three categories: Motion sickness caused by motion that is felt but not seen, caused by motion that is seen but not felt or caused by different motion detected by both the systems.

4. Virtual Reality and Motion Sickness

Motion sickness due to virtual reality is often called simulator sickness and is very similar to motion sickness induced by film. In virtual reality, however, the effect is made more acute as all external reference points are blocked from vision. [3] This is a form of induced motion sickness, which differs from “the regular” motion sickness. The motion sickness people are most familiar with comes from actual motion, e.g. the movement in a car causing car-sickness or in a boat causing sea-sickness.

The feelings of discomfort associated with simulator sickness occur when visual information from a simulated environment signals self-motion in the absence of any actual movement. However, it is still the actual conflict between the visual, vestibular, and proprioceptive senses that causes discomfort. Simulator sickness has several symptoms and they are similar to the symptoms for motion sickness in general. Some users might experience some degree of simulator sickness after a short period of time using VR goggles, while others may never experience it.

The symptoms of simulator sickness are primarily characterized by disorientation, nausea and oculomotor discomfort. The disorientation is caused by a sense of disrupted balance, called ataxia. Ataxia is disruptions in balance and coordination of movements. [1] The nausea is believed to stem from vection, the illusory perception of self-motion. The oculomotor discomfort gives symptoms that are unique to using a virtual environment, such as eye strain/fatigue. [2] In Project Argos this can actually be worse than normal simulator system, as the system is mounted in a vehicle in motion. This means we can potentially get the normal motion sickness on top of the simulator sickness. If the delay between the visual input and the physical input is large that could exaggerate the motion sickness effects.

5. Factors Contributing to Motion Sickness

The exact causes of simulation sickness and of all other forms of motion sickness are still being researched. The following section lists factors that have been studied as potential contributors to motion sickness. It is difficult to know the particular cause for simulator sickness and consequently different users will have different experiences. Sensitivity to different types of stimuli can also vary and the symptoms can take a while to manifest.

Motion sickness susceptibility varies in the population and correlates with the intensity of simulator sickness experiences. [4] This means users who know they tend to experience motion sickness in vehicles, rides, and other contexts should approach using VR carefully. People who use VR a lot will often tolerate more. Long exposure to virtual environments can train the brain to be less sensitive to their effects. [3] As such, a driver using Project Argos will probably become less susceptible to simulator sickness than most users.

5.1 Speed of Movement and Acceleration

The speed is not necessarily what causes simulator sickness. Slower speeds will generally feel more comfortable, but the most important issue is acceleration, which is the stimulus to which the inner ear vestibular organs respond. [5] Acceleration, linear or angular, in any direction conveyed visually, but not to the vestibular organs, constitutes a sensory conflict that can cause discomfort. An instantaneous burst of acceleration is more comfortable than an extended, gradual acceleration to the same movement velocity. Discomfort will increase as a function of the frequency, size, and duration of acceleration. The vestibular organs do not respond to constant velocity, so constant visual motion represents a smaller conflict for the senses. [6] The more coordinated the movement the driver feels with what he/she sees, the less conflict for the senses.

5.2 Degree of Control

Taking control of the camera away from the user or causing it to move in ways not initiated by the user can lead to motions sickness. Some theories suggest the ability to anticipate and control the motion experienced plays a role in staving off motion sickness, and this principle appears to hold true for motion sickness due to VR as well. Unexpected camera movement (or cessation of movement) outside the user's control can be uncomfortable. [6]

5.3 Duration

The longer the driver remains in a virtual environment, the more likely the driver is to experience motion sickness. But as stated in Section 5, the driver will also tolerate more after a period of much exposure to virtual environment.

5.4 Altitude

The altitude of the driver, or the height of the user's point of view (POV), can be an indirect factor in simulator sickness. The lower the user's POV, the more rapidly the ground plane changes and fills the user's FOV, creating a more intense display of visual flow. [6]

5.5 Binocular Display

Virtual Reality is perceived different from “real world”. One of the most important reasons for this is the screen you look at. With VR the object (more precisely the screen), is placed in a fixed potion. The eyes do not need to adjust to the right distance, because the distance from the screen to your eyes does not change.

The Oculus Rift has two separate displays. Binocular disparity is one of the key features in VR goggles, and gives the depth vision, but it is not without its costs. Stereoscopic images can force the eyes to converge on one point in depth while the lens of the eye accommodates (focuses itself) to another. [6] This can put an extra strain on the eyes, especially if the VR goggles do not have any eyesight correction adjustments or if the user has a pre-existing eye condition.

Figure 3: Binocular display

Content that the user focuses on for extended periods of time (such as menus or maps), should be placed in a range of 0.75 to 3.5 meters away. Some people find viewing stereoscopic images uncomfortable, and research has suggested that reducing the degree of disparity between the images to create a monoscopic display can make the experience more comfortable. [7] This will not apply to Project Argos yet, because the picture is not in 3D, but would be worth getting closer into in further development.

5.6 Field of View

Field of view can refer to two kinds of field of view: The area of the visual field subtended by the display, called display FOV (dFOV), and the area of the virtual environment that the graphics engine draws to the display, called camera FOV (cFOV). A wide dFOV is more likely to contribute to simulator sickness primarily for two reasons related to the perception of motion. Firstly, motion perception is more sensitive in the periphery, making users particularly susceptible to effects from both optic flow and subtle flicker in peripheral regions. Secondly, a larger display FOV, when used in its entirety, provides the visual system with more input than a smaller display FOV. [4] When that much visual input suggests to the user that they are moving, it represents an intense conflict with bodily (i.e. vestibular and proprioceptive) senses, leading to discomfort.

Reducing display FOV can reduce simulator sickness, but also reduces the level of immersion and situational awareness with the Oculus Rift. [8] This is not desirable in Project Argos, but to best accommodate more sensitive users who might prefer that compromise, it might be allowed for the user to adjust display FOV. Visibility of on-screen content should not be adversely affected by changing display FOV. Having a

cockpit or vehicle obscuring much of the vection-inducing motion in the periphery may also confer a similar benefit for the same reasons. However, the smaller the user's view of their environment is, the more he/she will have to move his/her head to maintain situational awareness, which can also increase discomfort.

Manipulating camera FOV can lead to unnatural movement of the virtual environment in response to head movements. For example, if a 10° rotation of the head creates a rotation of the virtual world that would normally require a 15° rotation in reality. In addition to being discomforting, this can also cause something known as vestibular-ocular reflex (VOR) gain adaptation. [10] The human eyes and vestibular system normally work together to determine how much the eyes must move during a head movement in order to maintain stable fixation on an object. If the virtual environment causes this reflex to fail to maintain stable fixation, it can lead to an uncomfortable re-calibration process both inside the VR goggles and after use.

5.7 Latency

There are different types of latency. It can be latency with the head mounted display (HMD), as you move your head the system uses some time to register the movement and move the virtual world. To combat this it is important to have a fast refresh rate on the display. It is also important to make sure the VR experience does not lag or drop frames. In Project Argos you can also experience delay from when the camera captures a frame until it is displayed in the HMD. This is called glass-to-glass latency and requires fast cameras, transfer speed and computational speed to reduce. Past research findings on the effects of latency are somewhat mixed. Many experts recommend minimizing latency to reduce simulator sickness because lag between head movements and corresponding updates on the display can lead to sensory conflicts and errors in the vestibular-ocular reflex. The producers of the Oculus Rift encourage minimizing latency as much as possible. [9]

It is worth noting that some research with head-mounted displays suggests a fixed latency creates about the same degree of simulator sickness whether it's as short as 48 ms or as long as 300 ms. [8] However, variable and unpredictable latencies in cockpit and driving simulators create more discomfort the longer they become on average. This suggests that people can eventually get used to a consistent and predictable bit of delay, but fluctuating, unpredictable delays are increasingly discomforting the longer they become on average. [8]

Adjusting to latency and other discrepancies between the real world and VR can be an uncomfortable process that leads to further discomfort when the user adjusts back to the real world outside of VR. The experience is similar to getting on and off a cruise ship. After a period feeling seasick from the rocking of the boat, many people become used to the regular, oscillatory motion and the seasickness subsides, but upon returning to solid land, many of those same people will actually experience a "disembarkment sickness". This is because the body has to readjust once again to its new environment. [10] The less you have to make the body adjust to entering and exiting VR, the better.

5.8 Distortion Correction

We have two types of distortion: The normal distortion that the HMD needs to account for, which the Oculus SDK specifies, and the distortion the cameras create in the video images. It is important that the HMD distortion is done correctly and according to the SDK's guidelines. Incorrect distortion can be perceived as fairly correct, but still feel disorienting and uncomfortable. The producer of the Oculus Rift carefully tune the distortion settings to the optics of the Rift lenses and are continually working on ways of improving distortion tuning even further. [9]

The lenses in the camera rig in Project Argos distort the image shown on the display in the VR goggles. The distortion depends on both the lenses and the image sensors on the cameras, is corrected by the post-processing in the software TinyArgos 2.0. The edges of the video will be slightly warped because of the wide angle of the lens, and in TinyArgos 2.0 we reverse that effect by de-warping the images. See the Technical Documentation on Lenses for more information.

5.9 Flicker

Flicker plays a significant role in the oculomotor component of simulator sickness. It can be worsened by high luminance levels and is perceived most strongly in the periphery of your field of view. Although flicker can become less consciously noticeable over time, it can still lead to headaches and eyestrain. The OLED displays used for VR provide many advantages, but also carry with them some degree of flicker, similar to CRT displays. See Technical Documentation on VR goggles for more information.

Different people can have different levels of sensitivity to flicker. However, the 75-hz display panels of the Oculus Rift DK2 used in Project Argos are fast enough that the majority of users will not perceive any noticeable flicker. Future VR goggles will have even faster refresh rates and therefore less perceptible flicker. This is more or less out of our hands as developers, but it is included here as a reminder for future development.

5.10 Experience

The more experience users have had with a virtual environment, the less likely they are to experience simulator sickness. [4] Most users will need time to acclimate to using the VR goggles and Project Argos while driving a vehicle. Drivers who use Project Argos repeatedly will probably be more resistant to simulator sickness than a new user. New users should not be thrown immediately into intense driving sessions.

6. Conclusion

Simulator sickness refers to symptoms of discomfort that arise from using simulated environments, and are caused by conflicts between the visual and bodily senses. Numerous factors contribute to simulator sickness, but several of the factors are out of the hands of the developers in Project Argos. Additionally, some of them do not apply to the project at this stage, but are worth a closer look in further development. Project Argos use Oculus Rift DK2. See technical documentation on VR goggles for more information.

Some problems will probably solve themselves by updating to the newest version of the VR goggles, but when performing the internal acceptance test for the system, none of the test drivers experienced any discomfort. It seems that the glass-to-glass latency is low enough that it do not exaggerate the effects of motion sickness, and the HMD latency is present but not too intrusive. However, the duration was not very long and the drivers did not wear the VR goggles for more than maximum 15 min at a time. Testing for motion sickness should be given more attention in further development.

Bibliography

[1] B. S. Hromatka, Y. Joyce and N. Ericsson, "Generic variants associsted with motion sickness point to roles for inner ear development, neurological processes, and glucose homeostasis," vol. 1, 2015.

[2] C. R. Sherman, "Motion sickness: review of causes and preventing srategies," 2002.

[3] R. S. Kennedy, N. E. Lane and K. S. Berbaum, "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness.," no. 3.

[4] K. M. Stanney, K. S. Hale and Nahmens, "What to expect from immersive virtual environment exposure: influences of gender, body mass index, and past experince," 2003.

[5] R. H. So, W. T. Lo and A. T. Ho, "Effects of navigation speed on motion sickness caused by an immersive virtual environment," 2001.

[6] A. Rolnick and R. E. Lubow, "Why is the driver rarly motion sick? The role of controllability in motion sickness," 1991.

[7] J. A. Ehrlich and M. J. Singer, "Simulator sickness in stereoscopic vs. monoscopic helmet mounted displays.," 1996.

[8] M. H. Draper and V. J. Gawron, "Effects of image scale and system time dealy on simulator sickness within head-coupled virtual enivironments".

[9] "Oculus Rift," [Online]. Available: http://developer.oculus.com. [Accessed 18 May 2016].

[10] J. T. Reason and J. J. Brand, Motion Sickness, Academic Press, Inc, 1975.

[11] S. H. Schwartz, Visual Perception, The McGraw-Hill Companies, 2004.

Argos User Guide 1.0

Created by: Morten J. Barbala
02.05.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

KONGSBERG

Table of Contents

1. Document Overview

The user guide is a guide and manual for the Argos system. It contains a list of controls (keyboard presses), a launch guide, a list of possible issues when troubleshooting and descriptions of standard use.

Describes

- how to successfully launch the system.
- possible issues when troubleshooting.
- all available keyboard controls and actions.
- standard use of the system.

1.1 Document History

Version	Changes	Date	Author
0.1	Created document	02.05.2016	Morten J. Barbala
0.2	Added new controls and corrected examples of use	06.05.2016	Morten J. Barbala
0.3	Implemented document standard. Wrote document overview. Fixed new controls and examples of use	13.05.2016	Morten J. Barbala
0.4	Added new keyboard controls and updated examples of use. Added troubleshooting, launch guide and system requirements. Created figure over keyboard controls.	18.05.2016	Leiv Fredrik Berge, Morten J. Barbala
0.5	Corrected document overview and headings. Removed “System Requirements”. Added list of figures and list of tables. Rewriting, corrections and clarifications.	19.05.2016	Morten J. Barbala
1.0	Final review	20.05.2016	Ingvild Damtjernhaug, Morten J. Barbala, Thomas Hansen

1.2 List of Figures

1.3 List of Tables

2. Launch Guide

You must launch the TinyArgos.exe executable file to start using the Argos system. The program must be launched from a disk that is fast enough to handle the data transfer speed if you want to use recording or playback. TinyArgos 2.0 creates a console window where the program status is displayed and a rendering window that duplicates the virtual world as it is displayed in the VR goggles. The console windows display framerate, package delivery and configuration loading status.

2.1 Troubleshooting

Make sure:

- The Oculus and tracker is connected
- The Oculus Rift is turned on
- The Oculus Rift is in direct to HMD
- The Oculus runtime library is updated
- The Oculus runtime is running
- The OpenGL runtime is updated
- The glew32.dll file is placed in the same directory as the executable
- The cameras is assigned correct IP-address
- The latest graphics card drivers is installed
- The latest network card driver is installed
- The four ports on the NIC is teamed up as one port
- The DHCP server application is running
- The executable is placed on a sufficiently fast drive
- The RAID0 setup is working correctly

3. List of Controls

The user will use the keyboard to control the Argos system. The list of controls contains the keys and the actions they are bound to. See fig. 1 for a visual representation of the keyboard controls.

1, 2, 3, 4, 5	Load main config file, this stops playback
6, 7, 8, 9, 0	Load HUD config file, this is disabled in playback
W, A, S, D	Rotate view (debug mode only)
Spacebar	Reset and center view
R	Start/stop recording
P	Bring up explorer to select file for playback
Right Ctrl	Pause/Play playback
Left, Right	Skip 10 seconds forwards/backwards in playback
Delete, Page Down	Skip 1 minute forwards/backwards in playback

Table 1: Keyboard controls

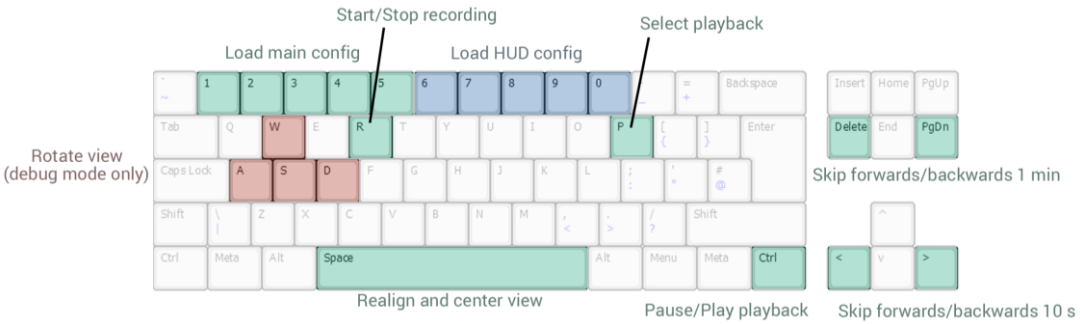

Figure 1: Keyboard controls

4. Examples of Use

The following sections describe typical usage scenarios for live video, recording and playback.

4.1 Live Video

1. Launch TinyArgos.
2. Use keys “0-5” to set virtual world from config files.
3. Choose HUD config with keys “6-0”.
4. Move head and look around to see the virtual world.
5. Change virtual world with keys “0-5”.
6. Change HUD “6-0”.
7. Move head and look around to see the virtual world.
8. Exit TinyArgos.

4.2 Recording

1. Launch TinyArgos.
2. Use keys “0-5” to set virtual world from config files.
3. Use keys “6-0” to set HUD.
4. Press “R” to start recording, an icon in the top-right corner will be visible to indicate that video is recording.
9. Press “R” to stop recording, the recording icon will disappear.
10. Exit TinyArgos.

4.3 Playback

1. Launch TinyArgos.
2. Press “P” to bring up the file explorer.
3. Navigate to recording folder. The explorer will open in the last recording folder that was played. To find any other recordings go back one level. If no recording exists, the computers “document” folder will be opened. The recordings are stored in an ‘YYYYMMDD.HHMMSS’ format.
4. Select a directory with recorded video.
5. Select config.xml file in the selected directory.
6. Press “Open” to start playback.
7. A loading icon is displayed and the playback will start after buffering for a couple seconds.
8. Move head and look around to see the virtual world.
9. Press “right ctrl” to pause/play.
10. Press “left” and “right” to skip 10 seconds forwards and backwards in playback.
11. Press “delete” and “page down” to skip 1 minute forwards and backwards in playback.
12. Use keys “1-5” to load a standard config file and quit playback.
13. Exit TinyArgos.

Glossary 2.0

Created by: Trond Egil Hammer
15.01.2016

Bachelor project 2016 at
Faculty of Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

KONGSBERG

1. Document Overview

The glossary contains an alphabetical list of terms with the definitions for those terms used in Project Argos.

1.1 Document History

Version	Changes	Date	Created by
0.1	Document created	15.01.2016	Trond Egil Hammer
0.2	Update, added descriptions	15.02.2016	Trond Egil Hammer
1.0	Added descriptions	10.04.2016	Trond Egil Hammer
1.1	Converted to word-file, updated, added descriptions	25.04.2016	Ingvild Damtjernhaug
1.2	Added descriptions	28.04.2016	Ingvild Damtjernhaug
1.3	Added descriptions, document overview, header and fixed formatting and front page.	02.05.2016	Ingvild Damtjernhaug
1.4	Added descriptions, formatted	18.05.2016	Ingvild Damtjernhaug
1.5	Added shader and vertex	20.05.2016	Thomas Hansen
2.0	Final review	20.05.2016	Ingvild Damtjernhaug, Morten J. Barbala, Thomas Hansen

2. Glossary

Terms	Descriptions
Allied Vision	Manufacturer of cameras and GigE Vision SDK.
API	Application programming interface. A set of routines, protocols and tools for building software applications.
Architecture	In the meaning Software Architecture. The high level structures of a software system.
Argos	The name of the project. Named after the Greek God that never closed his eyes.
Argos Machine(PC)	Computer that runs the Oculus Rift.
Artifact	A document produced from finished tasks.
Ataxia	Dysfunctions in parts in the human nervous system that coordinates movement.
Backbone	The server and the hardware.
Binocular display	A pair of identical or mirror-symmetrical displays mounted side-by-side.
Burndown report	Reports on condition in the project. It shows the actual and estimated amount of work to be done in an iteration.
C mount	A type of lens mount that provides the male thread.
Cat6/Cat5e	Category 6 or 5e twisted pair network cable. A type of Ethernet cable.
cFOV	Camera field of view.
Crossfire	Is a brand name for the multi-GPU solution by Advanced Micro Device (AMD). A technology that allows up to four GPUs to be used in a single computer.
Cyber-Physical System	A system composed of physical entities controlled or monitored by computer based algorithms.
dFOV	Display field of view.
DHCP	Dynamic Host Configuration Protocol. A standardized network protocol used on Internet Protocol networks for dynamically distributing network configuration parameters.
DirectX	A collection of APIs for handling tasks related to multimedia on Microsoft platforms.

DK2	Second Oculus Rift developer kit, which includes an HMD, external tracker, and an SDK.
eBus	Energy Bus. A 2-wire digital seral data-bus communication interface.
External supervisor	The supervisor from KDS (the company that owns the project).
Fish eye lens	A type of lens with ultra wide angle that produces strong visual distortion intended to create a wide panoramic or hemispherical image.
Flicker	Visible fading between cycles displayed in the VR-headset.
Flowchart	A type of diagram that represents workflow or process. Showing the steps as boxes.
FOV	Field of view.
Frame grabber	A unit that captures (i.e. “grabs”) individual, digital still frames from the cameras.
Fujinon	Manufacturer of lenses.
Gantt	A type of bar chart that illustrate a project schedule.
Gigabit PoE Switch	A network switch with 1 Gb/s transfer speed, see also PoE.
GigE Vision Camera	The cameras uses in the Argos System based on the GigE Vision standard.
GLFW	A utility library for use with OpenGL. It provides the ability to create and manage windows and OpenGL context, and handle keyboard and mouse input.
GPS	Global Position System. Spaced based navigation system that provides location and time anywhere on or near the Earth where there in an unobstructed line of sight to four or more GPS satellites.
GPU	Graphics processing unit.
Head mounted display	A general term for any display device worn on the head.
Heads-up display	A transparent display that presents data without requiring users to look away from their usual viewpoint. Takes it names from the heads-up displays used in modern aircraft.
HMD	See head mounted display.
HUD	See heads-up display.
I7 5930K	Intel Processor used in the Argos machine.

IDS	Integrated Defence Systems, a part of KDS.
Internal supervisor	Our supervisor at University College of Southeast Norway.
Iteration	The act of a repeating process. In agile software development, iteration is a single development cycle, often measured as two weeks.
Kanban board	A work and workflow visualization tool that helps optimizes flow of work. We use a physical board with sticky notes.
KDA	Kongsberg Defence & Aerospace.
KDS	Kongsberg Defence Systems.
KOG	Kongsberg Gruppen.
Latency	The time delay from when one thing happens till the effect is seen. E.g. glass-to-glass latency, the latency from a image is captured by the camera, till it is displayed on a monitor.
Linux	Computer operating system.
LMC	Last Mile Communication. Company that delivers lenses.
Markers	Elements that appears in HUD.
Merge pictures	Stitch live video from the cameras into a continuous video stream.
MTBF	Mean time between failures. The predicted time between inherent failures of a system during operation.
MTTR	Mean time to repair. A basic measure of the maintainability of repairable items.
Mutex	A synchronized mechanism for enforcing limits on access to a resource in an environment where there are many threads of execution.
MVC	Model-view-controller. A software architectural pattern for implementing user interfaces on computers.
N^2	N^2 chart or diagram. A diagram with the shape of a matrix, representing functional or physical interfaces between system elements.
NIC	Network Interface Controller, also known as LAN adapter or network interface card. A computer hardware component that connects a computer to a computer network.
nVidia Titan X	The Graphics Processing Unit in the Argos Machine.

NVMe	Non-volatile memory express. A specification that allows a solid-state drive to make effective use of the high-speed of a PCIe bus in the computer.
Oculomotor nerve	One of the nerves that controls the eyes movements and the ability to focus.
Oculus	The company behind the virtual reality goggles used in Project Argos.
OpenGL	A cross language, cross platform API. Typically used to interact with the GPU.
OpenUP	Project management model used in Project Argos.
Overlay graphic	Graphic added to viewed picture.
PBO	Pixel buffer object used by the OpenGL library. A Buffer Object that is used for asynchronous pixel transfer operations is called a Pixel Buffer Object.
Pipe-filer	A software architecture style used in Project Argos.
Pleora	The company behind eBus SDK.
PoE	Power over Ethernet. A description of any of several standardized or ad-hoc systems which pass electrical power along with data on Ethernet cabling.
Proprioceptive system	Located in the human brain, and work together with the vestibular system.
RAID 0	Redundant array of independent disks. A storage virtualization technology that combines multiple physical disk drive components into a single logical unit for the purposes of data redundancy, performance improvement or both.
Real-time	As in computing. Describes software and hardware systems that has some kind of real-time constraint, for example from an event occurs to the system response. Distinguish-reporting, depicting or reacting to event at the same rate and sometimes at the same time as they unfold.
Rift	VR HMD designed for use with desktop computers.
SDK	Software development kit.

Sequence diagram	An interaction diagram that shows how the processes operate with one another and in what order.
Shader	In computer graphics a shader is a computer program that is used to produce appropriate levels of color and special effects or do post-processing on video.
Simulation Sickness	A variant of motion sickness.
SLI	Scalable Link Interface is nVidias solution to connect two or more video cards together.
SSD	Solid state drive/disk. A solid state storage device.
Stakeholder	A person, group or organization with an interest in the project.
Stereoscopic images	Technique for creating the illusion of depth on an image.
Tamron	Company manufacturing photographic lenses, optical components and commercial/industrial-use optics.
Task	An activity that needs to be accomplished within a defined period of time or by a deadline.
Throughput	The rate of which something can be processed.
TinyArgos	The software in Project Argos.
TinyArgos 2.0	The version of the software that's being developed by our bachelor group.
TinyXML2	An OS independent XML parser for C++.
Topology	Network topology. A "map". The arrangement of the various elements of a computer network.
UML	Unified Modeling Language. A general-purpose modeling-language in the field of software engineering. It gives a standard way to visualize the design of a system. Project Argos uses multiple ULM-diagrams.
Use-case	An UML-diagram. A list of actions/event steps defining the interactions between an actor and a system.
Vection	The human illusion of self-motion.
Vergence	A vergence is the simultaneous movement both eyes in opposite directions to obtain or maintain single binocular vison. (Use both eyes to see)

Vertex	In computer graphics a vertex is a data structure that describes attributes as the position of a 2D or 3D point.
Vestibular system	Part of the human balance system, located in the inner ear.
Vestibular-ocular reflex	A reflex eye movement that helps stabilizes what the eyes see. If the head moves to the right, the eyes will move the left, to stabilize vison.
Vignetting	In the sense of photography and optics. Reduction of an image's brightness or saturation at the periphery compared to the image center.
Virtual reality	A computer-simulated environment designed to model real or imaginary worlds.
Visual Studio	Microsoft Visual Studio. An integrated development environment (IDS). The program we develop the software in.
VOR	See vestibular-ocular reflex.
VR	Short for Virtual Reality, see virtual reality for more.
WBS	See work breakdown structure.
Work Breakdown Structure	A deliverable-oriented decomposition of a project into smaller components. It organizes the group’s work into manageable sections.
Xeon	A brand of x86 microprocessors designed and manufactured by Intel Corporation.
XML	Extensible Markup Language. A markup language that defines a set of rules for encoding documents in a form that is both human-readable and machine-readable.

Gruppekontrakt 1.0

Skrevet av: Ingvild Damtjernhaug
15.01.2016

Bachelor project 2016 at
Faculty for Technology and Maritime Sciences,
University College of Southeast Norway
Project owner:
Kongsberg Defence Systems

HSN

KONGSBERG

1. Introduksjon

Dette dokumentet inneholder en bindende avtale mellom gruppemedlemmene i bachelorgruppe 2016-11. Gruppemedlemmene (heretter kalt «medlemmene») har utarbeidet kontrakten i fellesskap, og kontrakten er gjeldende gjennom hele bachelorprosjektperioden.

2. Medlemmene og rollefordeling

Gruppe 2016-11 «Project Argos» består av

Prosjektleder:	Leiv Fredrik Berge
Lead Developer:	Mathias Havdal
Analytiker:	Trond Egil Hammer
Dokument ansvarlig:	Ingvild Damtjernhaug
Arkitekt:	Thomas Hansen
Tester:	Morten J. Barbala

3. Avtalen

§1 Demokrati
Alle avgjørelser skal baseres på flertallet i gruppa. Medlemmene bør likevel prøve å komme fram til fullstendig enighet.
§2 Plikter
Det er møte- og forberedelsesplikt for alle. Ved sykdom eller annen gyldig fraværsgrunn skal ett eller flere av de andre medlemmene varsles så tidlig som mulig. Medlemmene forplikter seg også til å levere avtalt arbeid til rett tid, eller si fra i god tid hvis forsinkelser oppstår. Medlemmene forplikter seg til å følge opp ansvarsområdene som følger av rollefordelingen.
§3 Arbeidstider
Før påske vil arbeidet med bachelorprosjektet legges til tirsdag, halv dag onsdag, torsdag og fredag. Fra og med påske vil all alle fem arbeidsdager settes av til prosjektet. Medlemmene skal, om ikke annet er avtalt, være til stede på arbeidsrommet i «kjernetiden» mellom kl.09.00 og 15.00.
§4 Møtetider
Møtetider skal avtales slik at ingen i gruppen hindres i forberedelser eller oppmøte. Alle medlemmer plikter og stille klar og forberedt til det daglige «stand-up-møte» kl. 09.45.
§5 Tvister
Uenigheter og problemer bør bli behandlet og løses internt i gruppen. Hvis dette slår feil, skal veileder kontaktes.
§6 Arbeidsdeling
Alt arbeid skal deles så likt og rettferdig som mulig. Alle har rett til å påpeke forhold de mener er urettferdige. Totalt må hvert medlem bruke ca. 500 timer på prosjektet. Dette inkluderer alt relatert arbeid, slik som f.eks. møter med oppdragsgiver og veileder. Det skal føres individuelle timelister som dokumenterer tidsbruken.
§7 Samarbeid

I en gruppe gjelder en for alle, alle for en. Det er viktig at alle forsøker å dele tid, arbeid, erfaringer og kunnskap. Det er en selvfølge å hjelpe til hvis noen står fast.

§9 Kontraktsbrudd

Alvorlige og/eller gjentatte regelbrudd kan føre til eksklusjon fra gruppen. Dette må i så fall skje i samråd med veileder og skolen.

_____
Sted og dato

_____	_____	_____	_____
Leiv Fredrik Berge	Ingvild Damtjernhaug	Morten J.Barbala	Mathias Havdal

_____	_____
Thomas Hansen	Trond Egil Hammer