

Sensur av hovedoppgaver

Universitetet i Sørøst-Norge

Fakultet for teknologi og maritime fag



Prosjektnummer: **2018-08**

For studieåret: **2017/2018**

Emnekode: **SFHO3201-1 17H Bacheloroppgave**

Prosjektnavn

Argos 2.0

Utført i samarbeid med: Kongsberg Defence and Aerospace.

Ekstern veileder: Alexander Gosling

Sammendrag: Fortsettelse fra tidligere bachelorprosjekt fra 2016 og løpende sommerprosjekter, hvor ideen er å kunne se sine ytre omgivelser i et armert kjøretøy via kameraer og VR-briller. Argos 2.0 sin oppgave er å forbedre og videreutvikle dette for å få en løsning som har bedre ytelse, brukervennlighet og lettere å oppdatere til nye teknologier.

Stikkord:

- Real-time mixed reality
- Unreal Engine framework
- Research and development

Tilgjengelig: DELVIS. Kildekode og lavnivå arkitektur er ikke offentlig tilgjengelig.

Prosjekt deltagere og karakter:

Navn	Karakter
Steffen Amandus Nielsen	
Magnus Eiterstraum Muri	
Henrik Andreas Gjestvang	
Fredrik Wang Kåsin	
Andreas Grimestad Holm	
Vebjørn Simensen Tunold	

Dato: 8. Juni 2018

Radmilla Juric
Intern Veileder

Karoline Moholth
Intern Sensor

Alexander Gosling
Ekstern Sensor

AR GOS

2.0

The logo features the letters 'AR GOS' in a bold, black, sans-serif font. The 'A' has a '2.0' subscript. A teal-colored globe icon, composed of a network of white dots and lines, is positioned between the 'G' and 'O'. A thin teal line with circular endpoints connects the top of the 'G' to the top of the globe, and another line connects the bottom of the globe to the bottom of the 'S'.

Reality Made Virtual

21. May 2018

Andreas Holm
Fredrik Wang Kåsin
Henrik Andreas Gjestvang
Magnus Eiterstraum Muri
Steffen Amandus Nielsen
Vebjørn Simensen Tunold



KONGSBERG



Universitetet
i Sørøst-Norge

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

Table of contents

1 Document overview	5
1.1 List of Figures	5
1.2 List of Tables	9
1.3 Group Overview	12
2 Project Vision	13
2.1 Project participants	14
2.2 Why is this project needed	14
2.3 Our project	16
2.4 Stakeholders	21
3 Project Model	23
3.1 Introduction	24
3.2 Methodology	25
3.3 Scrum Tools	32
3.4 Our scrum approach	34
4 Requirements	37
4.1 Use Cases	38
4.2 System Requirements	44
5 Test Specifications and Results	53
5.1 Test procedure	54
5.2 Test Overview	55
6 Risk Analysis	74
6.1 Risk management in Scrum	75
6.2 DoD Risk Matrix	82
7 Architecture	83
7.1 System context	84
7.2 System Architecture	85
7.3 Software architecture	88
8 Integration	97
8.1 Unreal Engine	98
8.2 Insta 360	107
8.3 GigE Vision	115
8.3.2.1.1 Pixel Format	118
8.3.4 Future improvements	131
8.4 VR-Headgear	135
8.5 Tracking	139
8.6 Marking (TerraLens)	154
8.7 Marking (OpenStreetMap)	159
8.8 GUI-overlays	172
8.9 GPS	196



8.10 Software implementation diagram	207
9 Conclusion	208
9.1 Overview	209
9.2 Contribution.....	210
9.3 Related Work	210
9.4 Future Work	211
10 Sources	213
Appendix – A: User Manual	219
1 How to start the application	220
2 How to control the application	220
3 How to use the main menu.....	221
4 How the User Interface works in runtime.....	222
5 How to exit the application	223
6 Navigating the solution.....	223
Appendix – B: Project Plan.....	227
1 Overview.....	228
2 Initial project plan description	228
3. Initial project Gantt diagram	232
4. Result Project Gantt diagram	235
4. Conclusion	236
Appendix – C: Administrative Sprint Report.....	237
1 Administrative Sprint 1	238
2 Administrative Sprint 2	242
3 Administrative Sprint 3	245
4 Administrative Sprint 4	248
Appendix – D: Technical Sprint Report	251
1 Technical Sprint 1	253
2 Technical Sprint 2	255
3 Technical Sprint 3	258
4 Technical Sprint 4	260
5 Technical Sprint 5	264
6 Technical Sprint 6	268
7 Technical Sprint 7	271
Appendix – E: Epic User Stories	275
1 Marking and tracking.....	276
2 Gige vision camera interation.....	277
3 Graphical overlays	278
4 TerraLens Implementation	279
5 Conclusion	280
Appendix – F: User Stories.....	281



Appendix – G: Accounting.....	317
Appendix – H: Time Sheet	321
Appendix – I: Glossary.....	323

1 Document overview

This is the final documentation for the Argos 2.0 bachelor project 2018.

Project website: www.projectargos.net

Project blog: home.usn.no/web-gr8-2018

1.1 List of Figures

Figure 1: Timeline for Argos	17
Figure 2: Use case diagram.....	19
Figure 3: Scum process model methodology.....	24
Figure 4: Time Estimate in Jira	27
Figure 5: Test Picture	30
Figure 6: Jira sprint.....	32
Figure 7: Burn up chart.....	33
Figure 8: Use case diagrams.....	41
Figure 9:High level System architecture	84
Figure 10: System Architecture Diagram.....	87
Figure 11: Use Case Diagram	88
Figure 12: MVC sequence diagram	90
Figure 13: View Live Video Sequence Diagram	91
Figure 14: Record Video Sequence Diagram	92
Figure 15: Mark Targets Sequence Diagram.....	93
Figure 16: Track targets Sequence Diagram	94
Figure 17: GUI overlays Sequence Diagram	95
Figure 18: View recording Sequence Diagram	95
Figure 19: Software Architecture Diagram for computing intensive part of System Architecture (Figure 9).....	96
Figure 20: Argos 2.0.build.cs file	99
Figure 21: Blueprint Example	101
Figure 22: 360° Camera Component.....	108
Figure 23: Cube mapped projection	109
Figure 24: Equirectangular projection	109
Figure 25: Class diagram for viewing live video through the 360° camera	112
Figure 26: Mako camera system architecture.....	116
Figure 27: Device architecture.....	117
Figure 28: Mako camera.....	117
Figure 29: Operators Visual Representation.....	122
Figure 30: Image data stream flowchart	122
Figure 31: View Recording Video Datastream.....	123
Figure 32: Live Stream and Recording Class Diagram.....	124
Figure 33: View Recording Class Diagram	125
Figure 34: Camera startup routine.....	127
Figure 35: AquireImage Sequence Diagram.....	128
Figure 36: Recording Enabled Sequence Diagram	129



Figure 37: Showing tracking's role in the Argos application as well as its dependencies 140

Figure 38: Tracking activated displaying blue diamonds to show direction of targets 141

Figure 39: Class diagram for tracking 143

Figure 40: Sequence diagram for the UpdateBuffer function 145

Figure 41: Red lines show geocentric coordinates relative to each other, green line goes through center of earth and null island (coordinates 0,0) 146

Figure 42: Unit vectors from one coordinate to another 146

Figure 43: Left image shows earth rotated with the same coordinates now on top. Right image shows new unit vectors between the two coordinates. 147

Figure 44: Final relative unit vectors 147

Figure 45: Text displaying information about the target 151

Figure 46: Architecture view of TerraLens 155

Figure 47: Architecture View of component 160

Figure 48: OSM file export example 161

Figure 49: .OSM file opened in VS2017 IDE 161

Figure 50: OSM conversion flow chart 162

Figure 51: StreetMapActor in UE4 worldspace 163

Figure 52: Class diagram for Use Case 3 164

Figure 53: Flow chaw for coordinate conversion between OSM/UE4 165

Figure 54: Sequence diagram for the GenerateUECoordinates function 166

Figure 55: Sequence diagram for the UpdateArgosPosition function 166

Figure 56: Sequence diagram for the InsertTargetsFromBuffer function 167

Figure 57: UE4 editor for ArgosVehicleActor 168

Figure 58: Minimap in VR scene 169

Figure 59: System architecture GUI 173

Figure 60: UMG UI Designer 174

Figure 61: GUI-design 175

Figure 62: Compass bar 176

Figure 63: Mini-map Image 176

Figure 64: Footer 177

Figure 65: Final prototype widget 177

Figure 66: Operator in world space 178

Figure 67: Component hierarchy of ArgosOperator 179

Figure 68: GUI-overlays class diagram 180

Figure 69: Sequence diagram BindWidget 181

Figure 70: isVR boolean 183

Figure 71: Final prototype Compass 184

Figure 72: Sequence Diagram SetCompassRotation 184

Figure 73: Target information box 185

Figure 74: GetTargetInView Sequence Diagram 185

Figure 75: Mini map widget 186

Figure 76: Main Meny widget 187

Figure 77: Argos UI Widget 187

Figure 78: Menu Actor in World Space 188

Figure 79: Transition between maps 189

Figure 80: Menu system flowchart 190

Figure 81: Argos GUI flow chart 191

Figure 82: Argos UI in Runtime 191



Figure 83: Red Dot Widget Interaction Component	193
Figure 84: Widget Interaction Component	193
Figure 85: Blueprint for Interaction	194
Figure 86: IsVR boolean	195
Figure 87: GPS in the system architecture	197
Figure 88: Class Diagram GPS	199
Figure 89: Show the dataflow for the GPS data.....	200
Figure 90: Class diagram for read GPS logs	202
Figure 91: Dataflow while reading form a GPS log	203
Figure 92: Software implementation diagram. Full image is added as attachment.	207
Figure 93: Play the application	220
Figure 93: Red dot indicates mouse cursor	220
Figure 94: "Shift" functions as a mouse click	220
Figure 96: The Main Menu.....	221
Figure 97: Press "M" on the keyboard to activate ArgosUI Menu	222
Figure 98: Argos UI Menu will appear like this.....	222
Figure 99: Targets tracked as blue squares	222
Figure 100: Project schedule diagram	229
Figure 101: Initial Project Gantt Diagram.....	232
Figure 102: Final Gantt diagram for project Argos 2.0	235
Figure 103: Sprint Report	238
Figure 104: Burn Up Chart.....	239
Figure 105: Velocity Chart	239
Figure 106: Sprint Report	242
Figure 107: Burn Up Chart.....	243
Figure 108: Velocity Chart	243
Figure 109: Sprint Report	245
Figure 110: Burn Up Chart.....	245
Figure 111: Velocity Chart	246
Figure 112: Sprint Report	248
Figure 113: Burn Up Chart.....	249
Figure 114: Velocity Chart	249
Figure 115: Sprint Report	253
Figure 116: Burn up Chart	253
Figure 117: Velocity Chart	254
Figure 118: Sprint Report	255
Figure 119: Burn Up Chart.....	256
Figure 120: Velocity Chart	256
Figure 121: Sprint Report	258
Figure 122: BurnUp Chart.....	258
Figure 123: Velocity Chart	259
Figure 124: Uncompleted Tasks.....	259
Figure 125: Sprint Report	261
Figure 126: Burn Up Chart.....	261
Figure 127: Velocity Chart	262
Figure 128: Sprint Report	264
Figure 129: BurnUp Chart.....	264
Figure 130: Velocity Chart	265
Figure 131: Sprint Report	268
Figure 132: BurnUp Chart.....	269



Figure 133: Velocity Chart 269

Figure 134: Sprint Report 272

Figure 135: Burn Up Chart..... 273

Figure 136: Velocity Chart 273

Figure 137: Epic report for Marking and Tracking..... 276

Figure 138: User stories which makes up Marking and Tracking 276

Figure 139: Epic report for GigE vision integration 277

Figure 140: User stories which makes up GigE vision integration 277

Figure 141: Epic report for Graphical Overlays..... 278

Figure 142: User stories which makes up graphical overlays 278

Figure 143: Epic report for TerraLens Implementation 279

Figure 144: User stories for TerraLens implementation..... 279

Figure 145: Sprint time sheet diagram..... 322

Figure 146: Pie chart for Argos 2.0 workload..... 322



1.2 List of Tables

Table 1: View live video	41
Table 2: Record video.....	41
Table 3: Mark targets.....	42
Table 4: Track target real time	42
Table 5: GUI overlays	42
Table 6: View recording	43
Table 7: View user recording	43
Table 8: Priority explanation	44
Table 9: Classification explanation	44
Table 10: State explanation	44
Table 11: Requirement template description	44
Table 12: Functional requirements	46
Table 13: Non-Functional requirements.....	50
Table 14: Constraints	51
Table 15: Traceability to tests.....	52
Table 16: Standard table for listed tests	54
Table 17: Standard table for test report	54
Table 18: Functional test 1	55
Table 19: Functional test 1 result.....	55
Table 20: Functional test 2	56
Table 21: Functional test 2 result.....	56
Table 22: Functional test 3	57
Table 23: Functional test 3 result.....	57
Table 24: Functional test 4	58
Table 25: Functional test 4 result.....	58
Table 26: Functional test 5	59
Table 27: Functional test 5 result.....	59
Table 28: Functional test 6	60
Table 29: Functional test 6 result.....	60
Table 30: Functional test 7	61
Table 31: Functional test 7 result.....	61
Table 32: Functional test 8	62
Table 33: Functional test 8 result.....	62
Table 34: Non-functional test 1	63
Table 35: Non-functional test 1 result	63
Table 36: Non-functional test 2.....	63
Table 37: Non-functional test 2 result	63
Table 38: Non-functional test 4.....	64
Table 39: Non-functional test 4 result	64
Table 40: Non-functional test 5.....	64
Table 41: Non-functional test 5 result	64
Table 42: Non-functional test 6.....	65
Table 43: Non-functional test 6 result	65
Table 44: Non-functional test 7	65
Table 45: Non-functional test 7 result	65
Table 46: Non-functional test 8.....	66
Table 47: Non-functional test 8 result	66



Table 48: Non-functional test 9.....	66
Table 49: Non-functional test 9 result.....	66
Table 50: Non-functional test 10.....	67
Table 51: Non-functional test 10 result.....	67
Table 52: Non-functional test 11.....	67
Table 53: Non-functional test 11 result.....	67
Table 54: Non-functional test 12.....	68
Table 55: Non-functional test 12 result.....	68
Table 56: Non-functional test 13.....	68
Table 57: Non-functional test 13 result.....	68
Table 58: Non-functional test 14.....	69
Table 59: Non-functional test 14 result.....	69
Table 60: Non-functional test 15.....	69
Table 61: Non-functional test 15 result.....	69
Table 62: Non-functional test 16.....	70
Table 63: Non-functional test 16 result.....	70
Table 64: Non-functional test 17.....	70
Table 65: Non-functional test 17 result.....	70
Table 66: Non-functional test 18.....	71
Table 67: Non-functional test 18 result.....	71
Table 68: Non-functional test 19.....	71
Table 69: Non-functional test 19 result.....	71
Table 70: Non-functional test 21.....	72
Table 71: Non-functional test 21 result.....	72
Table 72: Constraints test 1.....	73
Table 73: Constraints test 2.....	73
Table 74: Hardware faults.....	76
Table 75: Software faults.....	78
Table 76: System lifetime risks.....	79
Table 77: Human errors.....	80
Table 78: Programmatic risks.....	81
Table 79: DoD Risk Matrix. Green is low risk, yellow is moderate, and red is high risk.	82
Table 80: Explanation of the likelihood and consequence levels. Numbers taken from the DoD risk, issue and opportunity management guide [14].....	82
Table 81: Comparison of capture cards.....	110
Table 82: Comparison VR headgear.....	136
Table 83: Description for GNGGA message [50].....	205
Table 84: Description of GNRMC message [50].....	205
Table 85: User Workload.....	240
Table 86: User Workload.....	244
Table 87: User Workload.....	247
Table 88: User Workload.....	250
Table 89: User Workload.....	254
Table 90: User Workload.....	257
Table 91: User Workload.....	259
Table 92: User Workload.....	262
Table 93: User Workload.....	265
Table 94: User Workload.....	270
Table 95: User Workload.....	274



Table 96: Accounting for the products bought by KDA during our project 318
Table 97: Accounting for the products bought by the bachelor group during our
project..... 319



1.3 Group Overview

Group 8-2018 Argos

Computer science group



Henrik Andreas Gjestvang

Henrik.gjesvang@gmail.com

+47 404 89 161

Developer - Product Owner



Steffen Amandus Nielsen

Steffen-94@hotmail.com

+47 954 09 586

Developer - Scrum Master



Andreas Grimestad Holm

Anholm@live.no

+47 938 60 756

Lead Developer



Fredrik Wang Kåsin

Fredrik.kasin@outlook.com

+47 454 42 297

Developer - Document master



Magnus Eiterstraum Muri

MagnusEMuri@gmail.com

+47 936 24 107

Developer - Hardware responsible



Vebjørn Simensen Tunold

Vebjorn.Tunold@gmail.com

+47 476 57 144

Developer - QA tester

Internal supervisor: **Radmila Juric** | 31 00 98 74 | radmila.Juric@usn.no

External supervisor: **Alexander Gosling** | 41 66 78 63 | alexander.gosling@kongsberg.com



2 Project Vision

This chapter describes the initial vision for the Argos 2.0 bachelor project, why the project is needed, and the scope regarding the vision for the end product that the group had at the start of the project. It also provides information about stakeholders and the history of **Project Argos**.

Describes

- Project Participants
- Why the products are needed
- Argos 2.0 project
- Argos 2.0 Stakeholders

Project websites:

www.projectargos.net

<https://home.usn.no/web-gr8-2018/>

Bachelor project 2018 at
Faculty of Technology, Natural Sciences and Maritime Sciences
University of Southeast Norway
Project owner:
Kongsberg Defence Systems



KONGSBERG



2.1 Project participants

The given assignment has been executed by six computer engineering students from the University of Southeast Norway(USN). Three of the students specialized in virtual systems and three in embedded systems.

The project assignment was given by Kongsberg Defense and Aerospace (KDA), which is the defense-branch of Kongsberg Gruppen. Kongsberg Gruppen is an international technology group that delivers technological systems and solutions, mainly to customers in oil, gas, navy, defense and aerospace. [1]

2.2 Why is this project needed

“in regione caecorum rex est luscus.”
- Desiderius Erasmus

The Latin proverb loosely translates to “the person with the mildest disadvantage is dominant”. If you consider boxing, keeping yourself protected behind your guard while still maintaining control of a situation using your eyes, has ever remained a challenge. Keep your guard too high and you will lose control of your opponent quickly. Keep it too low and you will get hurt.

The same can be said about modern warfare using Armored Fighting Vehicles. No matter the amount of protective capability your vehicle possesses, you will forever be at a disadvantage if it hampers your ability to correctly assess the situation you currently find yourself in.

The solution used in most light vehicles is to reduce the protection on vital components such as the windshield and windows by usage of ballistic glass. While ballistic glass offers the same visibility capabilities as ordinary glass, it does not offer the same protective capabilities that regular armor possesses, as it is only meant to withstand shrapnel and small arms fire.



The solution used in most heavier vehicles is to rely much more on the armor for protection at the cost of the operator's ability to get a comprehensive view of the surroundings, as most of the cognitive feedback the operator gets is through usage of periscopes, or in some very modern systems, a screen connected to cameras mounted outside the vehicle.

Maneuvering an Armored Fighting Vehicle (AFV) is not to be taken lightly. Not only must one take heed to the situations they are meant for, they are also heavy objects with a powerful engine capable of moving at relative high speed, and impact can cause devastating damage to any obstacle in its path, whether it be civilian structures, cars or people. Damage to the AFV itself is also troublesome as it can in worst case leave operators as sitting ducks in a vehicle unable to properly function when it is most needed. Therefore, the reduction of visual feedback has sometimes resulted in situations where the operator must expose themselves to hostile elements, thereby forfeiting any protection the vehicle has to offer, to regain control of the situation they find themselves in.

To summarize, the problem is that the personnel inside an armored vehicle does not have the required overview of their surroundings while staying safe inside the vehicle.



2.3 Our project

Argos is a continuation of a previous project and has been revisited for further development to improve the given system. The main goal of Argos was continuing to develop the solution that enables “see through armor” using “real time virtual reality”. Previous objectives included giving the crew of any AFV the ability to observe their surrounding environment from the inside of the vehicle through the usage of cameras, sensors and virtual reality headsets rather than ballistic glass and/or periscopes, to achieve the best possible situational awareness while still utilizing the maximum protective capabilities of the vehicles armor during any given mission. A working prototype of the described system did exist at project startup and was expected to be improved during the bachelor thesis of 2018.

2.3.1 Problem description

The work done on Argos earlier was developed for Windows 7. Windows 7 is an operating system developed for computers by Microsoft in 2009. In 2015 Microsoft developed Windows 10; roughly put, an updated and improved version of Windows 7. According to the windows lifecycle sheet [2], Windows 7 will stop receiving support and updates in 2020, which will leave the system outdated. Because of this, it was essential to update Argos to Windows 10, to ensure that the software is supported and runnable in the future. This also made room for general improvements of tools and software.

Oculus Rift is no longer the desired virtual reality (VR) product and Argos shall migrate to the open source virtual reality (OSVR) headset instead to support several VR platforms [3] rather than one particular headgear. (See chapter 4.)

Additionally, the project owner at KDA had expressed wishes for adding support for dynamic overlays, making it possible for the user to designate targets in the surrounding terrain. A reimagining of the engine that runs the application was welcome, as earlier attempts at creating extensions to the current Argos application has proven to be both difficult and time consuming.



All overlays were at startup statically drawn on the GUI and on the fly design changes was hard to implement unless handled by a programmer well versed in the Argos application. Parts of the codebase was based upon 32bits Win32 architecture and had to be migrated to 64bit Win64 to ensure future support from OS provider. There was also no applied compression to the video files stored on the high-speed disk storage, making them enormous in size.

Certain parts of the graphical elements were handled by a library that is outdated by over a decade and did not utilize the full potential of modern hardware. Our vision at startup was to build an updated prototype that should handle these issues and aim to identify challenges both future and present.

2.3.2 Project History

Argos [4] was first initiated as a student project at Kongsberg Defense and Aerospace by an interdisciplinary team of computer science, cybernetics, electrical and mechanical engineering students in the summer of 2015. It was continued as a bachelor thesis in 2016 where the main focus was to implement cameras, Oculus rift support and the ability to store recording of the captured video stream. A map, GPS and a compass were implemented during the student projects in 2016 and 2017. The development of Argos has gone through several development cycles since its initial conception and every iteration of Argos has improved upon the last.

Timeline for work on Argos since 2015

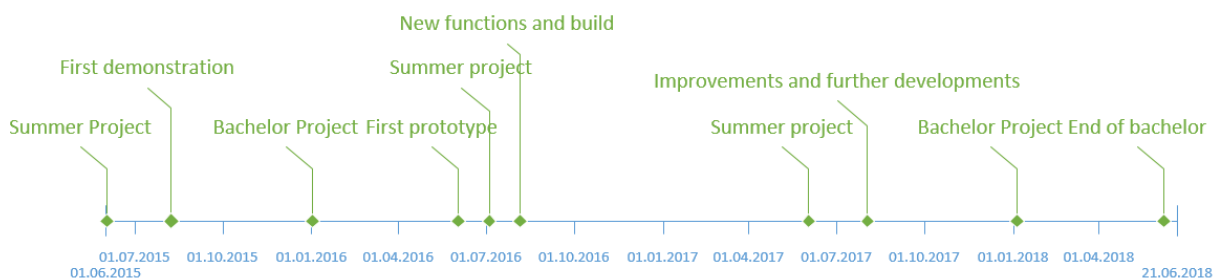


Figure 1: Timeline for Argos

This timeline presents which group has worked at each iteration from the projects' start up in the summer 2015, to our bachelor project is scheduled to be finished, summer 2018.



2.3.3 Project State

The state of Argos at handover in January 2018 was a functional prototype of how the system was intended to work. It consisted of the camera and sensor subsystem, the network, computer hardware, VR headset and the Argos software. The key software component called “TinyArgos” integrated all the Hardware parts and created an interface between the cameras and the VR headset (Oculus rift). A map, compass and a GPS had also been implemented in the initial prototype.

This gave the users of Argos the intended ability to observe through the VR headset from the inside of a vehicle and get a somewhat seamless panoramic view of their surroundings by using the cameras and sensors placed outside the vehicle. The camera feed was stored on a high-speed disk storage, and a map, compass and GPS were visible on the user interface.

As a product in alpha state it fulfills its purpose, but the panoramic view was far from perfect and felt displeasing to look at.

2.3.4 The Initial Project Scope

This is the initial project scope for the project at the start of the project:

For this bachelor thesis, we have continued to develop the Argos software to solve the current problems stated in chapter 2.3.1.

The main focus is to move the entire codebase to Windows 10 and make the product compatible with Open Source VR, rather than Oculus DK2.

The group will develop overlays and updates to the Graphical User Interface. The group shall consider and apply general improvements to the product, while always have in mind that the software for our product shall be lightweight enough to run on a laptop for demonstrational and presentational purposes. We want to remove the custom Argos game engine and implement Argos into a high end commercial game engine to test the viability of using state-of-the-art technology for the applications foundation.



Project Argos, TinyArgos 3.0:
Exclusively x64 architecture
Integrate support for OSVR
Reimagine application engine
Integrate support for Terralens
Graphical User Interface improvements

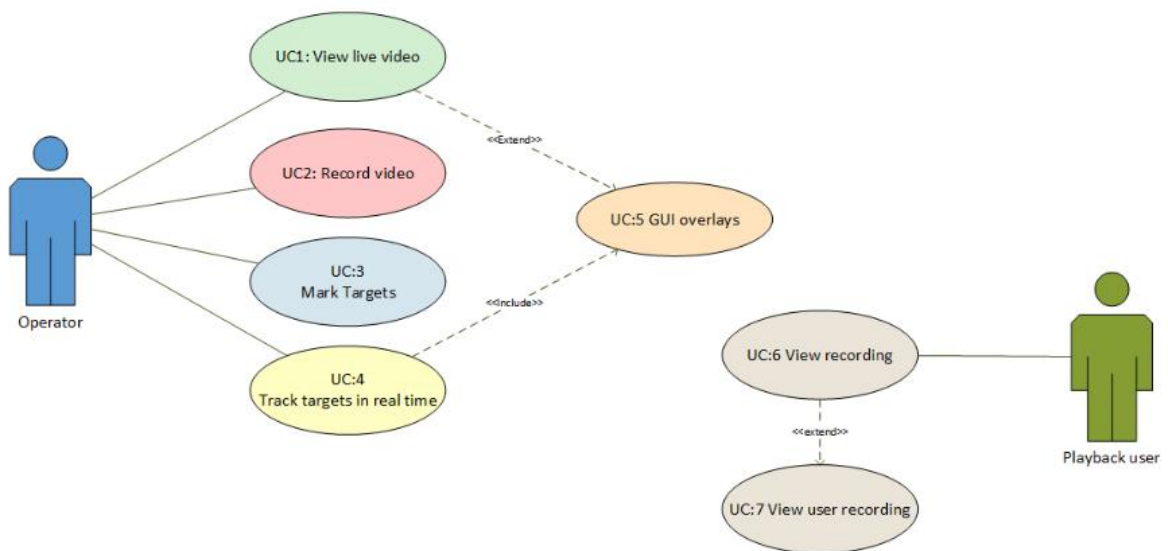


Figure 2: Use case diagram

This Use Case Diagram shows the different ways the Operator and the person that is going to view the recordings, the Playback User, can interact with the Argos system.



2.3.5 Project Goals

The main goals of this project were:

A functioning updated prototype of Project Argos at the end of the bachelor thesis. To solve the problems regarding the first prototype.

A new version of Project Argos that is based on x64 architecture exclusively and can handle both current and near future hardware and software improvements in the industry without having to rework the entire system. This will make the project much easier to work with in the future.

An updated prototype that will serve as a proof of concept and a technology showcase for KDA and facilitate a solid foundation for future development.

We aimed to make considerable amounts of improvements to the foundation of Argos and leave the project in a state where further extensions are easily fitted with the application. The ideal end state was a foundation where heavy duty task can be handled by senior developers, while easier task can be handled by junior developers or even personnel with design experience.



2.4 Stakeholders

The stakeholders of our system are KDA, the project owner, USN and the end users, both military and civilian.

2.4.1 Stakeholder requirements

Each stakeholder had different requirements for this project, depending on how they are connected to the project.

KDA had the following requirements (See chapter 4):

KDA requires the group to further develop the existing Argos application. This means to change the VR support from Oculus Rift to OSVR. To update the codebase to support Windows 10. KDA require more support for extensions such as TerraLens and a refresh rate that is fast enough to use in a moving vehicle. KDA requires well documented and structured code while the group complies with ethical guidelines.

The University (USN) had the following requirements:

USN requires the group to deliver a well-documented project while managing the workload with regards to time frame and deadlines and complies with the ethical guidelines connected to the project.

The end users had the following requirements:

The end user requires a system that is stable, comfortable, well-documented, safe and easy to use while it can be bought to an affordable price.





3 Project Model

Project models are an important tool in every project. This chapter describes the Scrum project model and its methodology. It also provides information on how the Argos 2.0 group used Scrum, and why we chose this model as our project model for Argos 2.0 bachelor project.

Describes

- Methodology
- Scrum related tools
- How Argos 2.0 applies scrum.



3.1 Introduction

Argos 2.0 is a research and development project and the project model would have to fit that purpose. A static model such as the waterfall model would therefore not be suitable. We considered Unified Process and Scrum, both good models which adapt agile thinking and iterations, which would be required for the Argos 2.0 project. Rather than completing the project and then showcasing it like waterfall, agile does very small increments and at the end of each increment we deliver value to the customer.

We decided to use Scrum because it provides an overview of tasks and their priority, fits small software groups, and includes powerful tools such as *user stories*, *stand-up* and *burn up charts*, which will be explained in this document.

We started with user stories from the customer. There are typically defines “*As a, I want, so that*” to capture the essence of requested product. These user stories are then put into a product backlog. Each iteration, we pick **user stories** from the product **backlog** and put them into a **sprint**. Daily scrum was applied during the sprint, and when the sprint had reached its deadline, we would have a potentially shippable product.



Figure 3: Scrum process model methodology



3.2 Methodology

To apply scrum correctly, we needed a shared understanding of the scrum methodology in terms of role definitions.

3.2.1 Roles

A Scrum team is a small group of people that work together to achieve the best possible solution a customer's needs. To always stay on the right path regarding the intent of the application, the scrum team has a **Product owner**. This person has a close dialogue with the customer. The **product owner** delivers user stories to our **scrum master**, which manages the process for how information is exchanged, and therefore becomes the facilitator for the agile development team called **scrum team**. [5]

3.2.1.1 Scrum master

The Project Argos 2.0 Scrum master is Steffen Nielsen, and he was responsible for:

- Helping the team determining what can be achieved during a specific period (sprint)
- Helping the team during the daily scrum
- Helping the team to stay focused, and to follow the agile methodology.
- Removing obstacles that are slowing down the team's progress

During the daily meetings the scrum master also asks the team members these three questions to keep the team on the same page

- What did you do yesterday?
- What will you do today?
- Are there any impediments in your way?

Although the title of **Scrum master** sounds powerful, the scrum master is not the project leader and is not held accountable for the outcomes [6]. The team as a whole is responsible for the outcomes.

3.2.1.2 Product owner

The Project Argos 2.0 Product owner is Henrik Gjestvang, and he oversaw:

- The product backlog(s)
- Understand the vision for the product and making user stories, clearly describing what needs to be done
- Organizing the product backlog for the best possible efficiency in achieving our goals
- Making sure we deliver the product that the business wants at the correct time
- Communication with the different stakeholders



3.2.2 User Stories

User stories are features that the product owner want to be developed, that provide value to the customer. A user story should be written from the perspective of the person who desires the new functionality, and should include **who**, wants **what functionality**, for **which purpose**.

To produce a successful user story, Argos 2.0 used two guidelines which describes how we should go forth when creating one. These guidelines make it easier to verify that the user story is properly made, implemented, and tested. One of these guidelines are **INVEST** [7].

- **I**ndependent (each user story stands by its own)
- **N**egotiable (the way the user story is going to be implemented, discussions in implementation. Don't make it too detailed, so discussion can't flow)
- **V**aluable (must create value for the customer)
- **E**stimable (must be able to estimate the time and effort for the user story)
- **S**mall (small enough to be done in a single sprint)
- **T**estable (must have objective test criteria)

The second guideline is **the 3 C's** of a user story:

- **C**ard (should be small enough to be able to be written on an index card).
- **C**onversation (the item that is used for the theme and the product owner to talk and discuss to make sure that the product owner and the team are on the same page when it comes to what is being developed and how it should look when it's done). [8]
- **C**onfirmation (each user story must have acceptance test on the user story card to be used in the sprint review). [8]



3.2.3 User story size estimation

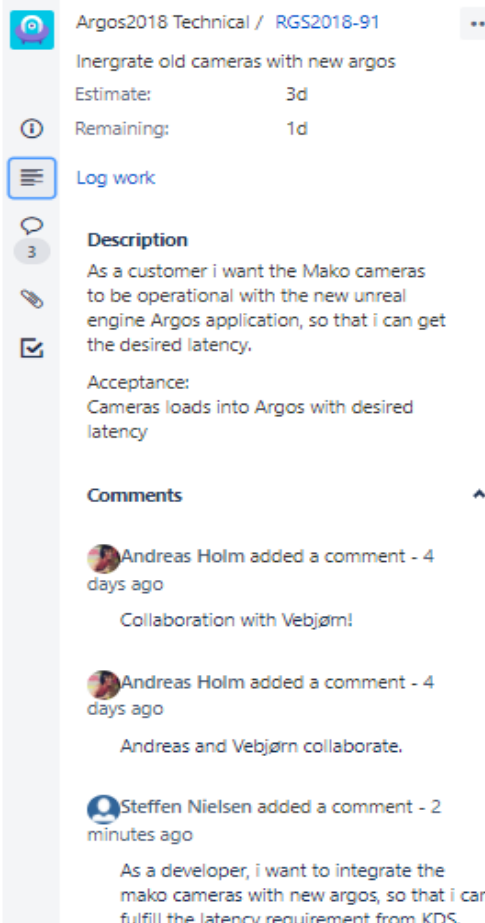
Argos 2.0 used **time tracking** when estimating the amount of work that goes into a user story. This includes **estimated time** and **remaining time**.

3.2.3.1 Estimated time

Estimated time is the time a user story takes to finish in working hours. The workload is calculated based on *the amount of work to do, the complexity of the work and risks and uncertainties in doing the work*. [9]

3.2.3.2 Remaining time

Remaining time is the remaining work estimated in time, that must go into a user story to finish it. This means that the person working on the given user story can let the Argos 2.0 team know how much work remains on the given user story. Our time estimate included everything involved in getting a product backlog item done. That means that our definition of done had to include testing and documentation for that user story before it can be marked as done. Using **time tracking**, we can provide our project- owner with status updates on every user story and keep track of the progress within the Argos 2.0 team. The other approach to user story estimation is **Story points**, which the group tried out but concluded that story point estimation mostly just became an alternative estimation of time.



The screenshot shows a Jira user story page for 'Argos2018 Technical / RGS2018-91'. The story title is 'Inegrate old cameras with new argos'. The 'Estimate' is 3d and 'Remaining' is 1d. There is a 'Log work' button. The 'Description' is: 'As a customer i want the Mako cameras to be operational with the new unreal engine Argos application, so that i can get the desired latency.' The 'Acceptance' criteria is: 'Cameras loads into Argos with desired latency'. There are three comments: two from Andreas Holm (4 days ago) and one from Steffen Nielsen (2 minutes ago). The comments discuss collaboration with Vebjørn and integrating Mako cameras.

Figure 4: Time Estimate in Jira

3.2.4 Themes / Epics

Themes are groups of related user stories. When we think of traditional stories, they all contribute to a common goal or are related but independent. Themes however, may depend on one another, they do not need to encapsulate a specific work flow or be delivered together.

When we encountered a user story that is too large we entitle it as an **“Epic”**. These are large user stories with low priority and was be split into multiple smaller, user stories. Epics are not valued as business value until the entire epic is complete, which means we rarely deliver part stories from an epic. It’s important to remember that a big user story in not necessarily an epic. If the user story can be split up into multiple, independent stories, we do that.

The Argos 2.0 project had 4 epics during its development:



- GigE Vision implementation
- GUI implementation
- Marking & Tracking implementation
- TerraLens/OpenStreetMap implementation

These are functionality which required multiple user stories to accomplish. Detailed information on these epics will be explained in the “Implementation” chapter.

3.2.5 Tasks

A task is the technical work a development team performs to complete a product backlog item. Most tasks are defined to be small, representing no more than a few hours to a day or so of work. [10] Argos 2.0 used tasks to split up user stories into smaller chunks of work. Tasks were necessary to view progress on larger user stories, which would not have been visualized and documented if we did not use them.

3.2.6 Sprint

When we filled up our product backlog with user stories, the Argos 2.0 team picked out which of the stories we wanted to work on based on the priority of the functionality. These user stories from the product backlog are then put into the **sprint backlog**, where some of the selected user stories received designated tasks. We then sat a time, usually from 1-2 weeks where we work to get a working increment of the software based on the selected sprint backlog items. This period is called a **Sprint**.



3.2.6.1 Sprint review

During the sprint review the Argos 2.0 team and stakeholders collaborated on what was done in the sprint and received feedback. Often a sprint review resulted in more user stories from the stakeholders. [11]

Argos 2.0 sprint reviews followed a specific agenda:

- The Argos 2.0 **product owner** explained what product backlog items have been “Done” and what has not been “Done”.
- The Argos 2.0 team discussed what went well during the sprint, what problems they ran into, and how those problems were solved.
- The Argos 2.0 team demonstrated the work that it has “Done” and answers questions about the increment.
- The Argos 2.0 **product owner** discussed the product backlog as it stands. He projects likely target and delivery dates based on progress to date.
- The Argos 2.0 team collaborated on what to do next, so that the sprint review provides valuable input to the next sprint planning.
- Review of the timeline, budget, potential capabilities and marketplace for the next anticipated releases of functionality and capability of the product. [11]

3.2.7 Velocity

Velocity is a tool that helps understand the productivity of the team. It is the amount of work that can be completed in a sprint. It is the sum of the size of completed stories in each sprint. It is an observation, not a prediction. Which means if we work on sprint 5 and try to determine our velocity of this sprint it would be the average of the last 4 sprints.

Velocity was used to answer question like “how many stories should we accept in each sprint? How many sprints do we need to complete a set of stories”? The velocity chart for Argos 2.0 is included in the Appendix part of this documentation.



3.2.8 Testing

Testing is essential in making sure your feature is fulfilling the user stories and quality requested from the stakeholder. In Argos 2.0 we used 3 testing methods:

3.2.8.1 Acceptance test

When we wrote a user story, we defined acceptance criteria for the feature. The criteria must pass the test before the feature could be approved. This is called an acceptance test. We can split an acceptance test into sub-tests, these are: *user*, *end-user*, and *operational acceptance tests*. We aimed to write one general acceptance test to each story including the three sub-tests.

3.2.8.2 Unit test

Unit tests are small tests of specific functions to tell you where your features are failing.

We had multiple unit tests for each user story to make sure that the functionality worked as intended or find where it failed. A set of these tests are names a “test suite”, and we had one test suite for each piece of new shippable features.

3.2.8.3 Integration test

Integration test are where individual units are combined and tested as a group to determine faults in the interfaces and interaction between integrated units. There are a couple of different integration approaches

- Big bang (all or most units are developed before we run integration test)
- Top down (top level units are tested first, uses top down development approach)
- Bottom up (bottom level units are tested first, uses bottom up development approach)
- Sandwich/hybrid (combination of top down and bottom up approaches)

One of the biggest mistakes in group related work was saving the biggest and most challenging user stories to the end because they are usually in the bottom level. We therefore used a Sandwich/hybrid approach to always have the capability to change between bottoms up and Top down approach when it comes to development and integration testing. [12]



Figure 5: Test Picture



3.2.9 Definition of “done”

To make sure that we are delivering features that are truly done, both in quality and functionality, we set a definition of “done” within the Argos 2.0 team. It’s simply a list of activities that adds or verify the value to the product. These can be coding comments, unit testing, integration test, release notes and more. We define our definition of “done” as potentially shippable. We have some guidelines for when features, sprints and releases can be set as “Done”. [13]

- Features are marked as “done” when they pass our acceptance test.
- Sprints are marked as “done” when all the user stories in the sprint backlog are marked as done, or when the sprint iteration runs out of time. The unfinished user stories would then go back into the backlog.
- Releases are marked as “done” when they have passed out test suite, integration tests and documentation requirement.

3.2.10 Daily scrum

The team huddled together each morning and reviewed progress. During the daily meetings, which are called “**scrums**”, the team elaborated to synchronize activities and create a plan for the next 24 hours. We used these meeting to optimize team collaboration and performance by inspecting the work since the last daily scrum and forecasting upcoming sprint work. Our three points for the daily scrum were:

- What did you do yesterday?
- What will you do today?
- Are there any impediments in your way?



3.3 Scrum Tools

Scrum contains tools which helped the scrum team review progress and tasks. These are the tools used during the Argos 2.0 project

3.3.1 Jira

Jira is a project management tool that supports the scrum methodology. It contains agile reports, planning, tracking and management of agile software development projects. We used Jira for our technical and administrative backlog, sprints, sprint reviews, tasks, and burn up charts.

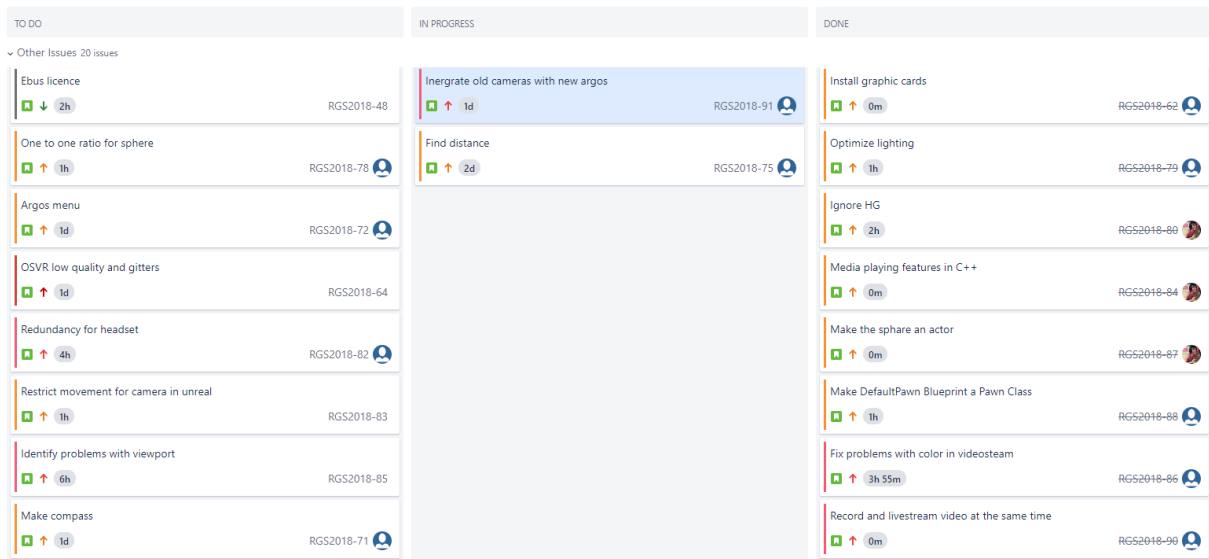


Figure 6: Jira sprint

3.1.2 Burn up charts

As the sprint starts, we map out a burn up chart. The purpose is to provide the Argos 2.0 team members information on how much work remains in the sprint. We list up time passed on the Y-axis and days on the X-axis. The team checked the burn up chart every day to see if we were going on as expected, or if we needed to recalculate. It is a tool that visualized our progress in real time and kept us motivated.

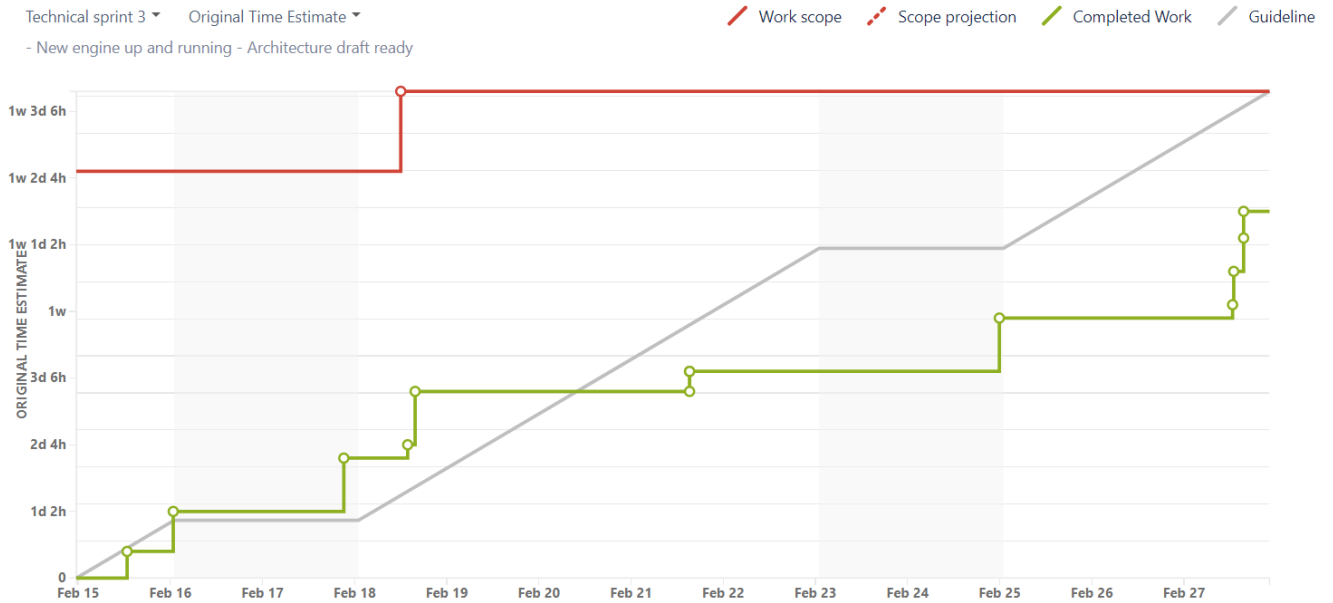


Figure 7: Burn up chart



3.4 Our scrum approach

The following subchapter will contain information on how the Argos group applied scrum.

3.4.1 Creating user stories

We defined user stories from requirements, using user story cards to make sure we defined user stories correctly, with INVEST and the 3 C's in mind, as explained in 3.2.2. We elaborated on *the amount of work to do, the complexity of the work and risks and uncertainties in doing the work* explained in 3.2.3. When the user story fulfilled our requirements, we put it into our product backlog (Jira).

3.4.2 Sprint planning

Every second Thursday we put the most important / valuable user stories into our sprint backlog. We selected user stories based on priority, stakeholders input and the time estimation. We used two kinds of sprints:

- Administrative (1 week)
Administrative sprints contained administrative tasks that needed to be done. These **tasks were not** written as “user stories” (see 3.2.2). They were defined as normal tasks but contained estimated time to estimate the effort needed to complete the task. We did this because it helped us maintain control of the project, and because it's commonly practiced in the industry. [10]
- Technical (2 week).
The sprints included the amount of user stories calculated based on the sprint time and estimated time. The user stories were selected based on priority and sprint review, and feedback from stakeholders.

3.4.3 Define tasks

Then we defined tasks for each user story, following the general rules of task definition, which is that it should not take more than one day of work. Each member in the team was assigned to the tasks they wanted to work on and set the status of that task to “in progress”.



3.4.4 Executing the sprint

Following the sprint as usual, keeping track of the burn up chart every day to stay on track and make changes if needed. The Argos 2.0 scrum master helped the team to reach the sprint goal and make sure challenges were dealt with as they arrived. When we found ourselves in a situation where we either had too much to do, or nothing to do at all, we would restructure the current sprint. We did this by adding user stories to the already existing sprint, or by putting remaining work back into the backlog.

3.4.5 Sprint review

The Argos 2.0 teams Product owner hosted a sprint review at the end of the sprint (Thursday) with our stakeholders (Alexander Gosling) and the rest of the Argos 2.0 team. We also collaborated on what was done and received feedback. We took the feedback into consideration, and sometimes received new user stories we could include in our product backlog. User stories that are NOT completed were put back into the product backlog. We also reviewed our sprint velocity each sprint review and kept track of progress.

This process was repeated with new defined user stories from each sprint review.





4 Requirements

This chapter lists the requirements for the project development of the Argos system for the bachelor project 2018. Argos 2.0 is a research and development project and will therefore contain old relevant requirements as well as new from KDA. Requirements is derived from user stories.

Describes

- User stories for system behaviour
- Functional requirements, non-functional requirements and constraints
- Traceability between requirements and tests



4.1 Use Cases

Use cases are used to visualize the situations and environments the system will be put in and is a useful tool to discover requirements and necessary functionality for the system. They are a big part of the early development of the project. In this chapter the use cases help to describe the necessity and importance of the functional requirements. Each use case has a unique ID for easy traceability in the connections to the functional requirements.

4.1.1 User stories and scenarios

4.1.1.1 Scenario (Live):

Optimal:

The operator needs a larger field of view than the tank periscope provides. The operator puts on the VR goggles and initiates the Argos system. When the system starts it displays the live camera feed of the surroundings in the VR goggles. The operator turns his head to look “through” the walls of the tank giving him/her a complete overview of his/her surroundings. Now that the operator has a full overview of his/her surroundings he/she can now navigate safely through dangerous or tight areas.

User story:

As an operator I want to be able to view live video from within my vehicle so that I can safely manoeuvre through dangerous areas.



4.1.1.2 Scenario (Recording):

Optimal:

The operator is driving with the Argos system active. Suddenly the operator enters an area of interest that needs further investigation by experts. The operator can then start recording the situation they're in so that an expert or an officer may look at it later for review.

User story:

As an operator want to be able to record video so that it can be reviewed at a later time.

4.1.1.3 Scenario (Tracking, marking)

Optimal:

The vehicle is close to an object of interest. A remote operator has gathered the exact location of the object of interest and sends it to the vehicle. The Argos system writes in graphic overlays to mark the object of interest as a target in the VR goggles to show the operator where to look for the object. The Argos system calculates the distance to the target and displays that with the graphic overlay in the VR goggles. The location is marked on a map so that the operator can easily see the relative position of the targets.

User story (Tracking):

As an operator I want to track targets in real time so that I can get detailed information about targets in the area.

User story (Marking):

As an operator I want to have an overview of targets in my area displayed on a map so that I can have full control over my surroundings.

User story (GUI overlays):

As an operator I want to have visual information about the vehicle and targets in the area displayed in the VR-goggles so that I can easily get information when I need it.



4.1.1.4 Scenario (View recording)

Optimal:

The operator has recorded a critical situation. The playback user looks through the full 360-degree recording for any abnormalities. The playback user then notices the operator taking an unusual decision based on what was captured by the camera. The playback user decides to watch the user recording to see exactly what the operator saw in that situation and sees that the operator didn't see the information necessary to make the right decision.

User story (View recording):

As a playback user I want to be able to have a full overview of the surroundings of the system when the video was recorded so that I can see the situation the system was in.

User story (View user recording):

As a playback user I want to be able to see exactly what the operator saw when the video was recorded so that I can tell if the operator took the right decision.



4.1.2 Use case diagrams

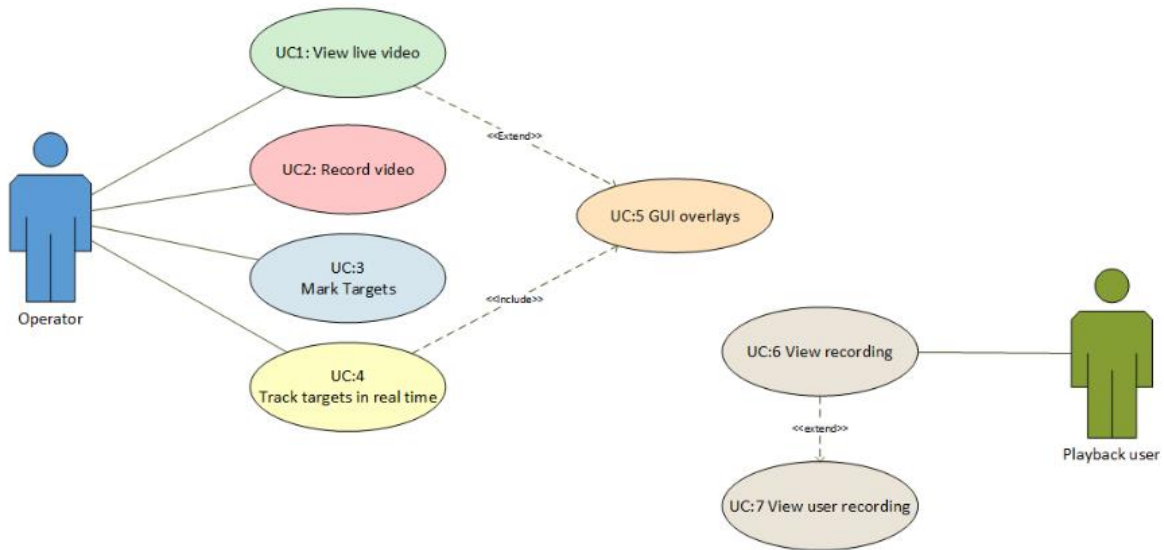


Figure 8: Use case diagrams

4.1.3 Descriptions

View live video:

ID: UC.1
 Actor: Operator
 Goal: Display live video for the operator from camera system.

Actor	System
Turns head to view surroundings	Display camera feed in VR headset for the direction the actor is viewing.

Table 1: View live video

Record video:

ID: UC.2
 Actor: Operator
 Goal: Record video for later viewing.

Actor	System
Push record button	Camera feed will be compressed and recorded.

Table 2: Record video



Mark targets:

ID: UC.3

Actor: Operator

Goal: Display up to multiple targets in a map.

Actor	System
Put in remote coordinates that needs tracking	Mark the targets in the map in the VR goggles

Table 3: Mark targets

Track targets in real time:

ID: UC.4

Actor: Operator

Goal: Track marked targets' relative position and distance.

Actor	System
Activate tracking	Display tracked target, with distance in the direction of the target.

Table 4: Track target real time

GUI overlays:

ID: UC.5

Actor: Operator

Goal: Display useful information to the operator in a screen overlay.

Actor	System
Switch on overlay	Display, HUD overlays like compass, tracked target info, mini map, etc.

Table 5: GUI overlays

View recording:

ID: UC.6

Actor: Playback user

Goal: Play a full recording from the system, displaying feed from every camera.

Actor	System
Play recording of session and look around in VR headset	Playback full stitched video and display the part of the video the playback user is looking at in the VR goggles

Table 6: View recording

View user recording:

ID: UC.7

Actor: Playback user

Goal: Play back exactly what the operator could see in his field of view.

Actor	System
Play recording of a session	Play exactly what the operator was looking at while recording the video

Table 7: View user recording



4.2 System Requirements

This part of the chapter lists the different requirements for the system. They are split into 3 different categories: Functional requirements, Non-functional requirements and Constraints for the system and the project. Each system requirement has a unique ID consisting of a type code (F, NF and C) and a number to make the ID unique. The requirements have a color-coded state which can be *pending*, *accepted*, *replaced* and *removed*.

Priority	Explanation
A	The requirement must be achieved
B	The requirement should be achieved
C	The requirement should be achieved, but is not a priority

Table 8: Priority explanation

Classification	Explanation
F	Functional requirement
NF	Non-functional requirement
C	Constraint for the system or project

Table 9: Classification explanation

State	Explanation
Pending	The requirement is being discussed and planned
Accepted	The requirement has been accepted
Replaced	The requirement was replaced by one or more other requirements
Removed	The requirement was turned down and has been removed

Table 10: State explanation

	[Date]	[Title]	[State]	[Priority]
[X.X]	[Origin]	[Use case ID] (Functional only)	[Test ID]	
	Description	[Description]		
	Comments			

Table 11: Requirement template description



4.2.1 Functional requirements

Functional requirements are the system requirements that describes the direct functionality of the system. These are the requirements necessary to fulfil to make the system function the way that it was imagined by KDA and is therefore provided by them.

F.1	17.01.2018	HUD	Accepted	A
	KDA	UC.5	T.1	
	Description	The display must show information on a HUD		
	Comments			
F.2	17.01.2018	Playback	Accepted	A
	KDA	UC.6, UC.7	T.2	
	Description	The system must be able to play back recorded video		
	Comments			
F.3	17.01.2018	Recording	Accepted	A
	KDA	UC.2	T.3	
	Description	The system must be able to record at least 5 minutes of video		
	Comments			
F.4	17.01.2018	Streaming	Accepted	A
	KDA	UC.1	T.4	
	Description	The cameras must be able to send live video to the system		
	Comments			
F.5	17.01.2018	Capacity	Accepted	A
	KDA	UC.1	T.5	
	Description	The system must be able to capture and handle video from all four cameras		
	Comments			
F.6	17.01.2018	Stitching	Accepted	A
	KDA	UC.1	T.6	
	Description	The system must be able to merge video from all cameras into one continuous image in VR goggles		
	Comments			



F.7	17.01.2018	Display	Accepted	A
	KDA	UC.1	T.7	
	Description	The system must be able to display the continuous image in VR goggles		
	Comments			
F.8	17.01.2018	Distance calculation	Accepted	B
	KDA	UC.4, UC.5	T.8	
	Description	The system must be able to calculate the range from an operator selected position in the image		
	Comments			
F.9	17.01.2018	Display BMS	Removed	C
	KDA	UC.3	T.9	
	Description	The system should be able to display BMS information overlay from an external data link		
	Comments	Removed due to lack of time and available information		
F.10	31.01.2018	Simulation	Removed	C
	KDA	UC.3	T.29	
	Description	The system shall have a simulator that simulate and send BMS related data to Argos		
	Comments	Removed due to lack of time and available information		

Table 12: Functional requirements



4.2.2 Non-functional requirements

The non-functional requirements are the system requirements that doesn't go at an expense of the functionality of the system. They are properties rather than functionality, meaning that if they are not implemented, the system may still function in the same way, although, not as well. Non-functional requirements improve the functionality of the system rather than adding to it.

NF.1	17.01.2018	Language	Accepted	A
	KDA	T.10		
	Description	Information and messages must appear in English (US)		
	Comments			
NF.2	17.01.2018	VR Platform	Accepted	A
	KDA	T.11, T.1, T.2, T.3, T.7		
	Description	The VR goggles supported shall be the OSVR HDK2		
	Comments	Also supports HTC Vive and other VR headsets.		
NF.3	17.01.2018	Weight	Removed	A
	KDA	T.12		
	Description	The VR goggles must weigh less than 400 grams		
	Comments	Contradicts NF.2. OSVR HDK2 weighs 650grams		
NF.4	17.01.2018	Refresh rate	Accepted	A
	KDA	T.13		
	Description	The VR goggles must have a refresh rate higher than 75Hz		
	Comments			
NF.5	17.01.2018	Resolution	Accepted	A
	KDA	T.14		
	Description	The VR goggles must have at least 960 x 1080 resolution per eye		
	Comments			
NF.6	17.01.2018	Field of view (VR)	Accepted	A
	KDA	T.15		
	Description	The VR goggles must have at least 100 degrees field of view		
	Comments			



NF.7	17.01.2018	Field of view (Camera)	Accepted	A
	KDA	T.16		
	Description	Cameras must have a combined field of view of at least 180 degrees		
	Comments			
NF.8	17.01.2018	Latency	Accepted	A
	KDA	T.17		
	Description	Glass to glass latency must be less than 75ms.		
	Comments			
NF.9	17.01.2018	Network capacity	Accepted	A
	KDA	T.18		
	Description	Network must have a minimum throughput of 512MB/s		
	Comments			
NF.10	17.01.2018	Capture rate	Accepted	A
	KDA	T.19		
	Description	Cameras must be able to capture at least 50 FPS		
	Comments	Partial. Only Gige cameras		
NF.11	17.01.2018	Laptop	Accepted	A
	KDA	T.20, T.1, T.2, T.4, T.7		
	Description	The system shall be able to run on a powerful laptop. The laptop shall be able to run recordings and live video from one camera		
	Comments			
NF.12	17.01.2018	Website	Accepted	A
	KDA	T.21		
	Description	The Argos web site (www.projectargos.net) shall be updated		
	Comments			
NF.13	17.01.2018	Startup time	Accepted	B
	KDA	T.22		
	Description	System startup should take maximum 20 seconds		
	Comments			



NF.14	17.01.2018	Startup effort	Accepted	B
	KDA	T.23		
	Description	Showing live video in VR goggles should take at most 5 clicks		
	Comments			
NF.15	17.01.2018	Learning brief	Accepted	C
	KDA	T.24		
	Description	It should be possible to learn key functionality in 10 minutes		
	Comments			
NF.16	17.01.2018	Learning extended	Accepted	C
	KDA	T.25		
	Description	It should be possible to master all functionality in less than 1 day		
	Comments			
NF.17	17.01.2018	Upgrading	Accepted	C
	KDA	T.26, T.1, T.2, T.4, T.6, T.7, T.17		
	Description	It should be possible to use a 360-degree camera		
	Comments			
NF.18	31.01.2018	Installation	Accepted	A
	KDA	T.30		
	Description	The installation and setup of Windows 10 for the Argos computers shall be documented		
	Comments			
NF.19	31.01.2018	Doxygen	Accepted	B
	KDA	T.31		
	Description	The source code shall support Doxygen documentation		
	Comments			
NF.20	21.02.2018	Casing	Removed	B
	Group	T.32		
	Description	The 360-degree camera shall have a casing for protection from the elements		
	Comments	Removed due the delivery time of casing		



NF.21	02.02.2018	GPS	Accepted	A
	Group	T.33		
	Description	The system needs a new GPS that doesn't drift for tracking the position of the system		
	Comments			

Table 13: Non-Functional requirements



4.2.3 Constraints

The constraints describe the requirements that limits the work process for the developers. They describe the restrictions for platforms, operating systems, version control etc. giving the owner the possibility to specify the possibility for further development and restrict the software to what they have available.

C.1	17.01.2018	Public restriction	Accepted	A
	KDA	T.27		
	Description	The system must not be available to the public		
	Comments			
C.2	17.01.2018	Programming language	Accepted	A
	KDA	T.28		
	Description	Programming language shall be C++		
	Comments			
C.3	17.01.2018	OS	Accepted	A
	KDA	T.27		
	Description	OS shall be Microsoft Windows 10 64-bit version		
	Comments			
C.4	17.01.2018	Version control	Accepted	A
	KDA	T.27		
	Description	Software version control system shall be used. It can be either Mercurial(Hg) or GIT		
	Comments			
C.5	17.01.2018	Map engine	Accepted	A
	KDA	T.28		
	Description	The map engine to be used shall be the Kongsberg product TerraLens		
	Comments	Partial, not TerraLens but OpenStreetMap map engine.		
C.6	17.01.2018	IDE	Accepted	B
	KDA	T.27		
	Description	IDE shall be Microsoft Visual Studio		
	Comments			

Table 14: Constraints



4.2.4 Traceability to tests

Requirement ID	Test ID
F.1	T.1
F.2	T.2
F.3	T.3
F.4	T.4
F.5	T.5
F.6	T.6
F.7	T.7
F.8	T.8
F.9	T.9
F.10	T.29
NF.1	T.10
NF.2	T.11, T.1, T.2, T.3, T.7
NF.3	T.12
NF.4	T.13
NF.5	T.14
NF.6	T.15
NF.7	T.16
NF.8	T.17
NF.9	T.18
NF.10	T.19
NF.11	T.20, T.1, T.2, T.4, T.7
NF.12	T.21
NF.13	T.22
NF.14	T.23
NF.15	T.24
NF.16	T.25
NF.17	T.26, T.1, T.2, T.4, T.6, T.7 T. 17
NF.18	T.30
NF.19	T.31
NF.20	T.32
NF.21	T.33
C.1	T.27
C.2	T.28
C.3	T.27
C.4	T.27
C.5	T.28
C.6	T.27

Table 15: Traceability to tests





5 Test Specifications and Results

Testing is important to make sure our system works as intended. This chapter specifies the tests Argos 2.0 will conduct to ensure compliance with the system requirements that have been received. The chapter also specifies the results of these test, and an explanation for why certain requirements were not met.

Describes

- Test procedure
- How the requirement will be tested
- Acceptance criteria for the tests
- Specific test results from final testing
- Additional information about test results



5.1 Test procedure

Before each test a test log shall be created with information about the test to be performed and the person or persons conducting the test. After conducting the test, the test results will be added to the test log. The test result includes a pass or fail of the Acceptance test, the specific test results and additional comments. The whole group should be made aware of the test conducted and the result.

5.1.1 Test standard

Test ID		Priority	
Requirement tested			
Unit test suite			
Acceptance test			
Integration test			

Table 16: Standard table for listed tests

Test ID	[Test ID]	Date	[Date]
Test Number		Result	
Tester			
Specific Results			
Comment			

Table 17: Standard table for test report



5.2 Test Overview

Many of the requirements Argos 2.0 have been given are continuations from the earlier project groups, and therefore many of the tests are similar, if not identical to tests from the earlier project groups.

5.2.1 Functional requirement tests

Test ID	T.1	Priority	High
Requirement tested	F.1, NF.2, NF.11, NF.17		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Put on VR headset 3. Look at HUD 4. Turn off system 		
Acceptance test	All HUD elements are present and legible and show correct information.		
Integration test	GUI overlays work in Unreal engine		

Table 18: Functional test 1

Test ID	T.1	Date	10/05/2018
Test Number	1	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	<p>Compass is visible, and rotates according to the direction the user is looking in.</p> <p>Navigation Information (groundspeed, course, position) is visible.</p> <p>Marker information is shown correctly when looking at a marker.</p> <p>Menus are visible, and function correctly</p> <p>Map is visible, the map view rotates according to the user's direction.</p>		
Comment			

Table 19: Functional test 1 result



Test ID	T.2	Priority	High
Requirement tested	F.2, NF.2, NF.11, NF.17		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Put on VR headset 3. Start playback 4. Look around in VR playback 5. Look at HUD 6. Turn off system 		
Acceptance test	The user is able to play back a recorded video		
Integration test	Playback plays without stuttering, video feeds are placed correctly in VR world, HUD elements shows recorded information correctly		

Table 20: Functional test 2

Test ID	T.2	Date	10/05/2018
Test Number	2	Result	Fail
Tester	Vebjørn Tunold		
Specific Results	<p>Recorded video from 360° camera is shown correctly.</p> <p>System does not support recording and playback of GigE Vision video.</p> <p>System can play back GPS information, this is not integrated with video playback.</p> <p>HUD element are placed correctly, but information is not played back since this has not been integrated yet.</p>		
Comment	<p>Due to issues with integrating the GigE Vision cameras we did not have the time to implement playback and recording of GigE Vision video. This is explained further in the GigE Vision challenges documentation.</p> <p>We have not found a solution for automatically synchronizing video recordings from the Insta 360 Pro and GPS recordings.</p>		

Table 21: Functional test 2 result



Test ID	T.3	Priority	High
Requirement tested	F.3, NF.2		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Put on VR headset 3. Start recording 4. Stop recording after five minutes 5. Start playback of recording 6. Look around in VR playback 7. Look at HUD 8. Turn off system 		
Acceptance test	The entire 5 minutes of recorded video is stored.		
Integration test	Playback plays without stuttering, video feeds are placed correctly in VR world, HUD elements shows recorded information correctly		

Table 22: Functional test 3

Test ID	T.3	Date	10/05/2018
Test Number	3	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	<p>System can record more than five minutes of video, and playback stored video from the 360 camera.</p> <p>Recording cannot be started from the system but must be done manually on the Insta 360 Pro camera.</p> <p>Recorded video must be stitched into a single video file before it can be played back.</p> <p>System cannot record video from GigE Vision camera</p>		
Comment	We have not integrated control over the Insta 360 Pro camera into the system. Software is provided with the camera to control it using a computer.		

Table 23: Functional test 3 result



Test ID	T.4	Priority	High
Requirement tested	F.4, NF.11, NF.17		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Display single camera feed 3. Turn off system 		
Acceptance test	The system can receive live video from the cameras		
Integration test	Camera feed appear correctly on screen, changes in the real world are perceived to be instantaneously shown on screen		

Table 24: Functional test 4

Test ID	T.4	Date	10/05/2018
Test Number	4	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	System can display video feed from single GigE Vision camera		
Comment	Latency about 130-170ms, perceived as almost instantaneous		

Table 25: Functional test 4 result



Test ID	T.5	Priority	High
Requirement tested	F.5		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Display four camera feeds 3. Turn off system 		
Acceptance test	The system is able to display live video from all four cameras at once		
Integration test	All camera feeds appear correctly on screen, changes in the real world are perceived to be instantaneously shown on screen		

Table 26: Functional test 5

Test ID	T.5	Date	10/05/2018
Test Number	5	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	<p>The system can connect to all four GigE cameras and display video from them.</p> <p>Changes in the real world are perceived to be instantaneously shown on screen</p>		
Comment			

Table 27: Functional test 5 result



Test ID	T.6	Priority	High
Requirement tested	F.6, NF.17		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Display merged camera feeds 3. Turn off system 		
Acceptance test	The system can stitch separate video feeds in real time to one image		
Integration test	The camera feeds appear as one continuous image, changes in the real world are perceived to be instantaneously shown on screen		

Table 28: Functional test 6

Test ID	T.6	Date	10/05/2018
Test Number	6	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	The feed from front, left and right facing cameras are displayed as a continuous image.		
Comment	There is a clear divide between the different camera feeds, this is a problem also present in previous versions of Argos.		

Table 29: Functional test 6 result



Test ID	T.7	Priority	High
Requirement tested	F.7, NF.2, NF.11, NF.17		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Put on VR headset 3. Look around in VR world 4. Turn off system 		
Acceptance test	The system can display stitched video in the VR-goggles		
Integration test	The VR world appears as one continuous image, changes in the real world are perceived to be instantaneously shown on screen		

Table 30: Functional test 7

Test ID	T.7	Date	10/05/2018
Test Number	7	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	<p>Using the Insta 360 Pro the VR world appears as one continuous image entirely surrounding the user. There is a noticeable delay before changes in the real world is shown on screen.</p> <p>Using the GigE Vision cameras the VR world appears as one continuous image, with seams between the different camera feeds. Changes in the real world are perceived to be instantaneously shown on screen.</p>		
Comment	GigE Vision has acceptable latency, Insta 360 Pro has too high latency		

Table 31: Functional test 7 result



Test ID	T.8	Priority	Medium
Requirement tested	F.8		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Put on VR headset 3. Select position in image 4. Look at calculated range 5. Turn off system 		
Acceptance test	The system can calculate the range between two coordinates.		
Integration test	The system can display a distance that matches the actual distance		

Table 32: Functional test 8

Test ID	T.8	Date	10/05/2018
Test Number	8	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	When looking at a marker information about the marker is displayed, including the distance to the marker. At large distances the distance is wrong, due to it calculating a straight-line distance, instead of the distance along the surface of the earth.		
Comment	The reason we chose to calculate the distance in a straight line instead of a more accurate calculation is due to the issues we faced with implementing it. This is explained further in the Integration chapter.		

Table 33: Functional test 8 result



5.2.2 Non-functional requirements tests

Test ID	T.10	Priority	High
Requirement tested	NF.1		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Put on VR headset 3. Look at HUD 4. Turn off system 		
Acceptance test	All information appears in English		
Integration test			

Table 34: Non-functional test 1

Test ID	T.10	Date	10/05/2018
Test Number	11	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	All information is in English		
Comment			

Table 35: Non-functional test 1 result

Test ID	T.11	Priority	High
Requirement tested	NF.2		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on system 2. Put on OSVR headset 3. Run functional requirement test 1,2,3,7 4. Turn off system 		
Acceptance test	The Argos system works with the OSVR headset.		
Integration test	The system fulfils the Acceptance test for tests 1,2,3,7 with OSVR headset		

Table 36: Non-functional test 2

Test ID	T.11	Date	10/05/2018
Test Number	12	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	System works with HTC Vive Pro, and HTC Vive OSVR headset is non-functional, we can connect to positional tracking and rotational tracking, but not the screen.		
Comment	OSVR HDK2 has been replaced by HTC Vive Pro. System is expected to be compatible with all consumer headset using SteamVR		

Table 37: Non-functional test 2 result



Test ID	T.13	Priority	High
Requirement tested	NF.4		
Unit test suite	1. Check refresh rate in OSVR documentation		
Acceptance test	Refresh rate is higher than 75 Hz		
Integration test			

Table 38: Non-functional test 4

Test ID	T.13	Date	10/05/2018
Test Number	13	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	HTC Vive Pro refresh rate is 90Hz OSVR HDK2 refresh rate is 90Hz		
Comment			

Table 39: Non-functional test 4 result

Test ID	T.14	Priority	High
Requirement tested	NF.5		
Unit test suite	1. Check resolution in OSVR documentation		
Acceptance test	Headset resolution is at least 960x1080 per eye		
Integration test			

Table 40: Non-functional test 5

Test ID	T.14	Date	10/05/2018
Test Number	14	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	HTC Vive Pro resolution is 1440x1600 per eye OSVR HDK2 resolution is 1080x1200 per eye		
Comment			

Table 41: Non-functional test 5 result



Test ID	T.15	Priority	High
Requirement tested	NF.6		
Unit test suite	1. Check field of view in OSVR documentation		
Acceptance test	Headset field of view is at least 100 degrees		
Integration test			

Table 42: Non-functional test 6

Test ID	T.15	Date	10/05/2018
Test Number	15	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	HTC Vive Pro field of view is 110 degrees OSVR HDK2 field of view is 110 degrees		
Comment			

Table 43: Non-functional test 6 result

Test ID	T.16	Priority	High
Requirement tested	NF.7		
Unit test suite	<ol style="list-style-type: none"> 1. Check camera field of view in documentation 2. Check angle of camera mounting 3. Combine the three cameras' field of view into a single field of view, and check total field of view 		
Acceptance test	Cameras combined field of view is at least 180 degrees		
Integration test			

Table 44: Non-functional test 7

Test ID	T.16	Date	10/05/2018
Test Number	16	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	GigeVison combined field of view is approximately 240 degrees, plus 200 degrees from rear facing camera. Insta 360 Pro field of view is 360 degrees		
Comment			

Table 45: Non-functional test 7 result



Test ID	T.17	Priority	High
Requirement tested	NF.8, NF.17		
Unit test suite	<ol style="list-style-type: none"> 1. Point Argos camera at screen showing solid color 2. Point high speed camera at both the screen showing solid color, and screen showing Argos system VR world 3. Turn on system 4. Start recording with high speed camera 5. Change color of screen showing solid color 6. Observe the number of frames it takes for the new color to display on Argos system output 7. Turn off system 		
Acceptance test	Calculated latency is less than 75ms, preferably less than 20ms		
Integration test			

Table 46: Non-functional test 8

Test ID	T.17	Date	10/05/2018
Test Number	17	Result	Fail
Tester	Vebjørn Tunold		
Specific Results	Latency measured to 130ms		
Comment	Latency is comparable to previous Argos projects who measured a latency of 125ms. The 5ms difference is within the margin of error for the method used to measure latency		

Table 47: Non-functional test 8 result

Test ID	T.18	Priority	High
Requirement tested	NF.9		
Unit test suite	<ol style="list-style-type: none"> 1. Check network switch documentation to find maximum throughput 		
Acceptance test	Throughput is at least 512MB/s		
Integration test			

Table 48: Non-functional test 9

Test ID	T.18	Date	10/05/2018
Test Number	18	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	Throughput is 52 Gigabits per second, 6.5GB/s		
Comment			

Table 49: Non-functional test 9 result



Test ID	T.19	Priority	High
Requirement tested	NF.10		
Unit test suite	1. Check camera FPS in camera documentation		
Acceptance test	Cameras FPS is at least 50 FPS		
Integration test			

Table 50: Non-functional test 10

Test ID	T.19	Date	10/05/2018
Test Number	19	Result	Partial
Tester	Vebjørn Tunold		
Specific Results	GigE Vision fps is 50, can be increased at lower resolution. Insta 360 Pro fps is 30 when livestreaming, can be up to 120fps when recording		
Comment	360 camera is for proof of concept, so it does not need to meet performance requirements.		

Table 51: Non-functional test 10 result

Test ID	T.20	Priority	High
Requirement tested	NF.11		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on Argos system on laptop 2. Run functional test 1,2,4,7 3. Turn off system 		
Acceptance test	Laptop runs Argos application		
Integration test	Laptop fulfils Acceptance test for functional tests 1,2,4,7		

Table 52: Non-functional test 11

Test ID	T.20	Date	10/05/2018
Test Number	20	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	T.1 Pass T.2 Pass T.4 Pass T.7 Pass		
Comment			

Table 53: Non-functional test 11 result



Test ID	T.21	Priority	High
Requirement tested	NF.12		
Unit test suite	1. Visit Argos web site		
Acceptance test	Argos website contains up to date information about current project, and current project group		
Integration test			

Table 54: Non-functional test 12

Test ID	T.21	Date	10/05/2018
Test Number	21	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	Website is updated with information about the current group and project		
Comment			

Table 55: Non-functional test 12 result

Test ID	T.22	Priority	Medium
Requirement tested	NF.13		
Unit test suite	<ol style="list-style-type: none"> 1. Start timer 2. Turn on system 3. Stop timer when VR world visible 4. Turn off system 		
Acceptance test	Time from system start up to VR world visible is less than 20 seconds		
Integration test			

Table 56: Non-functional test 13

Test ID	T.22	Date	10/05/2018
Test Number	22	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	Time from system startup to VR world is visible 7 seconds.		
Comment			

Table 57: Non-functional test 13 result



Test ID	T.23	Priority	Medium
Requirement tested	NF.14		
Unit test suite	<ol style="list-style-type: none"> 1. Start computer 2. Start all necessary software 3. Stop when entire system is running 4. Turn off system 		
Acceptance test	Number of clicks to get the full application running is 5 clicks or less		
Integration test			

Table 58: Non-functional test 14

Test ID	T.23	Date	10/05/2018
Test Number	23	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	Number of clicks required from the desktop to start the application is: One doubleclick		
Comment			

Table 59: Non-functional test 14 result

Test ID	T.24	Priority	Low
Requirement tested	NF.15		
Unit test suite	<ol style="list-style-type: none"> 1. Give an untrained user access to the system 2. Go through quick brief with user 3. Let user test system 		
Acceptance test	The user can use key features of the system after 10 minutes		
Integration test			

Table 60: Non-functional test 15

Test ID	T.24	Date	10/05/2018
Test Number	24	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	After a quick brief, an untrained user managed to enter view recording, toggle functions and exit the application within 2 minutes.		
Comment			

Table 61: Non-functional test 15 result



Test ID	T.25	Priority	Low
Requirement tested	NF.16		
Unit test suite	<ol style="list-style-type: none"> 1. Give an untrained user access to the system 2. Give the user the instruction manual for the system 		
Acceptance test	In 1 day or less, the user is able to use all functions of the system and understands how the system works		
Integration test			

Table 62: Non-functional test 16

Test ID	T.25	Date	10/05/2018
Test Number	25	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	After reading the manual, the test user was able to use the full functionality of the system, in 5 minutes.		
Comment			

Table 63: Non-functional test 16 result

Test ID	T.26	Priority	Low
Requirement tested	NF.17		
Unit test suite	<ol style="list-style-type: none"> 1. Connect a 360-degree camera to the system 2. Run test T.1, T.2, T.4, T.6, T,7 and T.17 		
Acceptance test	The system runs with a 360-degree camera		
Integration test	The system with the 360-degree camera fulfils T.1, T.2, T.4, T.6, T,7 and T.17		

Table 64: Non-functional test 17

Test ID	T.26	Date	10/05/2018
Test Number	26	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	System works with Insta 360 Pro. Latency is too high to meet requirements.		
Comment	Latency issues are acceptable with the 360 camera since it is proof of concept.		

Table 65: Non-functional test 17 result



Test ID	T.30	Priority	High
Requirement tested	NF.18		
Unit test suite	1. Write documentation for each Windows change while optimizing the system		
Acceptance test	All changes to made in Windows settings are documented		
Integration test			

Table 66: Non-functional test 18

Test ID	T.30	Date	10/05/2018
Test Number	27	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	System requires the Windows Media Framework. System requires SteamVR installed.		
Comment	This information is present in the documentation.		

Table 67: Non-functional test 18 result

Test ID	T.31	Priority	Medium
Requirement tested	NF.19		
Unit test suite	1. Generate Doxygen documentation for the source code		
Acceptance test	All code is described in Doxygen		
Integration test			

Table 68: Non-functional test 19

Test ID	T.31	Date	10/05/2018
Test Number	28	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	Doxygen documentation has been created		
Comment	Only the source code we wrote was documented with Doxygen, for unreal engine source code refer to official documentation		

Table 69: Non-functional test 19 result



Test ID	T.33	Priority	High
Requirement tested	NF.21		
Unit test suite	<ol style="list-style-type: none"> 1. Buy GPS 2. Test accuracy of GPS 3. Let GPS run over a long period of time 4. Implement GPS positioning into system 		
Acceptance test	The GPS is without noticeable inaccuracies in positioning and does not loose accuracy over time		
Integration test	The system is able to receive and use data collected from the GPS		

Table 70: Non-functional test 21

Test ID	T.20	Date	10/05/2018
Test Number	20	Result	Pass
Tester	Vebjørn Tunold		
Specific Results	System can receive and use GPS data in the map and in marking and tracking. GPS has acceptable accuracy, and does not lose accuracy over time.		
Comment			

Table 71: Non-functional test 21 result



5.2.3 Constraints tests

Test ID	T.27	Priority	High
Requirement tested	C.1, C.3, C.4, C.6		
Unit test suite	<ol style="list-style-type: none"> 1. Turn on development computers 2. Check internet connection 3. Check IDE 4. Check version control system 5. Check operating system version 		
Acceptance test	<ul style="list-style-type: none"> - Development computers are not connected to the internet - IDE is Microsoft Visual Studio - All coding documentation is located in a version control system (either Mercurial or Git) - Operating system is Microsoft Windows 10 64-bit version 		
Integration test			

Table 72: Constraints test 1

Test ID	T.28	Priority	High
Requirement tested	C.2, C.5		
Unit test suite	<ol style="list-style-type: none"> 1. Open documentation and software code 2. Check programming language 3. Check documentation 		
Acceptance test	<ul style="list-style-type: none"> - Programming language is C++ - Map engine in the system is Kongsberg TerraLens 		
Integration test			

Table 73: Constraints test 2





6 Risk Analysis

This chapter describes the risks of our project, and our risk mitigation strategies

Describes

- Risk analysis
- Risk mitigation



6.1 Risk management in Scrum

Scrum does not have a formal consensus on risk management [6]. We believe that our agile approach will manage risks throughout the development process. During scrum meetings and sprint planning we will assess the risks in our project and take them into account when giving story points to tasks. Tasks with large impact risks will be started early, to see if the risk develops into an issue. This gives us more time to mitigate the risks, solve the issue, and if necessary to choose a different solution.



6.1.1 Hardware Faults

To be able to reduce the risk of unexpected failures and delays, we looked at all the different hardware that is necessary to work at the project and how it would affect us if something failed. We then made mitigation strategies to either avoid the risk or reduce the impact of a failure.

ID	Risk	Description	Probability	Severity	Risk level
R1	Hardware faults	Technical or Hardware	2,17	2,5	4,8
R1.1	GPS failure	Earlier the Argos project have had a lot of broken GPS modules	4	2	8
			Mitigation: Buy two GPS modules when ordering, figure out why they are starting to drift.		
R1.1.1	GPS affected by electromagnetic noise	We suspect this might be some of the problems from previous Argos project	3	2	6
			Mitigation: Check possible interference from inverter		
R1.2	Malfunctioning work computers	One of the computers we use for development might fail	2	2	4
			Mitigation: Files are backed up on each workstation, additional external backup		
R1.3	Argos pc crashes	The main computer for running the Argos program	1	3	3
			Mitigation: Files are backed up on each workstation, additional external backup		
R1.4	Camera failure	The cameras for the Argos system are very expensive and have long delivery time	1	4	4
			Mitigation: Contact supplier for repairs, test system with fewer cameras and recordings until camera has been replaced		
R1.5	Malfunctioning network gear	We use a switch for LAN to share files between the computers	2	2	4
			Mitigation: Contact supplier for repair or replacement, use backup router.		

Table 74: Hardware faults



6.1.2 Software Faults

On this project we want to test out a few ideas to make the project easier to expand. Programs like Unreal Engine are originally designed to make games. If we could use it for Argos as well this would make expanding the project and adding new functionality easier and faster. The fact that Unreal Engine is not designed for this gives us an increased risk of something failing. We therefore need to identify where things may stop so that we can find a solution or abandon the idea before it affects our ability to complete the project.

R2	Software faults	Software related failures	1,44	2.11	4,44
R2.1	Unreal Engine	Unforeseen problems using Unreal Engine	3	4	12
R2.1.1	Unreal Engine is incompatible	Unreal Engine is a massive state of the art engine, but there might be collisions due to unforeseen reasons.	2	3	6
			Mitigation: Import project into unreal engine and check for software conflicts as soon as possible.		
R2.1.2	Unreal Engine is too slow	Unreal Engine has a lot of features superfluous to our project, might slow it down	3	4	12
			Mitigation: Take a latency test at the earliest possible time, so that if necessary we can choose another solution with minimal impact on our project plan		
R2.1.3	Unreal Engine is discontinued	The support for Unreal Engine is discontinued	1	2	2
			Mitigation: Make the project easily transferable to another engine		
R2.1.4	Unreal Engine insist on using Direct-X	TerraLens is based upon OpenGL, Unreal Engine supports OpenGL, but insists on using Direct-X	2	2	4
			Mitigation: Set aside extra time to find a workaround for this problem		



		for windows applications.			
R2.2	OpenGL	Unforeseen problems using OpenGL	1	1	1

R2.2.1	Using rendering engine based on GLEW32	Argos is currently running GLEW32 and most additional 3'rd party software is based around OpenGL.	1	1	1
			Mitigation: Argos is already using this with the build we received		
R2.2.2	Updating GLEW from 1.3.0 to 2.1.0	GLEW 1.3.0 which is utilized in the current Argos project is outdated and based upon OpenGL 2.0. GLEW 2.1.0 is based upon OpenGL 4.6. OpenGL communicates directly with the graphics card and a newer standard should improve performance.	1	1	1
			Mitigation: Do as soon as possible to find the software conflicts and resolve them		
R2.3	Jira failure	We use Jira to organize our user stories and keep track of backlog and progress.	1	1	1
			Mitigation: All of the information on Jira is also backed up on google drive		

Table 75: Software faults



6.1.3 System Lifetime Risks

The system lifetime risk is not something that could immediately affect our project, but it is a thing we should keep in mind when developing the project to avoid problems later in the project's lifetime.

R3	System lifetime risks	Risk that may occur after our project is finished, that won't affect our progress	1	1	1
R3.1	Vibrations and shaking when driving	If the system would be put into action it would most likely drive on rough roads.	1	1	1
R3.1.1	May damage equipment	A lot of the equipment used for the system is fragile that is not designed for vibrations over time	1	1	1
			Mitigation: Future summer projects or bachelors may consider adding stabilisation or dampening		
R3.1.2	Shaking camera may cause motion sickness		1	1	1
			Mitigation: Future summer projects or bachelor projects may consider adding some sort of stabilisation		

Table 76: System lifetime risks



6.1.4 Human Errors

Human errors are something we can't completely avoid, but if someone gets sick or accidentally destroys something we could make the impact of it less significant.

R4	Human errors	Errors caused by human mistakes	2,14	2,29	4,71
R4.1	Absence	Someone can't be available	2	2	4
			Mitigation: If possible, arrange for them to work from home		
R4.1.1	Absence due to illness	Someone is sick	2	2	4
			Mitigation: If possible, arrange for them to work from home		
R4.1.2	Absence due to sick children	Someone has a sick child	2	2	4
			Mitigation: If possible, arrange for them to work from home		
R4.1.3	Illness with internal supervisor	Radmila is sick	3	3	9
			Mitigation: Plan ahead so that missing a meeting won't affect the progress as much		
R4.1.4	Illness with external supervisor	Alexander is sick	2	3	6
			Mitigation: Plan ahead so that missing a meeting won't affect the progress as much		
R4.2	Lost data	A Group member accidentally overwrites data.	3	1	3
			Mitigation: Keep up to date backups of current and previous versions		
R4.3	Damaging equipment	Accidentally damaging equipment	1	3	3
			Mitigation: Contact supplier for replacement equipment as soon as possible		

Table 77: Human errors



6.1.5 Programmatic Risks

The programmatic risk are risks associated with management of the project. How we keep to the schedule or what we do when unexpected cost appears.

R5	Programmatic risks	Risks associated with management of the project	2	2	4,67
R5.1	Delays	Falling behind on schedule	2	2	4
			Mitigation: Use scrum to stay on schedule, the sprints are also setup with only work on weekdays so that we can catch up in the weekends		
R5.2	Cost	Unexpected cost appears due to things like HW failures in R1	1	1	1
			Mitigation: Contact supplier for replacement of equipment		
R5.3	Delivery Time	Some equipment we order might have a long delivery time	3	3	9
			Mitigation: Order equipment at earliest opportunity		

Table 78: Programmatic risks



6.2 DoD Risk Matrix

To estimate the impact of the risks we used the Department of Defence risk matrix [14].

Likelihood	5	5	10	15	20	25
	4	4	8	12	16	20
	3	3	6	9	12	15
	2	2	4	6	8	10
	1	1	2	3	4	5
		1	2	3	4	5
Consequence						

Table 79: DoD Risk Matrix. Green is low risk, yellow is moderate, and red is high risk.

The total impact of a risk is estimated by the likelihood of occurrence, and the consequence if it occurs.

Level	Likelihood	Probability of occurrence	Level	Consequence
5	Near certainty	80 - 100%	5	Critical impact
4	Highly likely	60 - 80%	4	Severe impact
3	Likely	40 - 60%	3	Moderate impact
2	Low likelihood	20 - 40%	2	Minor impact
1	Not likely	0 - 20%	1	Minimal impact

Table 80: Explanation of the likelihood and consequence levels. Numbers taken from the DoD risk, issue and opportunity management guide [14]

Each risk is placed in the risk matrix, which is colour coded to easily identify the total impact of a risk. Green means low impact, yellow means medium impact, red means high impact. This makes it easy to prioritize risks for mitigation.

Any risk classified as high impact should be monitored and mitigated. Mitigation other risks should be considered according to impact, and cost of mitigation.





7 Architecture

The purpose of this chapter is to give the reader a clear understanding of the system- and software architecture of Project Argos 2.0. After reading this you should know the design of the system- and software architecture. Furthermore, this chapter 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.
- Describes the software architecture
 - Show the justification for allocation of functionality, the architectural pattern and software interfaces.
- Describe functionality of key software components
 - Show in detail how key parts of the software works and interacts with hardware and other software components.



7.1 System context

Project Argos 2.0 is a system which is designed to operate in environments where vision is lacking or dangerous for the operator. Settings such as space exploration, combat, and general information gathering are all environments where Argos 2.0 can be applied. We therefore define Argos 2.0 as a “See-through armored vehicle”. Argos 2.0 enables an operator 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 the surroundings through VR-goggles instead of looking through the windows. Software are applied to the video feed, which provide detailed, visual information about surroundings to the operator.

Figure 9 shows high level system architecture for the Argos 2.0 solution.

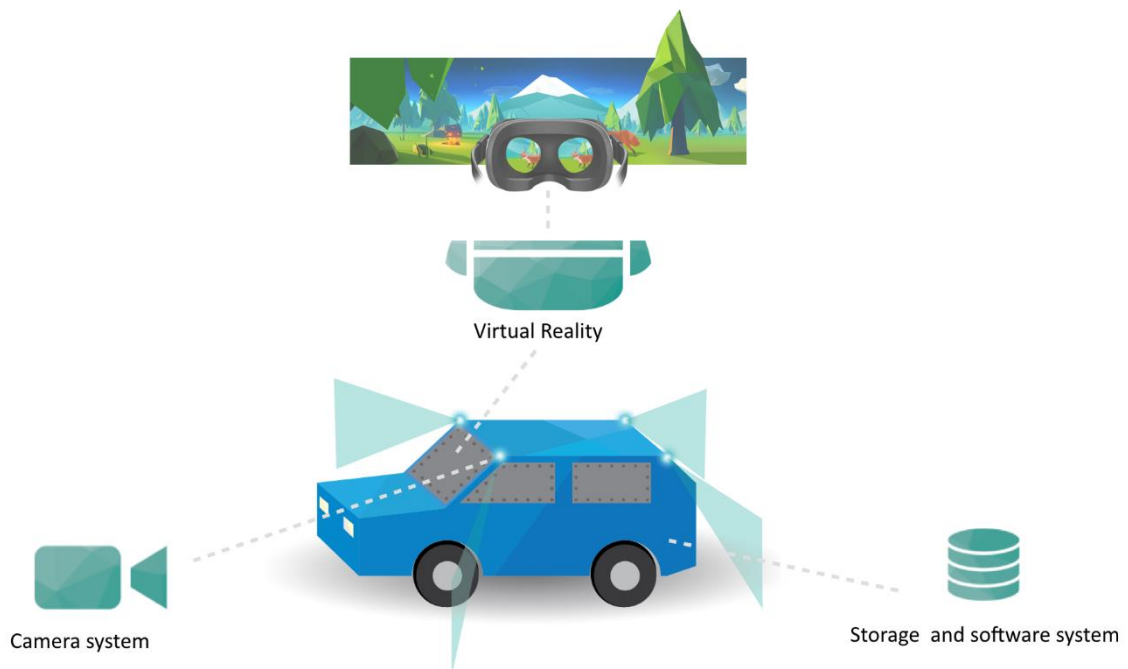


Figure 9: High level System architecture

7.2 System Architecture

Argos 2.0 are designing and developing a software intensive solution, which will manage the functionality of Mixed / Augmented reality. This software is made up by a combination of various devices and software solutions, which together makes up the system architecture of Argos 2.0

Argos 2.0 system architecture diagram describes Argos 2.0 physical components, and the interconnections between them. It is important to note that hardware and software are interwoven and interactions between components may have different origins and be of different nature. We therefore separate our components in this diagram into four types:

7.2.1 Components

- **(Black Rectangle)**
Hardware component available from third-party businesses used in the Argos 2.0 solution
 - **Insta 360 Pro camera**
A round camera which stitches 6 fish-eye lenses together to deliver a seamless 360° view through equirectangular projection.
 - **Elgato Capture 4K 60 Pro**
A Capture card which is used to stream the camera feed from the Insta 360 Pro camera to a texture in Unreal Engine.
 - **DeckLink Studio 4K**
A capture card which is used to capture footage from the application while running. This is used to fulfill UC:5 (View user recording)
 - **MAKO G-223C PoE**
4 high speed cameras which are used to stream real-time footage with minimal latency to the operator. These are included because the Insta 360 Pro camera suffers from high latency.
 - **SSD**
Disk station for saving, writing, and reading footage and files from the system / operator.
 - **GPS**
Hardware to provide data about position and speed to TerraLens / OpenStreetMap.
 - **HD2K VR Goggles**
Open-source Virtual Reality headset.
 - **HTC Vive Pro Goggles**
High end Virtual Reality headset which uses SteamVR to run



- **(Black circle)**
Software available from different third-party libraries used in the Argos 2.0 solution
 - **OpenStreetMap**
Open source software which gives provides geographical data such as maps and roads. OpenStreetMap is currently a substitute for TerraLens.
 - **Unreal Engine 4**
Framework and tools designed to meet ambitious artistic visions. It also delivers a flexible and powerful rendering engine as well as trustworthy performance [15] . It is used in Argos 2.0 because of its viewport designed for Virtual Reality applications, and it's combability with the C++ programming language
- **(Orange circle)**
Proprietary software made from stakeholder (KDA)
 - **TerraLens®**
Geospatial Platform which provides maps and geographical information to the operator.
- **(Green circle)**
Software developed by and for the Argos 2.0 team, to fulfill the systems use cases.
 - **Graphic overlays**
Software responsible for graphical overlays in the Virtual world
 - **Tracking & Marking**
Software responsible for Marking and tracking targets

Interactions between the systems components are defined as “Plugins”, and we separate them into three software types, and one physical type:

- UE4 plugin **(Red Line)**
Unreal Engine develop these plugins, and they are included in the game engine.
- Third Party plugin **(Black Line)**
Open source Plugins developed by third party actors. These plugins do **not** limit the option to distribute / sell the application due to their licensing.
- Argos plugin **(Green Line)**
Plugins developed by and for Argos, to connect components such as the MAKO cameras and Terralens.
- Physical **(Blue Line)**
Physical connection between components. HDMI / USB cables.



Detailed information on integration of each component will be covered in chapter 8 Integration

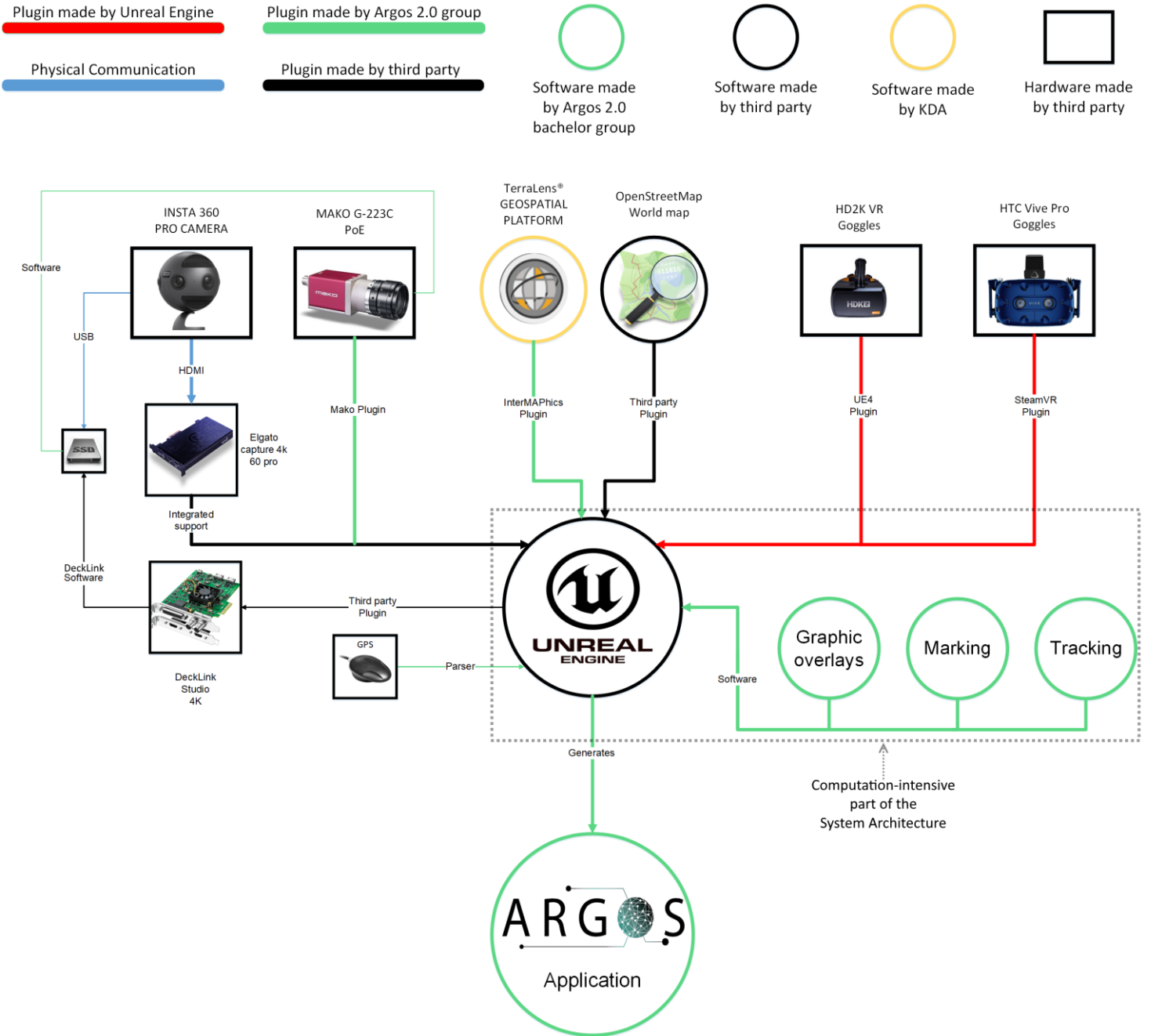


Figure 10: System Architecture Diagram



7.3 Software architecture

Software Architecture for the Argos 2.0 solution has been generated through UML modelling diagrams which include Use case, Sequence and Class diagrams. The desired functionality for the Argos application is discovered using user stories from the SCRUM process model. These user stories are made into Use cases. We get our main building blocks for the architecture through Sequence diagrams where we discover objects and abstractions, and class diagrams where we model the relationship between classes.

It is important to note that our software intensive solution exhibit reusability (it should fit any type of devices and technologies needed to deploy the SA solution), and deployability (we should be able to find technologies which will implement all the functionalities described in use case modes).

7.3.1 Use Case

The purpose of this use case diagram is to secure the discovery of main objects and classes which will contribute towards components of the Software Architecture model. These use cases derive from user stories from the Argos 2.0 project.

Figure 11: *Use Case Diagram* contains two actors with different roles, which interact with the Argos 2.0 software differently.

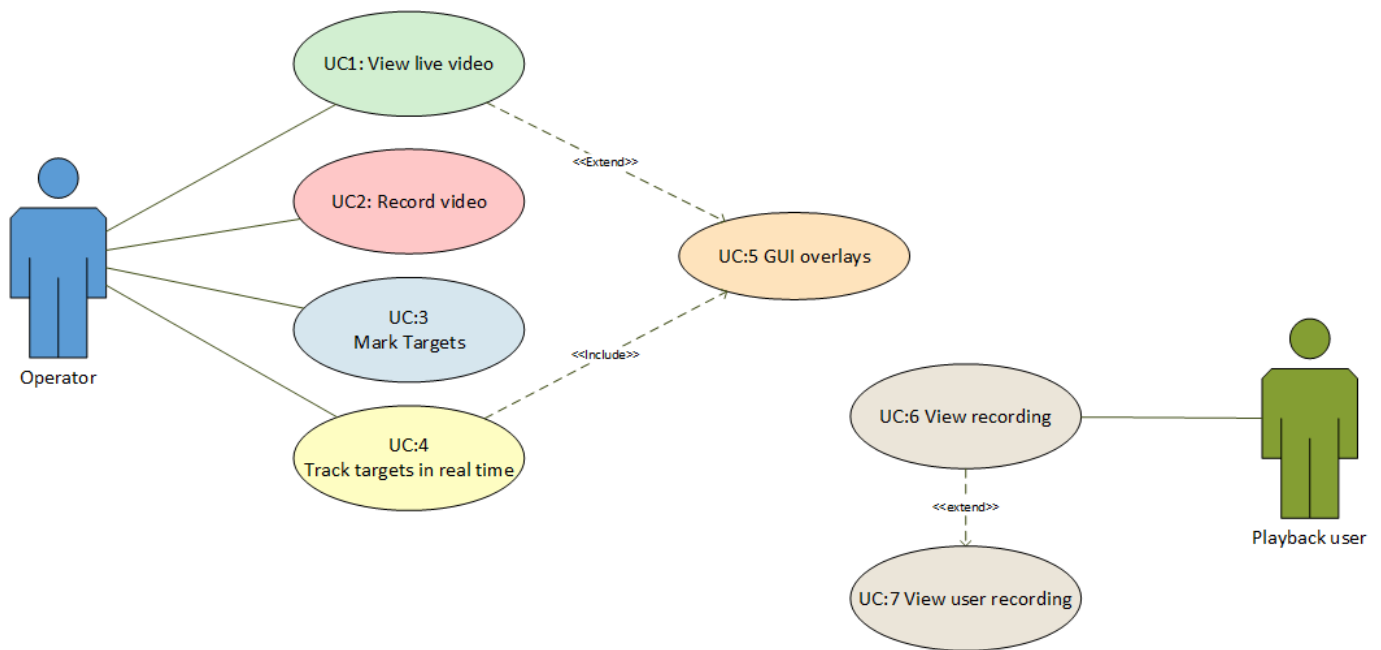


Figure 11: Use Case Diagram

Each use case description contains information on specific component from the System architecture diagram that goes into development of software that derives from these use cases. Information on these components are added in the description to get a better understanding of how Argos 2.0 components interacts with software and can be replaced or removed for reusability if so needed be.

- **Operator**

Actor which interact with the Argos 2.0 application and its features

- **View Live Video**

View the live feed from the camera(s) with *Virtual Reality* goggles.

- **Record Video**

Record footage with the *Insta 360 Pro* or *Mako* cameras

- **Mark Targets**

Visualize positions of the operator and targets in a map, such as *TerraLens* or *OpenStreetMap*

- **Track targets in real-time**

Mark targets contained in a list/buffer and track them using *geographical positioning calculation* and *GPS* data

- **GUI overlays**

Static graphic overlays in the HUD, such as *compass*, *map*, and general information about the vehicle

Dynamic graphic overlays in the HUD such as target location visualization and information about the targets position and type



Operator

- **Playback user**

Actor that can view footage recorded with the Argos 2.0 application by the operator at runtime

- **View recording**

The ability to **dynamically view** recording in **360°**.

- **View user recording**

The ability to view a **static video** from the **operator's** field of view.



Playback user



7.3.2 Sequence Diagrams

All use cases are software based and require detailed planning and execution. In this section we will follow each use case from planning to execution, Sequence diagrams helps identifying objects and model the interaction between them.

Our software follows the Model-view-controller (MVC) design, and we will therefore distinguish between User interfaces (green box), Computational (pink box) and Data objects (blue box) in each sequence diagram. We create a sequence diagram for each use case in Figure 11, which will provide us a collection of objects These objects will be building blocks for our software solutions defined though the SA model.

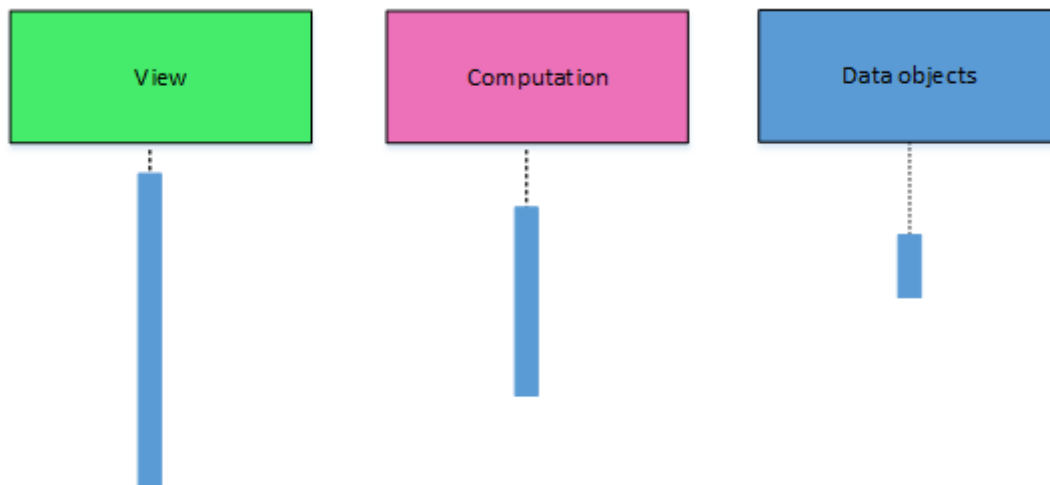


Figure 12: MVC sequence diagram

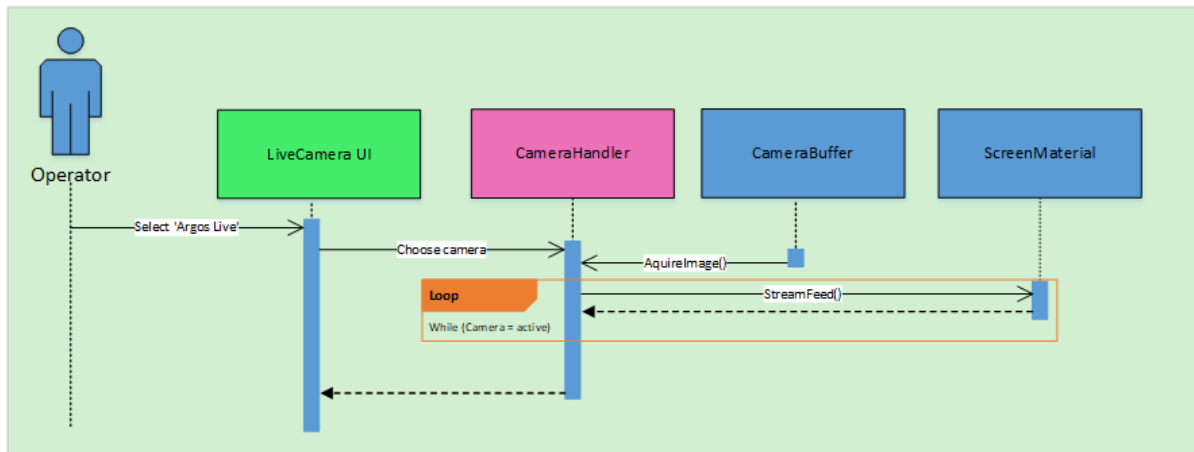
UC1: View Live Video

Figure 13: View Live Video Sequence Diagram

The operator clicks “Argos Live” which camera setup he wants to use. This activates a *CameraStreamHandler* for the desired camera, which fetches the image from the *CameraBuffer* and applies it to the *ScreenMaterial* texture (screen) in the Argos application



UC2: Record Video

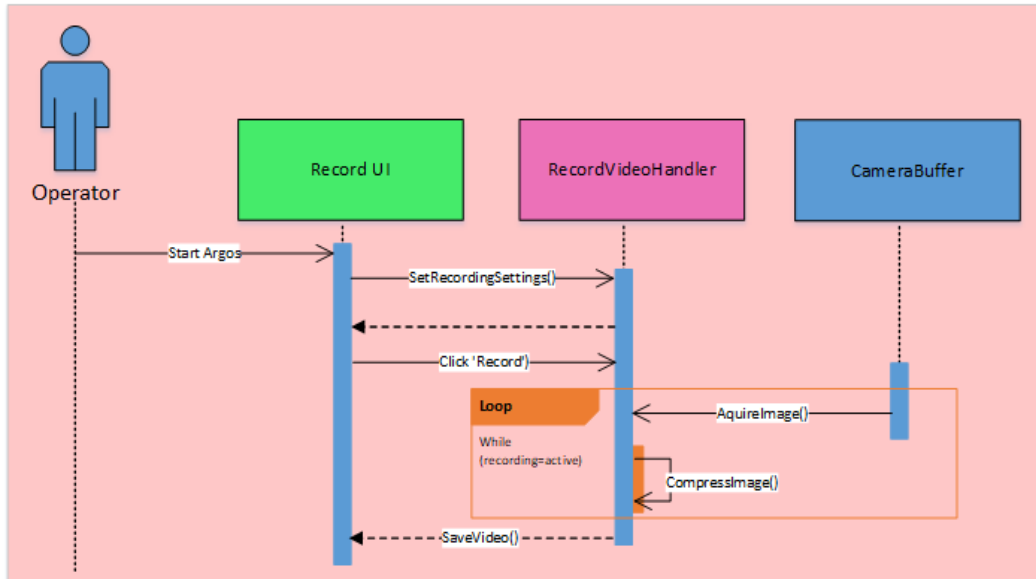


Figure 14: Record Video Sequence Diagram

The operator clicks the “Record” button in the user interface within the application. This will apply the recording settings for which camera the system will record from. At this point we start recording with the set camera. *RecordVideoHandler* fetches images from the *CameraBuffer* and compresses it. When recording is stopped, the file will be saved to the disk, much like how a video camera works. *Argos* deals with two types of cameras, the *Insta 360 Pro* and the *MAKO G-223C PoE*. This sequence diagram can only be applied to the *MAKO* cameras at this stage because the *Insta 360 Pro* camera does not include a *SDK* to let us control the data from the camera. Video Recording for the *Insta 360 Pro* camera will be done manually.

UC3: Mark Targets

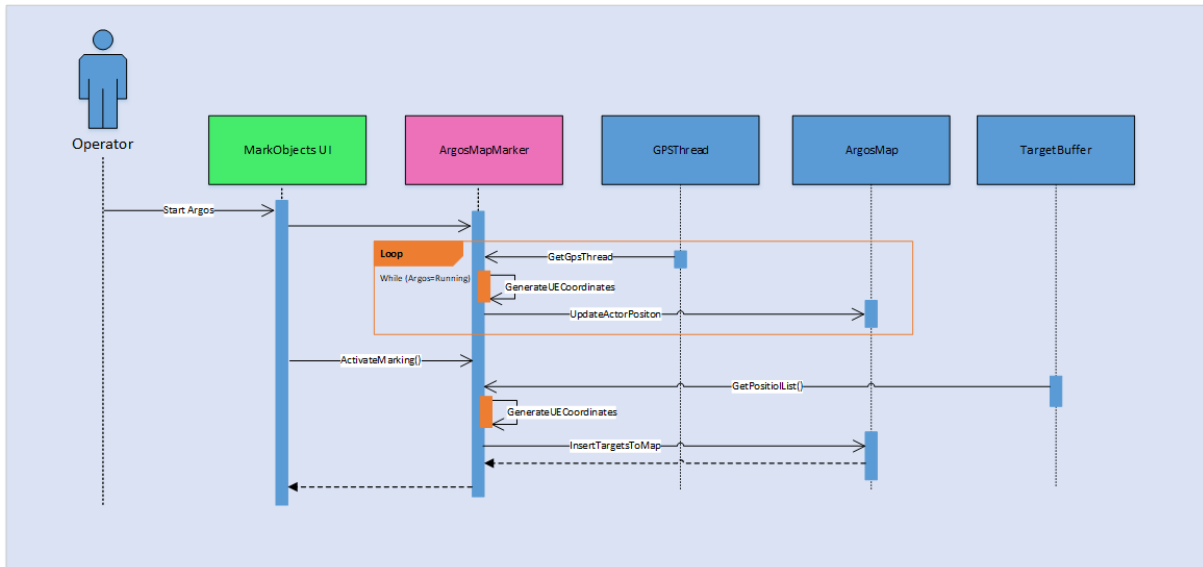


Figure 15: Mark Targets Sequence Diagram

The operator clicks “start” which tells *ArgosMapMarker* to start fetching data from the GPS. This data is then converted into x/y coordinates, which is applied to set the position for the actor in the *ArgosMap*. This sequence loops if the application is running. If the operator chooses to activate marking in the GUI, the *ArgosMapMarker* will start fetching target positions from the *TargetBuffer*, convert them into x/y coordinates and insert them into the *ArgosMap* as the correct position relative to latitude/longitude.



UC4: Track targets in real time

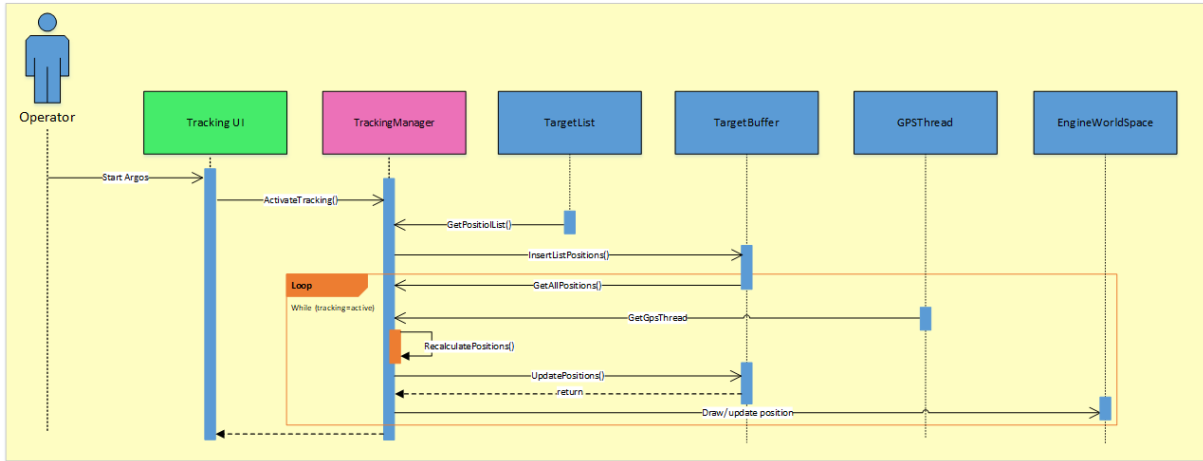


Figure 16: Track targets Sequence Diagram

The operator clicks “Tracking” which activates *TrackManager*. This class fetches data such as positions, target type etc. from a list of targets, which the *ArgosTracker* class inserts into the *Targetbuffer*. It then enters a loop which fetches all targets in the buffer, recalculate their positions and their values. These updated positions are then inserted back into the buffer. The final function in this loop is to insert positions into *EngineWorldSpace*. When the system is shut down, or tracking is deactivated, the loop ends.

UC5: GUI Overlays

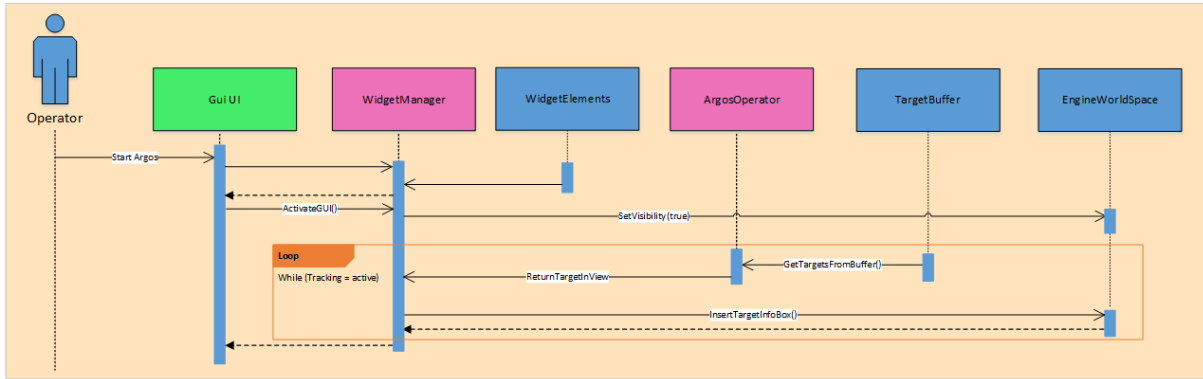


Figure 17: GUI overlays Sequence Diagram

The operator clicks “start” which loads the widgets for the GUI plugin and return the status for the component to true, which indicates that the component is ready to use. The operator then clicks “GUI” in the application user interface which activates *WidgetManager* and *ArgosOperator*. *WidgetManger* fetches the widgets in the application and apply them to the *worldspace*. *ArgosOperator* enters a loop. This loop reads positions from the *Targetbuffer*, then returns the target view the operator is looking in. if the direction the operator is looking in equals the position of a target, the *WidgetManager* will generate a virtual box as an overlay in the virtual reality glasses. When the system is shut down, or GUI is deactivated, the loop ends.

UC6: View recording

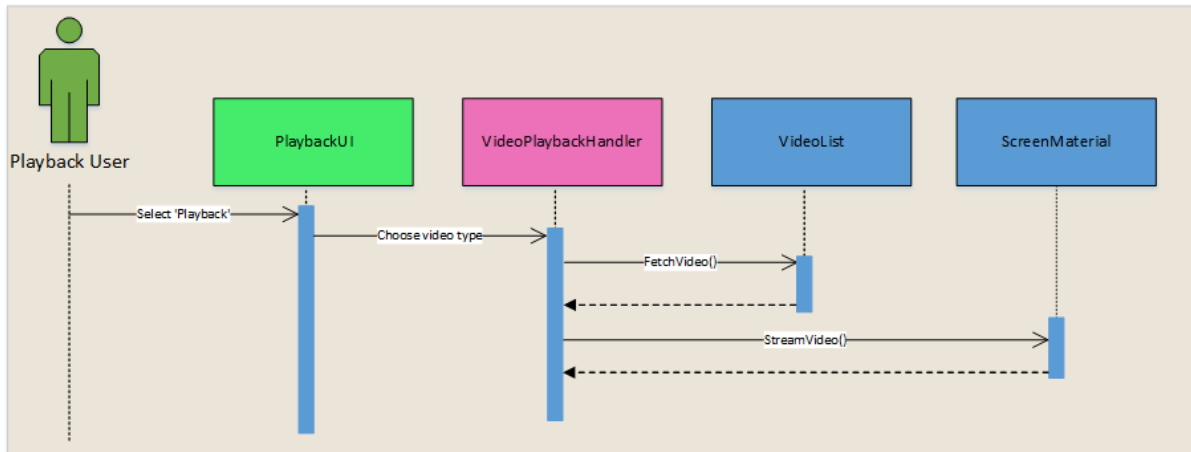


Figure 18: View recording Sequence Diagram

The operator clicks “playback”, which allow him to choose the video type he wants to view (360 video or regular full-HD video). This activates the desired *VideoPlaybackHandler*. The handler fetches a video from the *VideoList* component and applies it to the *ScreenMaterial* (screen).

UC7: View user recording

View User Recording will not have its own sequence diagram because it does not interact with any components besides a media player



7.3.3 Software Architecture

From abstractions discovered in Sequence diagram we find building blocks for our application and the connection between them makes up our Software architecture diagram. This diagram models the computing intensive part of our System architecture diagram. We use the MVC pattern to separate the data from control functions and user interfaces. The *implementation* chapter will describe how software and hardware components are implemented into the Argos 2.0 solution

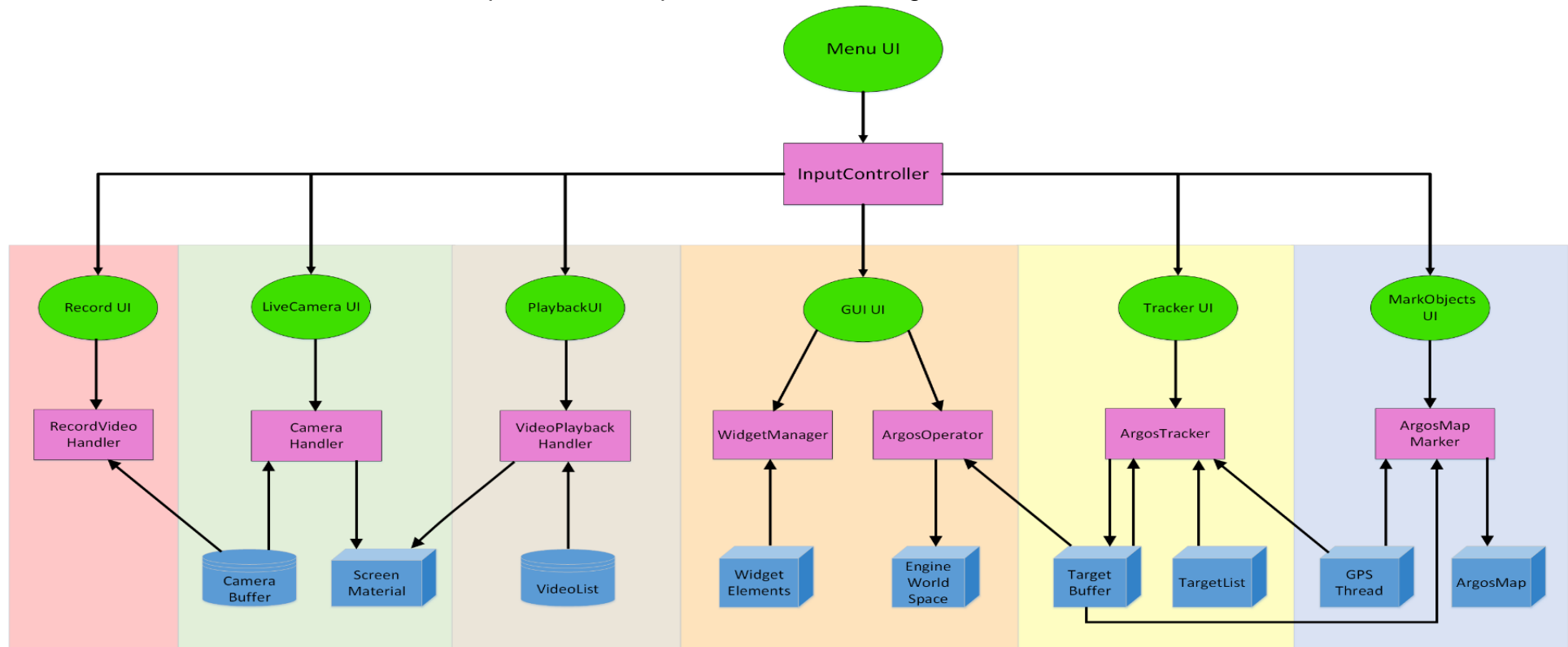


Figure 19: Software Architecture Diagram for computing intensive part of System Architecture (Figure 9)





8 Integration

This chapter describes the Argos 2.0 bachelor projects technical documentation and integrations with the system architecture in mind. The technical documentation includes design choices, how the system is implemented, challenges during development and suggestions for further development.

Our components:

- Unreal Engine
- Cameras
- Tracking of targets
- Marking of targets
- Graphical overlays
- GPS





8.1 Unreal Engine

Unreal engine is a framework originally developed for games, however it has also found use in professional applications.

This chapter describes how Unreal Engine has been used in this project, how Unreal has its own approach to programming and other factors needed for the development.

- 8.1.1 Third party integration with Unreal Engine 99**
- 8.1.2 “Unrealifying” C++ 100**
- 8.1.3 Blueprints..... 101**
- 8.1.4 Level 102**
- 8.1.5 Plugins (third-party + integrated)..... 104**
- 8.1.6 Defining classes 105**
- 8.1.7 Dependencies 106**

8.1.1 Third party integration with Unreal Engine

Unreal engine provides a framework which allow for development of applications in C++. This allows for integration of external C++ software such as TerraLens.

When you create a project in Unreal engine, a file is created which contains the build settings for that project. The file is called <yourproject>.Build.cs. Libraries linked in the project properties section of the IDE and other settings will therefore not be included when you build the application.

To link static libraries the Unreal engine's build system must be used. Each unreal project or plugin contains a "build" file, which is where we link our libraries, header files or other dependencies to the third-party software. This will ensure that the libraries are included in the project when we build the application. Note that you will have to link each library and header file separately in the build file.

Figure 20 displays how we include libraries in the build file for the Argos 2.0 project.

```

- references
public bool LoadInterMAPhics(ReadOnlyTargetRules Target)
{
    // Start library linking here
    bool isLibrarySupported = false;

    string InterMAPhicsGLPath = Path.Combine(ThirdPartyPath, "InterMAPhicsGL");

    // Path to library
    string LibrariesPath = "";

    bool isDebug = Target.Configuration == UnrealTargetConfiguration.Debug && BuildConfiguration.bDebugBuildsActuallyUseDebugCRT;

    if (Target.Platform == UnrealTargetPlatform.Win64)
    {
        LibrariesPath = (Path.Combine(InterMAPhicsGLPath, "Libraries"));
        isLibrarySupported = true;
    }
    else
    {
        string Err = string.Format("{0} dedicated server is made to depend on {1}. We want to avoid this, please correct module dependencies", Target.Platform.ToString());
    }

    if (isLibrarySupported)
    {
        // Add Header files
        PublicIncludePaths.AddRange(new string[] { Path.Combine(InterMAPhicsGLPath, "Includes") });

        // Add Library path
        PublicLibraryPaths.Add(LibrariesPath);

        // Add Static Libraries
        PublicAdditionalLibraries.Add(".....lib");
        PublicAdditionalLibraries.Add(".....lib");
        PublicAdditionalLibraries.Add(".....lib");

        // Add Dynamic Libraries
        PublicDelayLoadDLLs.Add(".....dll");
        PublicDelayLoadDLLs.Add(".....dll");
        PublicDelayLoadDLLs.Add(".....dll");
        PublicDelayLoadDLLs.Add(".....dll");
        PublicDelayLoadDLLs.Add(".....dll");
        PublicDelayLoadDLLs.Add(".....dll");
        PublicDelayLoadDLLs.Add(".....dll");
        PublicDelayLoadDLLs.Add(".....dll");
        PublicDelayLoadDLLs.Add(".....dll");

        //// test your path with:
        // Console.WriteLine("... librariespath -> " + LibrariesPath);
        Definitions.Add(string.Format("WITH_INTERMAPHICSGL_BINDING={0}", isLibrarySupported ? 1 : 0));

        return isLibrarySupported;
    }
}

```

Figure 20: Argos 2.0.build.cs file



8.1.2 “Unrealifying” C++

Unreal engine has its own type system in place of the C++ Standard Template Library (STL). This type system is used similarly to the STL, however some functions are named differently.

Unreal engine also disables some C++ features such as exceptions and runtime type information. Exceptions can be enabled in Unreal engine, but their use is discouraged. We decided to not enable exceptions, and Argos does not make use of them.

Not being able to use runtime type information caused some issues when integrating code from old Argos, especially the GigE Vision code due to the use of `dynamic_cast`. This problem was solved by rewriting the code to not use `dynamic_cast`. Unreal engine also provides its own casting function, `Cast<>`.

We decided to exclusively use Unreal’s type system in our code, so part of integrating code from previous Argos projects was to change the types used. We also followed Unreal’s code standard (<https://docs.unrealengine.com/en-us/Programming/Development/CodingStandard>). In Unreal all type names have a prefix, T for template classes, A for classes inheriting from `AActor`, E for enums etc.

Unreal engine’s editor parses the code for use with its blueprint system. This lets you assign variable values in the editor and in blueprint code. Making variables and functions available to the editor is done using macros such as `UPROPERTY` for variables, and `UFUNCTION` for functions [16]. Specifiers such as `BlueprintReadWrite` are used to control how the engine uses them.



8.1.3 Blueprints

Blueprints is a scripting system in Unreal Engine, which utilises the idea of “Visual Scripting” or programming without writing code. This makes the programming easier and weights more on the design rather than code.

The blueprint system is practical for simple interactions and can be extended by writing blueprint accessible functions in C++. Epic Games, the creators of Unreal encourages the use of both C++ and blueprint. For our project however we wanted everything to be in C++, and only used blueprints for simple prototyping. Blueprints are used however for assigning variables in the editor, which makes it unnecessary to recompile for small configuration changes.

The concept is to connect different nodes in a node-interface to define classes or objects. [17]

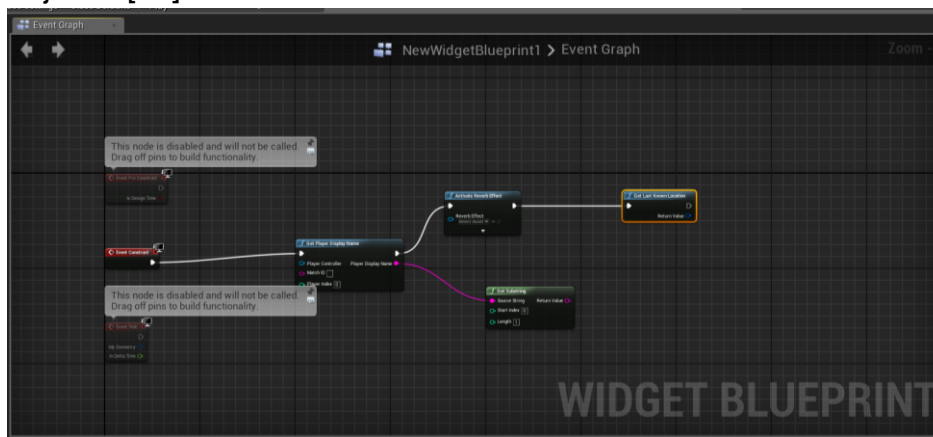


Figure 21: Blueprint Example



8.1.4 Level

In Unreal Engine 4 terms, a Level is made up of a collection of Static Meshes, Volumes, Lights, Blueprints and more all working together to bring the desired experience to the player. [18]

For Argos levels provide a simple system for separating functionality, such as having one level for the GigE Vision cameras and one for the 360° camera.

Levels are used in Argos by placing the appropriate actors for the wanted functionality into the level. For GigE Vision and the 360° camera we have separate levels with their respective screens placed in the level.

When starting Argos you are brought to the menu screen, which allows you to select the mode Argos should run in, GigE Vision livestreaming or playback, or Insta 360 livestream or playback. When changing levels all actors associated with that level is deleted, to keep data on level change we use the game instance to remember settings.





8.1.5 Plugins (third-party + integrated)

Plugins are used in Unreal engine to extend the functionality of the engine and they are easy to add to an Unreal engine project. To add a plugin that is included with Unreal engine it simply needs to be enabled clicking edit then plugins in the menu. Find the desired plugin and activate it.

A third-party plugin can be used as a project plugin, or an engine plugin. Project plugins are added by putting them in the plugins subfolder in the project folder. Engine plugins are added by putting them in the plugins subfolder in the engine source code folder.

For the plugin to be available the project or the engine must be compiled, depending on if the plugin is used as project or engine plugin. It can then be activated the same way as an included plugin.

An engine plugin is available for all projects, while a project plugin is available for that project only.

The plugins used in the project are OSVR and SteamVR for the VR headsets, and the Decklink plugin for the Decklink capture card. We also use an OpenStreetMap plugin to convert OSM files to actors.



8.1.6 Defining classes

Adding new classes to an Unreal engine project should be done using the Unreal editor. When adding a new class using the editor Unreal will generate the required macros to make the editor capable of parsing the code and include it in the editor view and blueprints.

New C++ classes are made by entering the content browser, right clicking and selecting “New C++ Class”. This also gives you the option to choose which Unreal engine class, if any, the new C++ class will inherit from.



8.1.7 Dependencies

The Unreal Engine Media Framework requires the Windows Media Framework to function. This caused us issues in the beginning of the project, since we installed a barebones version of windows without the Windows Media Framework, since we believed it would not be necessary for the project.

For the VR headsets to function they require additional software to be installed. The OSVR headset require the OSVR software [3]. The HTC Vive Pro requires SteamVR. The SteamVR software is installed with Steam, however the development computers cannot be connected to the internet. To install the SteamVR software it is necessary to install Steam on a computer that can be connected the internet, and then copy the SteamVR software to the development computers.





8.2 Insta 360

This document describes the 360° camera, and the capture cards used in the Argos project, and how they were integrated with the rest of the system.

Describes

- The Elgato 4K60 Pro and the DeckLink Blackmagic 4K Studio
- The Insta 360 Pro camera, and why it is used
- Integration of the capture cards with the system
- Why capture cards are needed in the system

8.2.1 The 360° Camera component	108
8.2.2 Capture cards	110
8.2.3 Unreal Integration.....	112
8.2.4 Challenges	114



8.2.1 The 360° Camera component

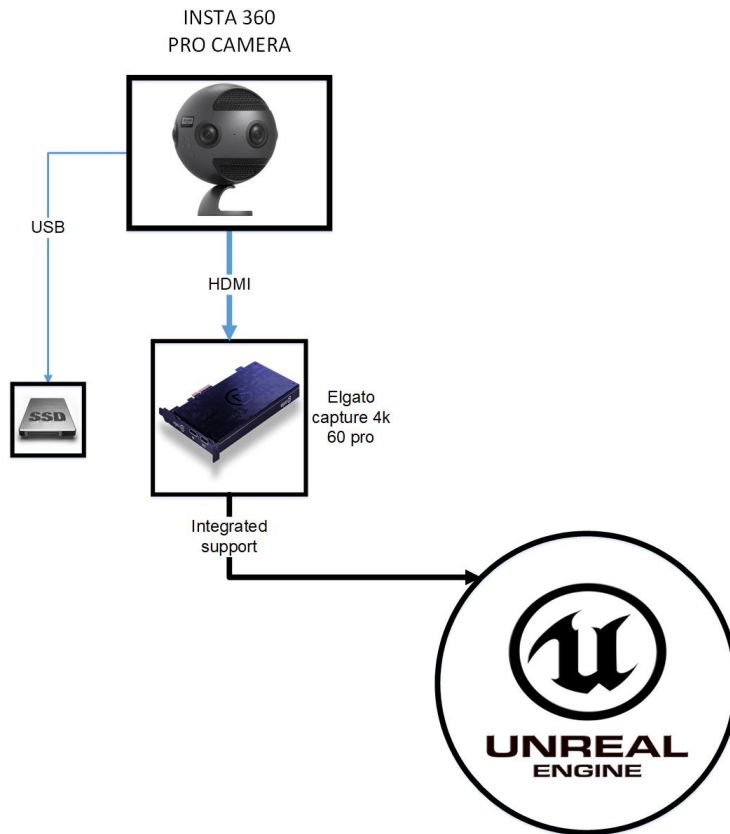


Figure 22: 360° Camera Component

The previous projects used four GigE Vision cameras, these cameras have good resolution, framerate and offer low latency for the Argos system. The problem with these cameras is stitching the separate feeds into a single continuous image. The solution used for this Argos project, and previous Argos projects is to place the video feeds as screens around the user in the virtual world. With careful placement of the cameras this gives a useable result, there is however a clear disconnect between each video feed.

For Argos it was decided to integrate a purpose built 360° camera, the Insta 360° Pro in addition to the GigE Vision cameras. This camera offers real-time stitching, livestreaming 4k footage and the ability to record 8k footage which can be stitched offline.

The stitched footage gives Argos a 360-degree field of view. The footage is seamless and does not suffer from any disconnects between the different camera feeds. The system has less direct control over the 360° camera, and there is no SDK available for the camera, so some features such as starting and stopping recordings from the Argos application have not been implemented. Instead the camera must be started and stopped manually. Programs are supplied with the camera to control it

over Ethernet, Wi-Fi and USB, these are closed source however and cannot be integrated with the system.

When livestreaming the camera can deliver 3840x2160 pixel video at 30 frames per second. The camera offers equirectangular and cube mapped projection.



Figure 23: Cube mapped projection



Figure 24: Equirectangular projection

These projection types are ways to represent the surface of a sphere as a flat surface. Equirectangular projection was chosen since this was supported by both livestreaming and offline stitching. In order to display the video, the video is mapped to a sphere in the virtual world.



8.2.2 Capture cards

The Insta 360 Pro camera has the capability to stitch and livestream 360° video over Wi-Fi, ethernet and HDMI. Streaming over HDMI was chosen since this solution offered the lowest latency.

To receive HDMI signals the computer running the Argos system needs HDMI input. Capture cards offer HDMI input, and can be plugged into consumer computers with a PCIe slot. They are mostly used for streaming video games, but they are also useful for the Argos system.

	Decklink Studio 4k	Elgato 4k60 Pro
Max resolution(HDMI)	3840x2160 [19]	3840x2160 [20]
Framerate at max resolution	30fps [19]	60fps [20]
Unreal compatibility	Plugin required	Fully compatible
Cost	Kr 6195,- [21]	Kr 3990,- [22]

Table 81: Comparison of capture cards

8.2.2.1 The capture cards used

During development two different capture cards were used, the Elgato 4K60 Pro and the Decklink Studio 4K. The Elgato card is aimed at the consumer market, and the Decklink card is aimed at the professional market.

8.2.2.1.1 The Elgato capture card

The Elgato card offers HDMI 2.0 input and output. The output is a passthrough for the input signal, allowing other sources access to the HDMI input [23].

The Elgato card supports several different resolutions and framerates. The 360° camera uses 3840x2160 at 30 frames per second. This is the highest resolution the card supports; however, it supports this resolution at up to 60 frames per second [20]. This makes it possible to upgrade the 360° camera to a new model filming at up to 60 frames per second without changing any other part of the Argos system.

8.2.2.1.2 The Decklink capture card

The Decklink card offers HDMI 1.4b input and output. The output is a passthrough for the input signal. The Decklink card also offers support for several resolutions and framerates. The highest resolution the card support using HDMI is 3840x2160 at 30 frames per second, as this is the maximum supported by the HDMI 1.4b standard



[24]. The card supports higher resolutions using other connections [19], but they are not used in Argos.

8.2.2.1.3 Capture card chosen for the system

The Elgato card was chosen for the Argos system. This card is easier to integrate with Unreal, since third party software is not needed. It also works better with the camera, and receives the full resolution image, while the Decklink card only receives 1280x720 resolution image. This might be because the Elgato card uses HDMI 2.0 while the Decklink card uses HDMI 1.4b.



8.2.3 Unreal Integration

To display video in Argos the Unreal Media Framework is used. This framework is used both for livestreaming 360° video and for playing recordings. The framework can open several media sources such as a web camera, or a video file. The media that is played is displayed on a media texture, this texture is then used in a material. In 3d rendering a material is used to tell the renderer how light interacts with an object. A material defines how an object is shaded with a bump map, or how it reflects light with a specular map. For the purpose of displaying video only the colour of the object is relevant.

The object used to display video is a sphere, this is because the equirectangular projecting used in the 360° videos map to a sphere.

The Decklink card is not recognized by the Unreal Media Framework as a media source. To use it Argos a plugin is needed to make it available to the unreal media framework. The plugin used is developed by The Mill [25], it makes the Decklink card accessible as a media source in Unreal.

The Elgato card is recognized by the Unreal Media Framework without any need for a plugin. This is a benefit since it does not make Argos depend on software developed by third parties.

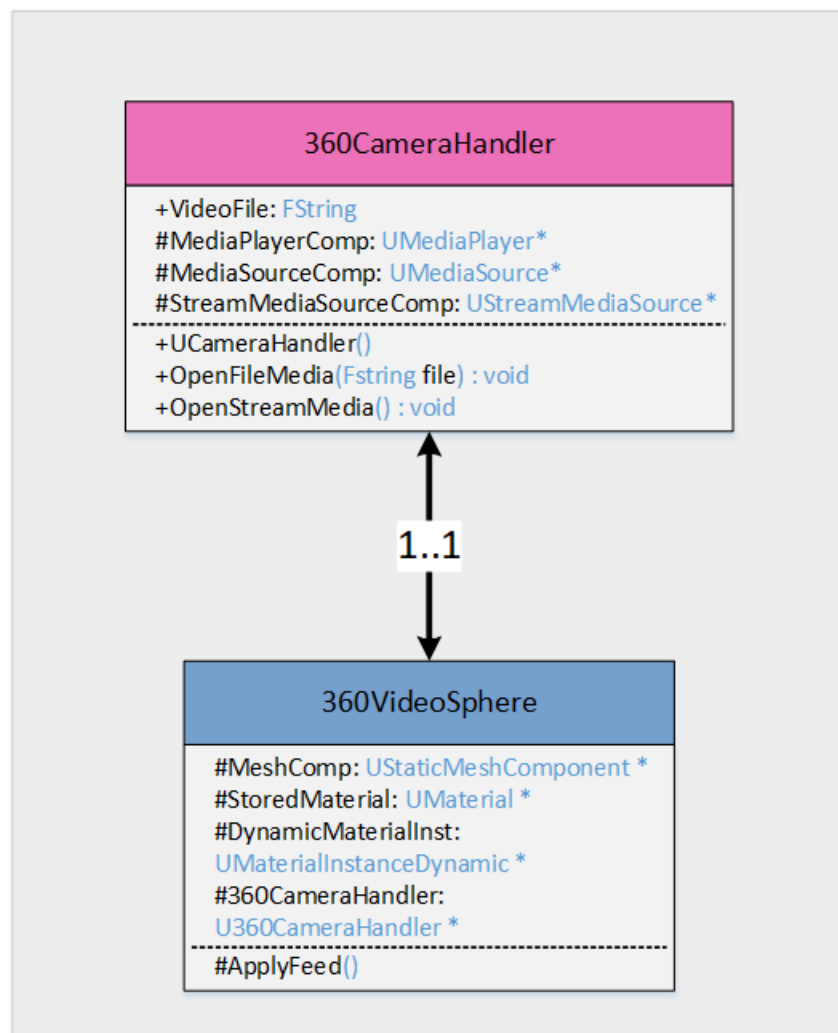


Figure 25: Class diagram for viewing live video through the 360° camera

The 360CameraHandler is responsible for opening media sources, both recordings and livestreams from the 360-degree camera. This media source is applied to a media texture, which is used in the dynamic material instance in the



360VideoSphere. This material is applied to the spherical mesh component belonging to the 360VideoSphere. The user which is placed in the centre of the sphere can then see the video.



8.2.4 Challenges

The project has had issues using the capture cards in our system. The first issue was integrating the Decklink capture card. The Decklink card was not recognized by Unreal Engine as a media source. First, we tried writing our own plugin for the card, however this proved challenging as this was early in the project and we were unfamiliar with unreal engine.

We also found the sample code offered with the Decklink SDK to be difficult to understand and learn from.

This challenge was solved when we discovered that there already existed a plugin for Decklink cards with a license that allowed it to be used in the project [25].

The project also had resolution issues with the Decklink capture card, as it only received a 1280x720 resolution image. Both the Decklink card and the 360° camera should be capable of using 3840x2160 resolution images. The cause of this issue is unknown, and a solution to make the capture card receive a higher resolution from the camera was not found. When plugging a computer into the capture card it received images at the correct resolution. A solution was not found for this issue.

Due to the difficulties in using the Decklink card it was decided to buy a new capture card. The card chosen is an Elgato 4K60 Pro. This card was recognized by Unreal Engine as a media source, without the need for any plugins, and it received images from the 360° camera at the correct resolution without issues.

This card did not need any plugins to work with unreal, and it can receive video at the correct resolution without issues.

The card requires that a HDMI 2.0 cable is used for it to function correctly. If a HDMI cable that does not follow the HDMI 2.0 standard is used the capture card will not receive any signals.

We have also had issues with the latency in the 360° camera. The latency requirement is 75ms, the latency in the 360° video is around 400ms.

During testing the 360° camera was plugged directly to a screen in order to eliminate any potential latency from the rest of the system. The latency when the camera was directly connected to a screen was still 400ms.

We have concluded that the latency originates in the camera, and we are unable to improve this. Due to this the 360° camera will only be used as a proof of concept, and the GigE Vision cameras will be used to meet the requirements for the system.





8.3 GigE Vision

This document describes how the GigE Vision standard has been utilized to enable real time video streaming and recording from Mako G-223C cameras in the Argos 2.0 application.

8.3.1 GigE Vision Component	116
8.3.2 Hardware	117
<i>8.3.2.1 Camera setup.....</i>	<i>117</i>
<i>8.3.2.2 Switch setup</i>	<i>119</i>
<i>8.3.2.3 Network adapter setup.....</i>	<i>120</i>
<i>8.3.2.4 Hardware communication</i>	<i>120</i>
8.3.3 Software Introduction	121
<i>8.3.3.1 Async Multithreading.....</i>	<i>121</i>
<i>8.3.3.2 Software Introduction</i>	<i>121</i>
<i>8.3.3.3 Classes</i>	<i>123</i>
<i>8.3.3.4 Unreal Objects Descriptions</i>	<i>130</i>
8.3.4 Future improvements	131
<i>8.3.4.1 Challenges</i>	<i>131</i>
<i>8.3.4.2 Recommendations</i>	<i>133</i>



8.3.1 GigEVision Component

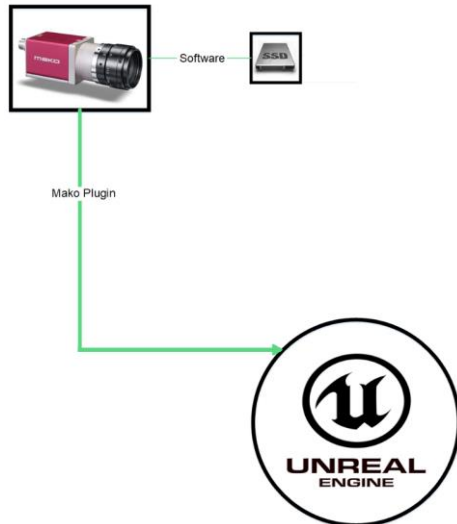


Figure 26: Mako camera system architecture

Mako G-223C PoE are high speed cameras, that utilizes the GigE Vision standard protocols to enable real time streaming and recording in the Argos 2.0 application. The component is implemented into our solution through statically linked libraries supplied by Pleora Visions eBus SDK.

The GigE Vision component was implemented to fulfil the functional requirements F.2, F.3, F.4, F.5, F.6 and F.7 (Table 12). Which derives from the user stories UC.1 (Table 1: View live video), UC.2 (Table 2: Record video) and UC.6 (Table 6: View recording).

Initially the group pushed towards Insta360⁰ integration to fulfil the requirements, but due to the delay present in the Insta360 camera, the inherited camera rig with 4x Mako G-223C cameras was implemented into the Argos 2.0 application. The inclusion of the Mako G-223C cameras means that the NF.8 (Table 13) requirement of glass to glass latency could be met.

This chapter describes how the Argos 2.0 application utilizes the Unreal Engine and the Mako G-223C PoE cameras to view live stream video, view recorded video footage and record the video stream.

8.3.2 Hardware

The setup consists of four Mako G-223C PoE cameras [26] connected to a Cisco SG220-26P switch. The switch utilizes the League of American Communications Professionals(LACP) standard for Link aggregation, sends the camera feed to the Argos computers through the Network Interface Card(NIC), where it is received and interpreted by the Argos 2.0 Application, so that the systems operator can view the camera feed using VR goggles. It is recommended that for this setup to work all components must be able to handle Jumbo packets, and as high a value for retrieving buffer as possible [27]. Especially when dealing with large image formats Jumbo packets must be enabled to ensure that the buffer can handle an entire image and not wait for several packets to arrive.

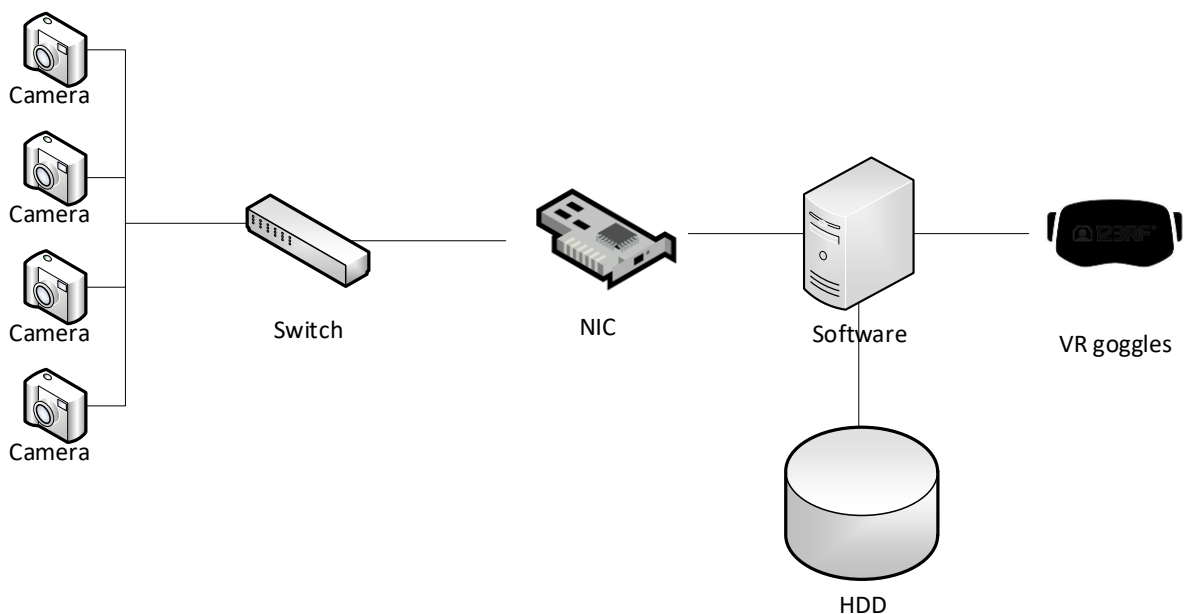


Figure 27: Device architecture

8.3.2.1 Camera setup



Figure 28: Mako camera

- 4x Allied Vision Mako G-223C GigE Vision cameras with power over ethernet (PoE) support
- 1x Fujinon FE185C086HA-1 fisheye lens
- 3x Kowa LM6JC wide angle lenses



8.3.2.1.1 Pixel Format

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 when relying on the standard BayerBG8 pixel format. However, the standard 2D Textures in Unreal engine are unable to read BayerBG8 and only takes RGB formats, so Argos 2.0 had to rely on the format converter supplied with the eBus SDK to get a readable format. This takes some toll on the dataflow.

8.3.2.1.2 Delay

The greatest advantage of the Mako G-223C is that it does not buffer images before sending them. This means that we get instant access to any frame produced by the camera and can display them in our application with minimal delay.

8.3.2.1.3 Power over Ethernet

The Mako G-223C cameras uses PoE which means that it can draw power from the ethernet cable. This is a great advantage as they do not require standalone power.

8.3.2.1.4 GigE Vision Protocols

GigE Vision does not rely on the standard RTSP (Real Time Streaming Protocol) that most conventional IP Cameras do. They use a combination of the GVSP (GigE Vision Streaming Protocol) and GVCP (GigE Vision Control Protocol) which is a standard designed under the umbrella of AIA (Automated Imaging Association) and is a UDP/IP based lightweight, high-speed protocol. [28]

GigE Vision Streaming Protocol

Runs on the UDP protocol. Covers the definition of data types and the ways images can be transferred via GigE.

GVSP provides a protocol for streaming non-compressed and – since version 2.0 – also compressed data streams. Mako G-223C cameras does only support version 1.2 so this is not useful in the Argos 2.0 project however. For transmission an image is split into several packages and the packages are recompiled when received. Like with GVCP, an optional package resend ensures data security. The streaming protocol not only supports transmission of most image formats, it can also add metadata, e.g., trigger information or histograms, to the image data, in addition, every image contains a camera specific time stamp. Version 2.0 and higher versions



optionally support IEEE 1588(standard for precision time synchronization) so that different cameras use a common system time, which is a feature that would have been useful in Argos 2.0. [29]

GigE Vision Control Protocol

Runs on the UDP protocol. The standard defines how to control and configure devices. Specifies stream channels and the mechanisms of sending image and configuration data between cameras and computers.

The core purpose of the GVCP is the control of cameras on a register basis. This means applications read out and write individual data blocks. Every data block (register) represents one or several features (such as the exposure time for the sensor). Whenever a register is accessed, feedback packages are sent to make sure that the camera has received the control package. Using GenICam (which is the standard that Pleora Visions SDK is based upon) cameras can be configured easily and the camera state can be polled. GVCP also defines a back channel through which the camera can send messages(events) to the application. Along this path, changes of state such as start of the image exposure or exposure errors can be communicated. [30]

8.3.2.2 Switch setup

- 1x Cisco SG220-26P

8.3.2.2.1 Switch

A switch serves as a controller, enabling networked devices to talk to each other efficiently. Project Argos 2.0 uses the switch to enable communication between the four Mako G223-C cameras and the Argos computers NIC. It would however not be needed if the provided NIC supported PoE, and it is recommended not to use a switch by GigE Vision camera manufacturers, since it causes additional delay on the data stream. The introduction of a switch into the system, means that Argos 2.0 must rely on Link Aggregation through LACP for the four port Intel I350 NIC to be able to read the data stream properly. [31]

8.3.2.2.2 LACP

LACP is a subcomponent of IEEE 802.3ad and provides additional functionality for link aggregation groups (LAGs). The use of link aggregation feature to aggregate one or more Ethernet interfaces to form a logical point to point link, known as a LAG, virtual link, or bundle. The MAC client can treat this virtual link like a single link. [32]



8.3.2.3 Network adapter setup

- 1x Intel I350 Server Adapter

8.3.2.3.1 Network Interface Card

A network interface card(NIC) is a computer hardware component that connects a computer to a computer network.

The network controller implements the electronic circuitry required to communicate using a specific physical layer and data link layer standard such as Ethernet or Wi-Fi. This provides a base for a full network protocol stack, allowing communication among computers on the same local area network(LAN) and large-scale network communications through routable protocols, such as Internet Protocol(IP) [33]. Unfortunately, the NIC on the Argos computer does not support PoE, which is recommended by GigE Vision camera manufacturers [27]. Leaving no choice but to use a Cisco SG220-26P switch which does support PoE, with LACP for Link Aggregation which puts its toll on the transfer rate from the cameras to the application. The reason is because the switch must interpret the data from the four different cameras into a single signal before pushing it to the NIC. If the extra cost of using a switch was eliminated the transfer rate would improve.

8.3.2.4 Hardware communication

To enable communication with the Mako Cameras, Pleora Visions eBus SDK have been statically linked and thereby implemented in the functionality of Unreal Engine. This means that any developer can have full access to the eBus SDK while utilizing Unreal Engine.



8.3.3 Software Introduction

Realtime streaming from multiple cameras has been achieved by using multithreading, but rather than the traditional way of using worker threads that are instanced and alive in the application, Argos 2.0 uses the asynchronous calls that was introduced in the C++11 STL. The reason for this decision is to maximize the efficiency of data flow and reduce overhead, while not interrupting other important tasks that the application runs. This approach has been used to achieve the desired results for Use Cases; UC1, UC2 and UC6 in chapter 4 and is described by the sequence diagrams View Live Video in Figure 13, Record Video in Figure 14 and View Recorded Video in Figure 18.

8.3.3.1 Async Multithreading

Async is a function that was included in C++11 STL [34] and has since gotten its own adaptation into Unreal Engine [35]. It behaves in such a way that the application can get a promise that a certain condition is filled in a future object and continue without interrupting the data flow. The future object spawns a worker thread on demand so that it can fulfill its purpose and reports back to the main thread once it has achieved its goal. This way of multithreading is a very safe and efficient way of spawning and killing threads without having to worry about reducing performance of the application.

8.3.3.2 Software Introduction

Real time video streaming can be a resource intensive task, especially in a mixed reality environment. While a jittering headset can make for an unpleasant experience for the operator of the system, delay and dropped frames can render the system unusable. It is therefore imperative that the software perform as resource efficient as possible so that all components involved are allocated the correct amount of computation time and remains able to perform their task quickly once they are called.

8.3.3.2.1 Visual Representation



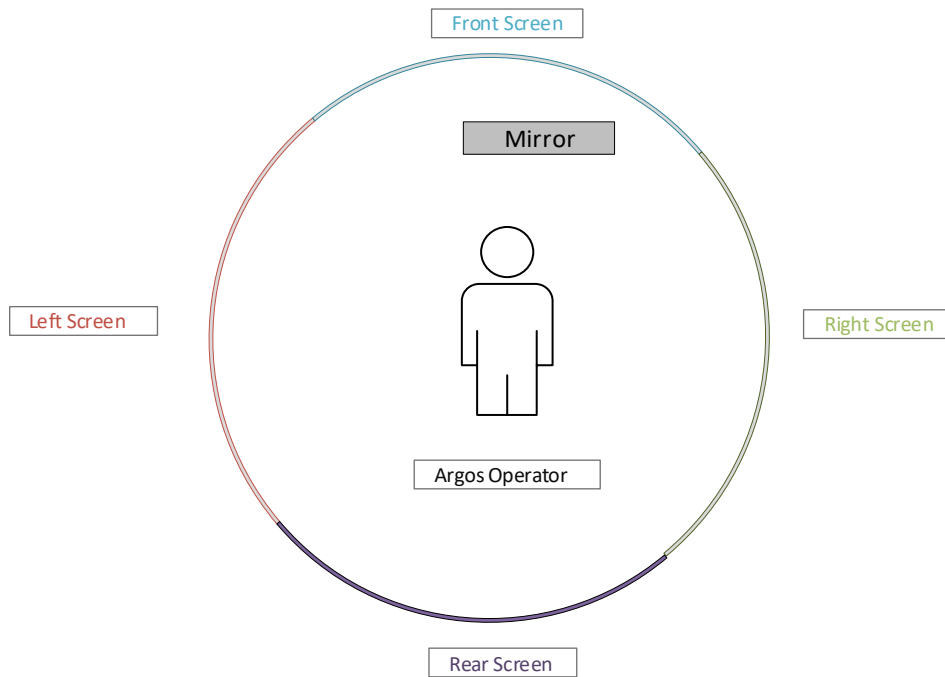


Figure 29: Operators Visual Representation

The visual input to the operator is represented as with three curved screens in the front, a curved screen which is behind the operator, and a mirror which holds the same material as the rear screen.

8.3.3.2.2 View Live Video Datastream Description

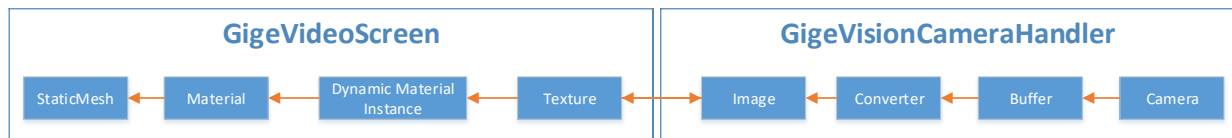


Figure 30: Image data stream flowchart

Flowchart description

The image data stream flowchart describes the flow of image data from each Mako G-223C camera and the path it takes until it shows up on the screen of the operator.

Each *GigeVideoScreen* object has a Static Mesh, Material, Dynamic Material Instance and Texture attached. The Static Mesh is the Screen that the video feed is displayed on. The Material the paint that is applied to the Static Mesh.

The material is manipulated using Dynamic Material Instances which interpret the information stored in its designated Texture. The texture is updated with new data using asynchronous calls to their designated *GigeVisionCameraHandler* with a

promise that a future condition will be reached. This lets the application continue its work uninterrupted while a worker thread is spawned that uses a lambda function call to acquire an image from the camera. The worker thread updates the pointer data using `FUpdateTextureRegions2D` [36], which analyses the texture and only update the sections which does not correspond with the buffered image. If the operator has enabled recording it also releases another thread that handles pushing the current image to a Mp4 file. When the operation is complete, the thread calls back and terminates. Once the texture is successfully updated it is pushed to the dynamic material instance which then becomes visible on its material slot on the static mesh.

8.3.3.2.3 View Recording Video Datastream Description

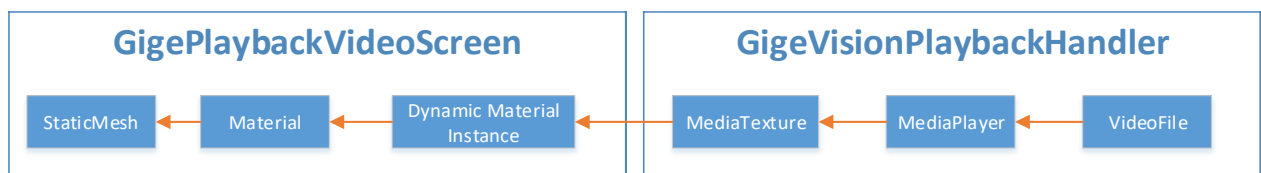


Figure 31: View Recording Video Datastream

Flowchart description

When viewing recorded Mako G-223C footage in Argos 2.0, Unreal Media Framework is used to play the video file. Each `GigePlaybackVideoScreen` has a `GigeVisionPlaybackHandler`. The `GigeVisionPlaybackHandler` opens the assigned video file and starts to play it in the assigned `UMediaPlayer`. The `UMediaPlayer` then transforms the video to a `UMediaTexture` which is picked up by the `GigePlaybackVideoScreens` dynamic material instance so that it can be displayed on the static mesh via the appropriate material slot.

8.3.3.3 Classes

Live stream and Recording class diagram



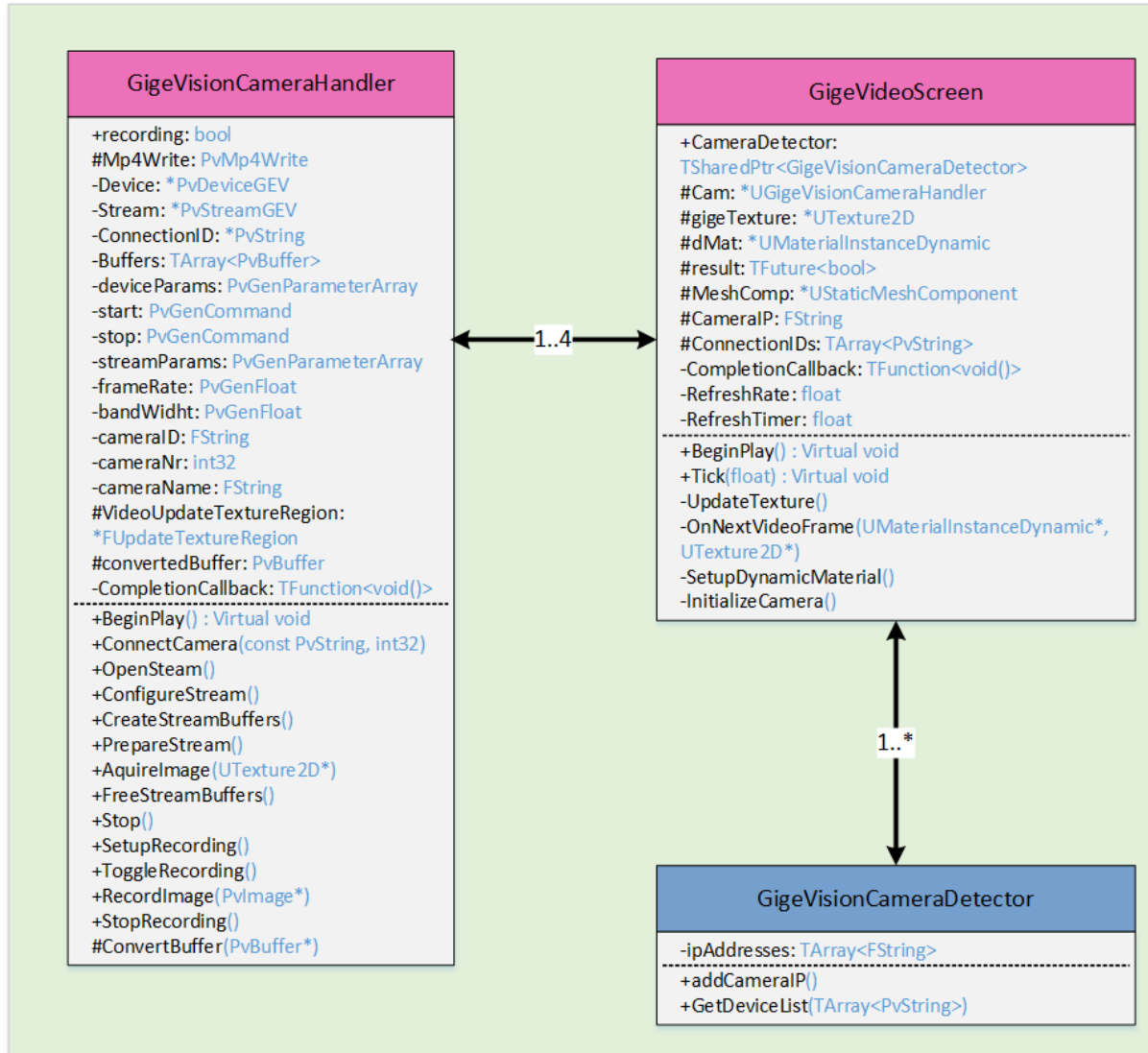


Figure 32: Live Stream and Recording Class Diagram



View recording class diagram

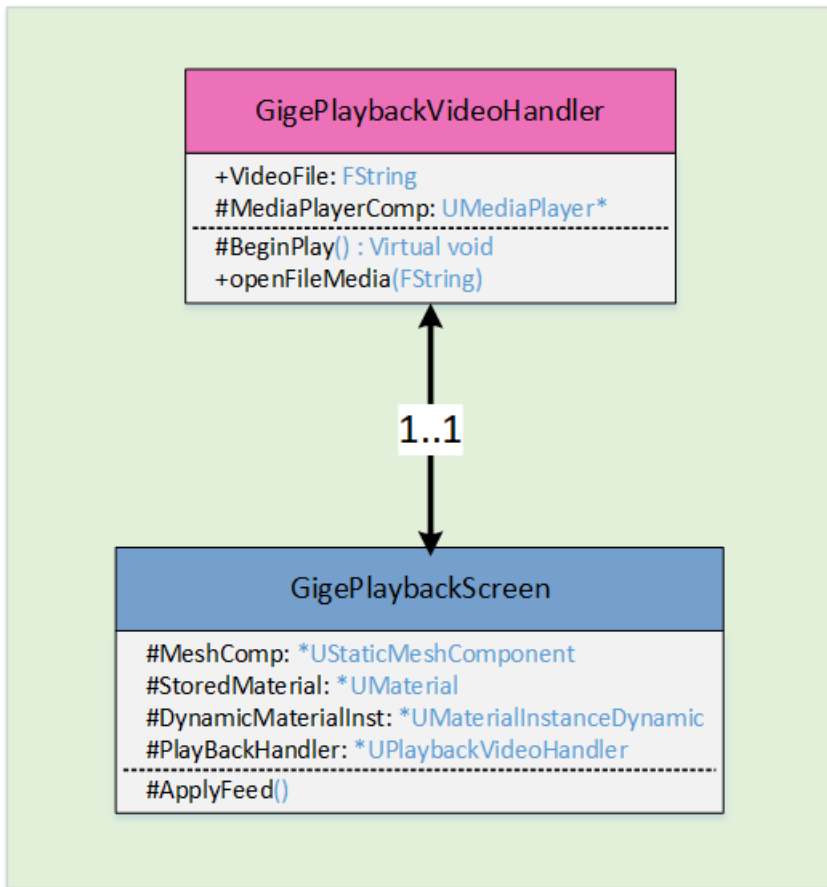


Figure 33: View Recording Class Diagram



8.3.3.3.1 GigeVideoScreen Object

GigeVideoScreen is the object that lives in the World space. It is responsible for displaying the camera feed to the operator.

Startup Phase

When the GigeVideoScreen object becomes alive in the Argos 2.0 application it first checks to see if it can find its designated Mako G-223C camera. The InitializeCamera function checks if the CameraIP corresponds with the IP addresses that resides in the TArray of the GigeVisionCameraDetector. When the check is performed the camera is assigned a GigeVisionCameraHandler, a UTexture2D [37] and UMaterialInstanceDynamic. The SetupDynamicMaterial function then assigns the UMaterialInstanceDynamic to the desired Material slot on the UStaticMeshComponent. Lastly the InitializeCameras function enables the camera by setting in motion the cameras startup routine.

Displaying camera feed

To achieve displaying the camera feed the class has a pointer to the GigeVisionCameraHandler, and a TFuture condition for enabling multi-threading.

It also includes a refresh timer that ensures that the Unreal Engine Tick function is not needlessly updating the texture on every Tick, but rather when the refresh rate calls for it. This is a performance issue since it would be unnecessary to update the texture more regularly than the camera is able to deliver a new image.

At the given refresh rate, the Tick function calls the UpdateTexture function that sets in motion the steps that gets the image from the camera and onto the UStaticMeshComponent.

The TFuture spawn a thread that work asynchronously using Unreal Engines version of the C++11 STL Async library [34], while waiting for its condition to be achieved. The condition is achieved when the lambda function that runs its GigeVisionCameraHandler AcquireImage function updates the UTexture2D object with the new image data. If the operator has chosen to record the session, the same image is also sent for Mp4 compression. Once the condition has returned positive, it is detected and the UTexture2D [38] data is pushed to the dynamic material instance by using the OnNextVideoFrame function. This enables the UStaticMeshComponents to detect the change to the Material and display the freshly arrived image on the surface.

This sequence will continue until the operator choses a different view option or exits the application.



8.3.3.3.2 GigeVisionCameraHandler Object

Each GigeVisionCameraHandler is responsible for a Mako G-223C camera and initializes the data stream with the desired configuration, it also assigns each camera a buffer that receives the raw image data, and a pixel format converter that converts each data buffer from BayerGB8 to a BGRa8(Blue Green Red Alpha 8-bit) image. AquireImage which receives the image buffer data from the data stream, writes the new image data to the texture, and if desired spawn the thread that writes the image to a mp4 file. The function is public and available to be called from the GigeVideoScreen object.

Startup Phase

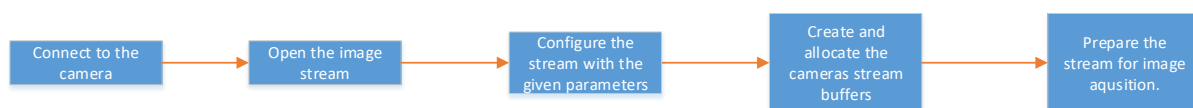


Figure 34: Camera startup routine

The ConnectCamera function connects the GigeVisionCameraHandler to the given Mako G-223C camera, ensuring that it is ready to receive input from the Argos 2.0 application. When the connection is complete the OpenStream function opens the stream from the camera, while the ConfigureStream function ensures that the camera is streaming with the correct parameters. The CreateStreamBuffers function then allocates memory in the Argos computer that is used to store the image data from the camera. Finally, the PrepareStream function ensures that the freshest image from the camera is ready to be sent to the application once the AquireImage function is called.



Acquiring the image

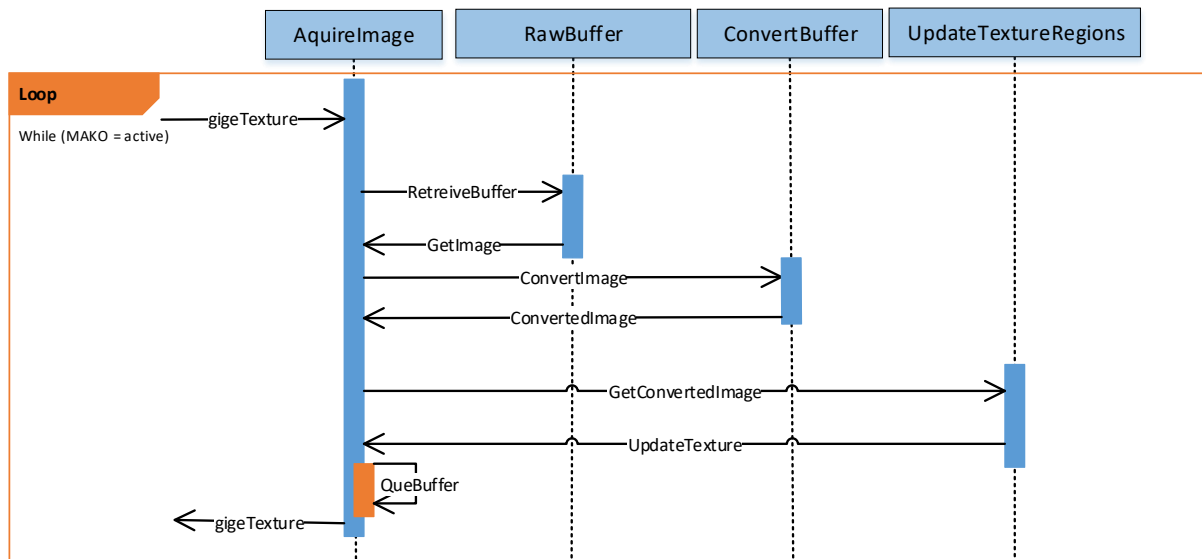


Figure 35: AcquireImage Sequence Diagram

The function AquireImage is the interface between the GigeVisionCameraHandler and the GigeVideoScreen object. When the function is called, a query to the camera is sent to retrieve the image data in the buffer. Once it has been determined that the buffer data is an image, the ConvertBuffer function converts the image pixel data from BayerGB8 to RGBa8. With the converted image available, the FUpdateTextureRegions analyses the converted image data with the data in the UTexture2D pointer and updates the regions of the texture that does not correspond. Then the image data is pushed to a Mp4 file via a spawned thread if the operator has chosen to record the session. The buffer data is then released, and the sequence can begin again.

When the Argos 2.0 operator is done using the system or changes viewing options, the FreeStreamBuffers function flushes the buffer, and the Stop function disconnects the camera.

Recording the images to video files

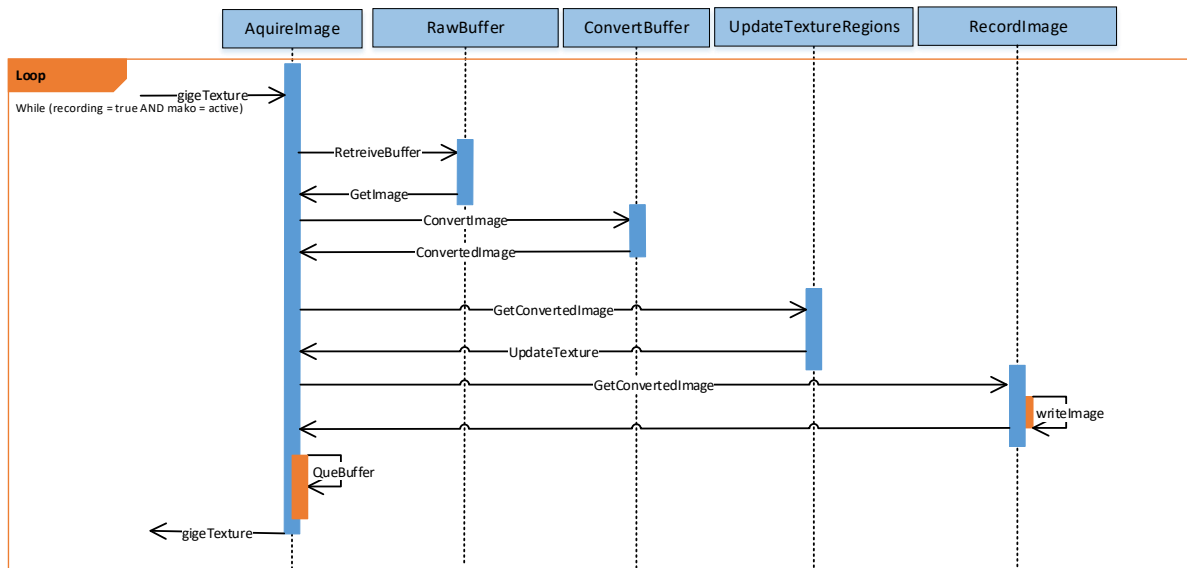


Figure 36: Recording Enabled Sequence Diagram

If the Argos 2.0 operator enables “Recording” via the ArgosUI, the application enables recording of the image stream, with h.264 encoding, to Mp4 files.

When the operator pushes “Recording” in the user interface, the function ToggleRecording, a switch enables or disables the bool “recording”. If “recording” is true, the change is detected in the AquireImage function and the image pointer is sent to the RecordImage function. There it fires off an async thread that writes the image to the Mp4 file. This repeats until the operator toggles recording off or enters a different viewing mode.

The SetupRecording function handles setting up the initial values for the PvMp4Writer and designates the file path to where the file is to be stored.

8.3.3.3 GigeVisionCameraDetector Object

The GigeVisionCameraDetector object handles the discovery of any Mako G-223C cameras connected to the NIC. The public function addCameraIP is a function that any GigeVideoScreen object can use to signal that they are looking for a camera with the given IP address. The GetDeviceList function checks if the IP addresses corresponds with the IP addresses that resides in the TArray of found cameras and returns positive if the device is present.

8.3.3.4 GigePlaybackScreen Object

GigePlaybackScreen objects lives in the Worldspace of the view recording map. They handle displaying previously recorded video footage on the UStaticMeshComponents that acts as the Argos 2.0 operator screens. The



ApplyFeed function creates a UMaterialInstanceDynamic of the Material created by the GigePlaybackVideoHandler object and places it on the UStaticMeshComponent.

8.3.3.3.5 GigePlaybackVideoHandler Object

The GigePlaybackVideoHandler object handles initiating the playback of the assigned VideoFile. The UMediaPlayer opens the video file and writes the feed to a UMediaTexture that is picked up by the GigePlaybackScreen and displayed to the operator.

8.3.3.4 Unreal Objects Descriptions

8.3.3.4.1 Texture

Textures are images that are used in Materials. They are mapped to the surfaces the Material is applied to. Either textures are applied directly – for example, for Base Color Textures – or the values of the texture’s pixels are used within the Material as masks or for other calculations. In some instances, textures may also be used directly, outside of Materials, such as for drawing to the HUD. For the most part, textures are created externally within an image-editing application, such as Photoshop, and then imported into Unreal Editor through the Content Browser. However, some textures are generated within Unreal, such as Render Textures. These generally take some information from the scene and render it to a texture to be used elsewhere.

A Single Material may make use of several textures that are all sampled and applied for different purposes. For instance, a simple Material may have a Base Color Texture, a Specular Texture, and a Normal Map Texture. In addition, there may be a texture for the Emissive and Roughness stored in the Alpha Channels of one or more of these textures. [39]

8.3.3.4.2 Material

A Material is an asset that can be applied to a mesh to control the visual look of the scene. At a high level it is probably easiest to think of a Material as the “paint” that is applied to an object. But even that can be a little misleading, since a Material literally defines the type of surface from which your object appears to be made.

In more technical terms, when light from the scene hits the surface, a Material is used to calculate how that light interacts with that surface. These calculations are done using incoming data that is input to the Material from a variety of images(textures) and math expressions, as well as from various property settings inherent to the Material itself. [40]



8.3.3.4.3 Dynamic Material Instance

In Unreal Engine 4, Material instancing is used to change the appearance of a Material without incurring an expensive recompilation of the Material. Whereas a typical Material cannot be edited or changed without recompiling, an instanced Material can be made to change without such recompilation. Certain types of instanced Materials can even change during runtime in response to runtime events. [41]

8.3.4 Future improvements

8.3.4.1 Challenges

Getting the Mako G-223C cameras to stream a live video feed to the Argos 2.0 application has been a major challenge, not only does it require a fast running software solution, it also involves a lot of hardware components that require the correct parameters to work properly.

8.3.4.1.1 Problem

One Mako G-223C camera has less delay than the others. It is improbable, but not impossible that this issue is caused by the application code, switch or NIC. The reason for this statement is because no matter in which order the code is organized or which port on the switch or NIC the cameras are connected, the problem remains, even if all parameters on all devices are identical. A probable cause might be wear and tear on the cameras themselves, so that the one camera that is performing well, might also be susceptible for this issue soon. The fast camera always has a slightly higher temperature than the others which might be an indicator that it operates on a higher clock speed than the others.

8.3.4.1.2 Hardware

The Mako G-223C camera that does suffer from a slight delay has been tried rectified by several different solutions. Firstly, it was important to make sure that all hardware components including the cameras themselves were configured correctly according to their instruction manuals. Then using the official guidelines from Pleora Vision and other GigE Vision camera manufacturers, to ensure the system inhabited the right parameters for streaming from multiple cameras. This was essential to ensure that the hardware side of the system was able to handle the dataflow from camera and into the application. With Jumbo frames enabled across the system and the correct parameters for receiving buffers, dataflow should be sufficient to stream



from all cameras without delay. However, the delay persisted, concluding that the problem should reside somewhere in the Argos 2.0 software.

8.3.4.1.3 Threads

The cameras were initially running on traditional threads. Not the C++ STL threads, since STL and Unreal Engine do not work well together, but rather the Unreal Engine variant of using FRunnable threads. FRunnable threads work in a similar manner on the technical side but has a vastly different implementation from a software coding aspect, making the process of getting them up and running correctly in the first-place time consuming. For small operations FRunnable is great, but it quickly becomes complicated since the syntax used is different from the STL version of threads. Since FRunnable was the unknown factor in the Argos 2.0 application, and computation time was a probable culprit for the delay, a different approach was contemplated. This led to switching standard multi-threading with the newer and more modern asynchronization model, which was introduced in the C+11 STL standard. STL as previously stated is not compatible with Unreal Engine, but they have introduced their own standard for asynchronization, and the implementation is a lot like the STL one. The delay persisted, even though the Argos 2.0 software now made a much smaller dent on computation time.

8.3.4.1.4 Image to texture

The next probable culprit was the time it took between updating the texture with the image provided from the buffer. Initially it was performed by using UTexture2Ds function UpdateResource. It works in such a way that it bulk uploads the new image data each time it is called. This essentially means that it deletes and recreates the texture every time the function is called. Looking for a more efficient method of achieving the same result, FUpdateTextureRegions was discovered. FUpdateTextureRegions does not bulk upload an entire image, but rather analyses at the data on the texture and updates the regions that do not correspond with the data represented to it. This function looks great on the official Unreal Engine Documentation webpage, but it is unfortunately implemented differently inside the engine itself. Once again making it necessary to analyze the function from deep inside the engine for it to work properly. EPIC GAMES have been notified about this issue and it will probably get fixed. The texture now updates as quick as possible using the RGBa8 pixel format, but the delay is still present.

8.3.4.1.5 Image conversion

At this point the RGBa8 pixel format was suspected to be the reason for the delay, since the application had to convert the raw image to the RGBa8 format at runtime. So, an effort was made to write a function that enabled the textures to display BayerGB8 formatted images as is. This was partially successful as the image



became visible on the assigned material with only slightly distorted colors, and by doing this the need for converting the image was eliminated. The delay still persisted and due to the slightly distorted colors on the image, this method for displaying the video feed was retracted and abandoned.

8.3.4.1.6 Solution

Lastly the camera with the least amount of delay was repositioned as the forward-facing camera, to ensure that the operator would have the best camera in the angle that the car is facing.

8.3.4.2 Recommendations

8.3.4.2.1 Implement Unreal Media Framework

Using a BGRa8 format is the slowest and most taxing conversion of an image for both the eBus SDK and the Unreal Engine, unfortunately it is the only viable option for displaying a video feed in our application since Unreal Media Framework [38] is still at its infancy and very much still under construction. The documentation available on the official Unreal documentation web page cannot be trusted, since it differs a lot from how the actual coded implementation works. However, EPIC GAMES, are working hard at updating both the code and documentation at present moment, and once it is at a usable state, performance could be improved dramatically by choosing to use a YUV422 formatted Media Texture and YUV422 on the eBus converter.

8.3.4.2.2 Update Cameras

The Mako G-223C are beginning to age, and does not support the newest GVSP 2.0 protocol, which offers great improvements over the currently used GVSP 1.2 protocol. Boasting much better handling of a multiple camera setup and is even able to convert the feed on the fly to h.264 mp4 format. This will be a huge relief for whoever inherits this project as Unreal Engine offers excellent support for Mp4.

8.3.4.2.3 Consider Different Protocol

Unreal engine offers support both the NDI and the RTSP standards. If the cameras in Argos 2.0 was switched to any camera supporting these protocols, implementation will be a walk in the park as the Unreal Engine offers support for these standards out-of-the-box, with little to no effort. It can even be achieved using the Unreal Engine editor and the Unreal Engine Media framework without the need to produce any of the software manually.



8.3.4.2.4 Improve Image Recording Implementation

At present moment, the Argos 2.0 application is only able to handle one set of recorded footage files from the Mako G-223C cameras and writes over the same files each time the operator presses record. Fixing this issue by dynamically creating a folder with date time insignia each time the record button is pressed would be preferable. Any future developers could then use a similar method of parsing through the different recordings that has been implemented with the Insta360 camera solution, to allow the operator to decide which set of recordings are to be presented.





8.4 VR-Headgear

This chapter describes why the Argos system uses virtual reality head mounted displays instead of more conventional screen. It describes the different head mounted displays used in the system and the challenges that were encountered when integrating them.

8.4.1 Why our system uses VR headgear	136
8.4.2 What VR headgear our system uses.....	136
8.4.3 Integration of VR Headgear	137
8.4.4 Challenges Integrating VR Headgear.....	138



8.4.1 Why our system uses VR headgear

The purpose of the Argos system is to increase the situational awareness of the user. To this end, the way information is displayed to the user is important. There are two options for displaying information to the user, using one or more screens in the vehicle, or using a VR headset.

Using several screens mounted in the vehicle can display a great deal of information, but the screen borders will lead to a disconnect between each image. Using several screens will also take up a lot of room in an already cramped interior and is therefore not an optimal solution.

Using a VR headset can give the user the experience of viewing from outside the vehicle, giving them a far better overview, while still being protected inside the vehicle. The VR headset will also not suffer from any borders between screens and will take up less space than several screens would.

8.4.2 What VR headgear our system uses

The previous projects have used the Oculus DK2, and before the start of this project an OSVR HDK2 VR headset was bought for the project. Due to issues with the OSVR HDK2 we bought a new HTC Vive Pro. This headset offers better specifications than the OSVR headset.




	Oculus DK2	OSVR HDK2	HTC VIVE PRO
			
Resolution	960x1080 per eye [42]	1080x1200 per eye [43]	1440x1600 per eye [44]
Field of view	100 [42]	110 [3]	110 [44]
Refresh rate	75Hz [42]	90Hz [43]	90Hz [44]

Table 82: Comparison VR headgear



8.4.3 Integration of VR Headgear

To integrate the VR headsets in Unreal plugins are needed to extend the functionality of the engine. The Unreal Engine comes with over 100 built-in plugins, most of these are made by Epic Games, others are made by third parties.

The plugins used in Argos are the OSVR plugin and the SteamVR plugin.

The OSVR plugin makes the project compatible with the OSVR HDK2, and other VR headsets using the OSVR framework.

The SteamVR plugin makes the project compatible with the HTC Vive Pro, and other headsets using the SteamVR framework.

In addition to enabling the plugins in the project the two VR frameworks also need separate software installed for the headsets to function. The OSVR software is available at the OSVR website [3].

The SteamVR software is installed with Steam, which is Valve's online games store. The computers used in the project cannot be connected to the internet, for information security. The solution to this is to install steam on a computer connected to the internet, and then copy the SteamVR software to the offline computer.

When the plugins and required software is installed on a computer Argos can be used with VR headgear. However, for an optimal experience the Argos had to be made with VR in mind.

In traditional 3D application the user interface is drawn on a canvas overlaid the rendering of the 3D world. This solution is not suitable for VR applications, because the user interface will appear to be right in front of the user, and might be only visible to one eye, or split between both eyes depending on location.

The solution used in Argos is to have surfaces placed in the 3D world, onto which the user interface elements are rendered. This is a common solution in VR applications, since it allows the user interface to be placed at a distance the eyes can comfortably focus on.

Since the VR headgear covers the entire field of view of the user using keyboard buttons for settings, such as playback or recording become impractical, since the user would be required to remove the headgear to see the keyboard buttons.

The solution used in Argos is to have menus in the application, the user looks at a button to select it and clicks on it with the right shift button. Having a single button that is easy to find on the keyboard eliminates the need to remove the headgear when selecting options.



8.4.4 Challenges Integrating VR Headgear

The main challenge concerning the integration of VR into Argos was the OSVR headset. During development the OSVR headset proved to be unreliable, requiring it to be plugged in and out several times before it would work. Even when working it would often stop working after running the Argos application a single time. Eventually the OSVR headset stopped working completely.

The Argos system can run using a normal screen instead of a VR headset but testing how the system behaves in VR became impossible. This was especially problematic for the development of the user interface. User interface elements in a VR application must be placed differently than in a normal application, since the users field of view is different. Placing UI elements at the edges of the screen, as in traditional 3D games is not practical in VR. This is because it is hard to focus at the edge of the screen in VR, and the lens distortion makes the edges too blurry for the fine detail required for the user interface.

A solution was to borrow an HTC Vive to develop with. Integrating the HTC Vive and SteamVR into Argos was problem free and allowed for the further development of Argos.

During any future development of Argos, a replacement for the malfunctioning OSVR HDK2 had to be found. Since the HTC Vive performed well the decision was made to acquire the HTC Vive Pro headset, which offers several improvements over the HTC Vive and OSVR HDK2, notably the resolution at a 70% higher pixel count [44].





8.5 Tracking

This document describes the technical solution and implementation of tracking objects in the virtual environment in the Argos 2.0 application. It takes on the challenges of implementation and how we worked around them, as well as how it works with the rest of the system, why it's necessary and how it can be further improved.

Describes

- Implementation of tracking in the virtual environment
- Why it's needed
- Challenges faced when implementing it and how we worked around them
- Future improvements and suggestions

8.5.1 The tracking component	140
8.5.2 Implementation	143
8.5.4 Future ideas	152



8.5.1 The tracking component

Tracking is a software component that acts as an aiding tool for the user of the system to allow them to be more aware of their environment. They calculate relative coordinates from geodetic coordinates of the system and the targets and displays overlays to inform the user about direction, distance, and what the target looks like.

The tracking component is made from the user story “Tracking” (see chapter 4.1.1 on Requirements). Tracking is implemented into use case “U.4” in the Argos 2.0 use case diagram (see Figure 11). Tracking also fulfils requirement F.8 “Calculate Distance” (Table 12). The tracking sequence diagram (Figure 16: Track targets Sequence Diagram) is derived from U.4 and shows the flow of data in the Tracker.

A Glossary for terms and abbreviation can be found at the Appendix section of the documentation.

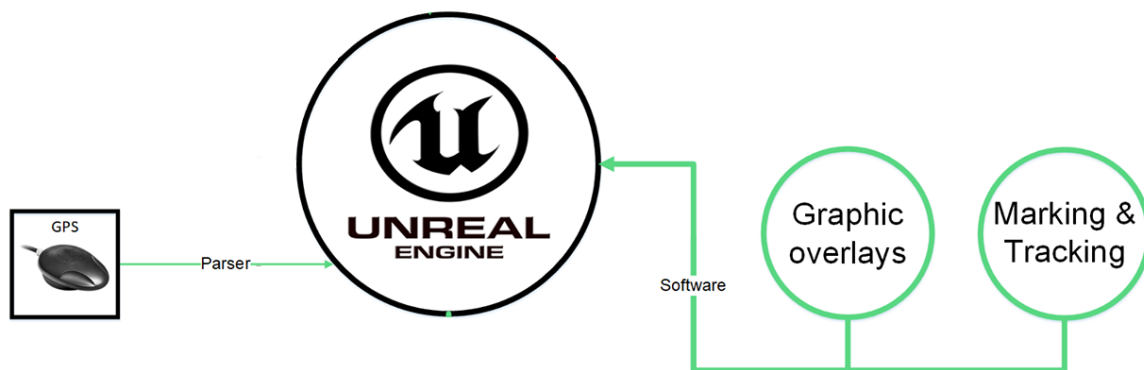


Figure 37: Showing tracking’s role in the Argos application as well as its dependencies

8.5.1.1 How it works

Tracking uses the camera feed displaying a real-time image of the real world as a background for overlays. Tracking takes in the systems position from the GPS parser and the geodetic coordinates of the targets. It uses these coordinates to calculate the relative position, unit direction, distance (F.8) and bearing of the target. The relative position is given as a vector describing x meters north, y meters east, and z meters up from the system, with negative values being the opposite direction. It then calculates the distance to the target in a straight line, and then the initial compass direction of the target (bearing).

The vector containing the targets relative position is normalized and used as in-world coordinates for a blue diamond in the virtual world to display where on the screen the user must look, to look directly towards the target. This is displayed in a similar matter to popular first-person shooter games like Battlefield.



Figure 38: Tracking activated displaying blue diamonds to show direction of targets



8.5.1.2 Why is this needed?

While driving, especially in stressful situations like vehicle combat, being aware of your surroundings is particularly important. It is also easy for a driver to lose their sense of direction when driving in unfamiliar areas, leading to disorientation, wrong turns, driving back the direction you came from without being aware or simply driving past the object of interest without noticing it. The tracking algorithm in Argos ensures that objects of interest is clearly marked with bright colors so that the user is fully aware of their surroundings at all times. This eliminates the possibility of driving in the wrong direction or past the target.

8.5.1.3 Contributions to other parts of the system

The tracking algorithm mostly depend on other parts of the system such as the GPS-parser for location and directional data, and the camera handler to give context to the information it calculates. However, it does also contribute vital information to the GUI handler. The GUI handler takes care of on screen information and is dependent on the calculations of the tracker to display information about the target like distance, bearing and directions.



8.5.2 Implementation

Both tracking and marking are implemented entirely through code. They are both implemented using known mathematical formulas and the calculations are accurate enough information that errors are unnoticeable. There are however errors due to the limitations of a game engine that primarily uses floating point variables. Because of this, there has been made some sacrifices to the accuracy of the system calculations, like the distance ignoring the curvature of the earth.

8.5.2.1 Target.h

Information on each target is stored in a Target data class.

The target class contains:

- The targets geodetic coordinates (latitude, longitude, altitude)
- The targets relative position to the system given in meters
- The targets distance from the system in a straight line given in meters
- The compass direction from the system to the target
- Other relevant necessary information such as id and description
- Get and set functions for different uses

The targets are stored in a target buffer that is accessible to all parts of the system.

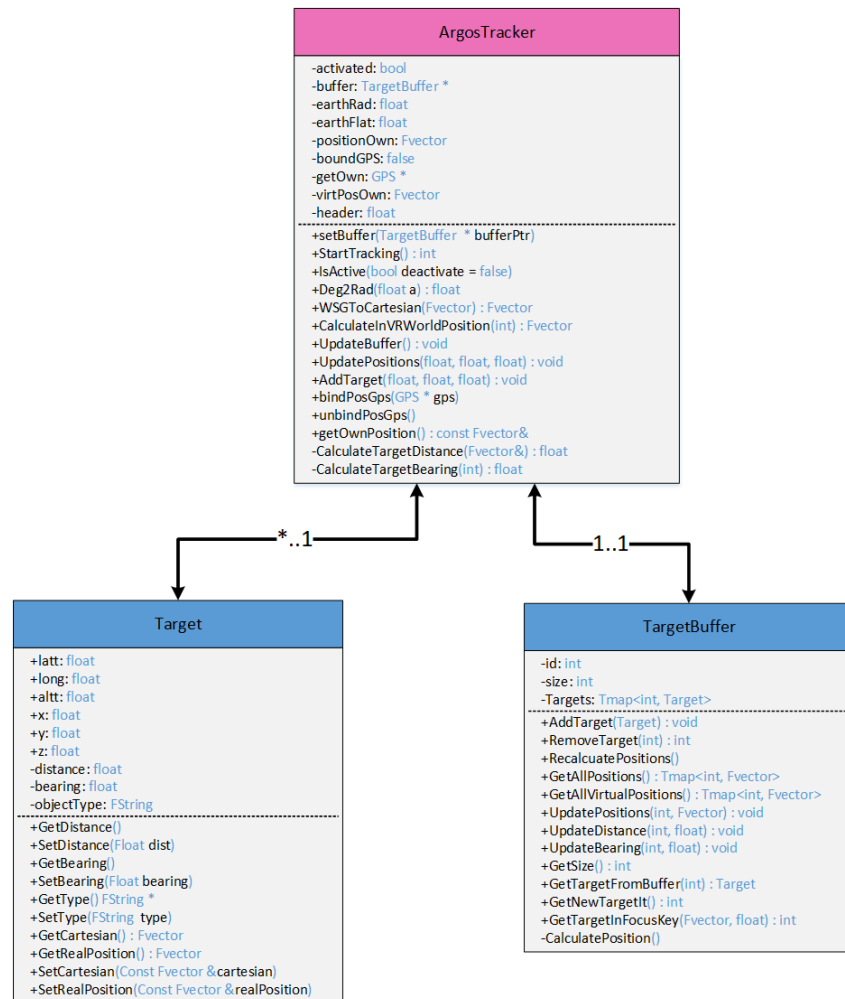


Figure 39: Class diagram for tracking



8.5.2.2 TargetBuffer.h

The target buffer contains a TMap<int, Target> for storing targets. TMap is Unreal engine's map container-class and is in this setting used to store targets with an integer value as a key. This TMap is put in a buffer class to more easily specify the type of information you want to get out of the buffer and making sure that each target has a unique id. The TargetBuffer class contains individual functions for returning component specific data. These functions include:

- Adding/Removing targets in the buffer
- Returning a single target from the buffer
- Returning all targets from the buffer
- Returning all geodetic coordinates
- Returning all relative positions
- Returning all distances
- Returning all bearings

This implementation eases the use of the buffer for other programmers.

8.5.2.3 Tracking

The computing class of the tracking software component is the ArgosTracker class. This class has a reference to the shared target buffer (as shown in Figure 39) and will always make sure that the information in the buffer is up to date. It does this by continuously going through the buffer and recalculating each individual target's relative position, distance and bearing based on new coordinates given by the GPS parser. The tracker assumes that targets don't move in the real world and relies on outside information to change a target's real world geodetic coordinates.

Targets are typically added to the buffer through the tracker as this class automatically calculates a targets relative position, distance and bearing as soon as it's added. When a target is added to the buffer it's added with only a description, id and geodetic coordinates. The tracker then fetches the newly added target and then does the necessary calculations. All targets are continuously updated through the UpdateBuffer function.



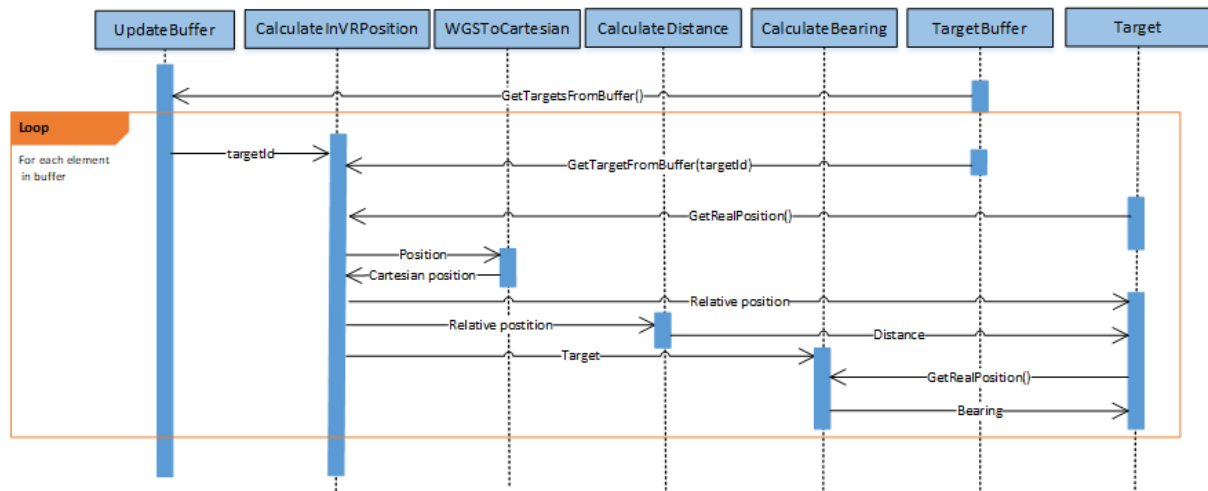


Figure 40: Sequence diagram for the UpdateBuffer function

8.5.2.3.1 Calculating relative position

Calculating relative position requires a few steps. It is done by first translating geodetic coordinates to geocentric coordinates. This gives us a vector with x, y and z coordinates in meters from the earth's center. Geocentric cartesian coordinates can be directly calculated from geodetic coordinates through the following simple formulae: [45]

$$\begin{aligned}
 r &= 6378137\text{m} \\
 x &= (r + alt) * \cos(lat) * \cos(long) \\
 y &= (r + alt) * \cos(lat) * \sin(long) \\
 z &= (r * (1 - 1/298.257223563) + alt) * \sin(lat)
 \end{aligned}$$

Note: Our implementation inverts the x value to mirror the result for correcting result. We also use the WGS84 radius for the earth instead of the radius of the curvature.

Since we need the relative vector from the system to the target, we also need to do the same for the systems coordinates. We could just get the relative position by subtracting the systems geocentric coordinates from the targets geocentric coordinates, but this causes some issues.



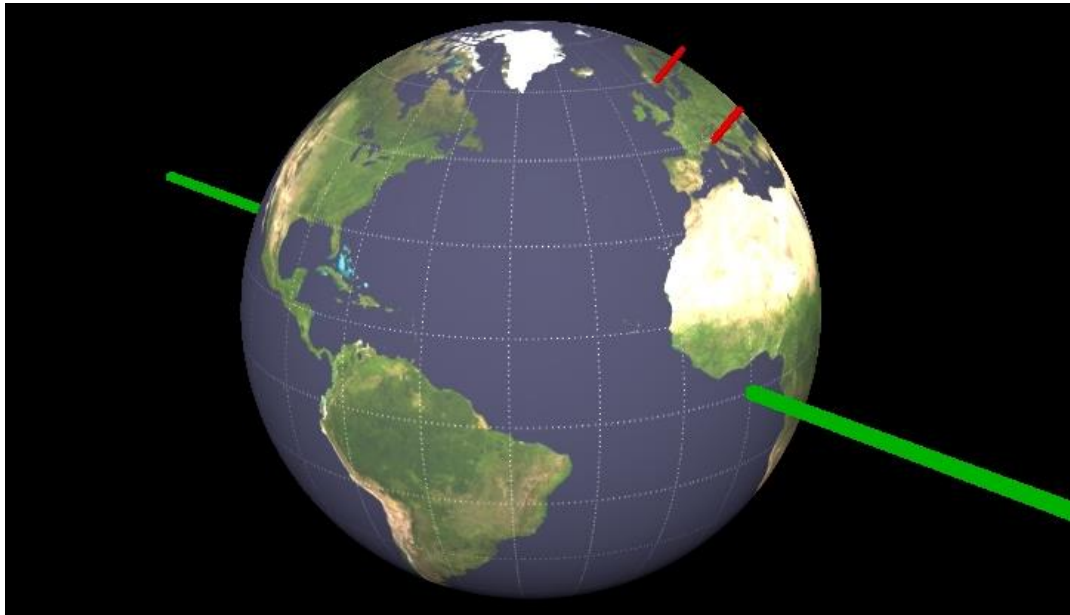


Figure 41: Red lines show geocentric coordinates relative to each other, green line goes through center of earth and null island (coordinates 0,0)

Figure 41 above shows two coordinates on earth. If we were to subtract one of them from the other the result will be a vector with unit vectors as the blue arrows shown in Figure 42. This causes some issues if you were to use the vector between these two coordinates as a directional vector for a target, mainly because this will not display the target in front of the system but rather more below.

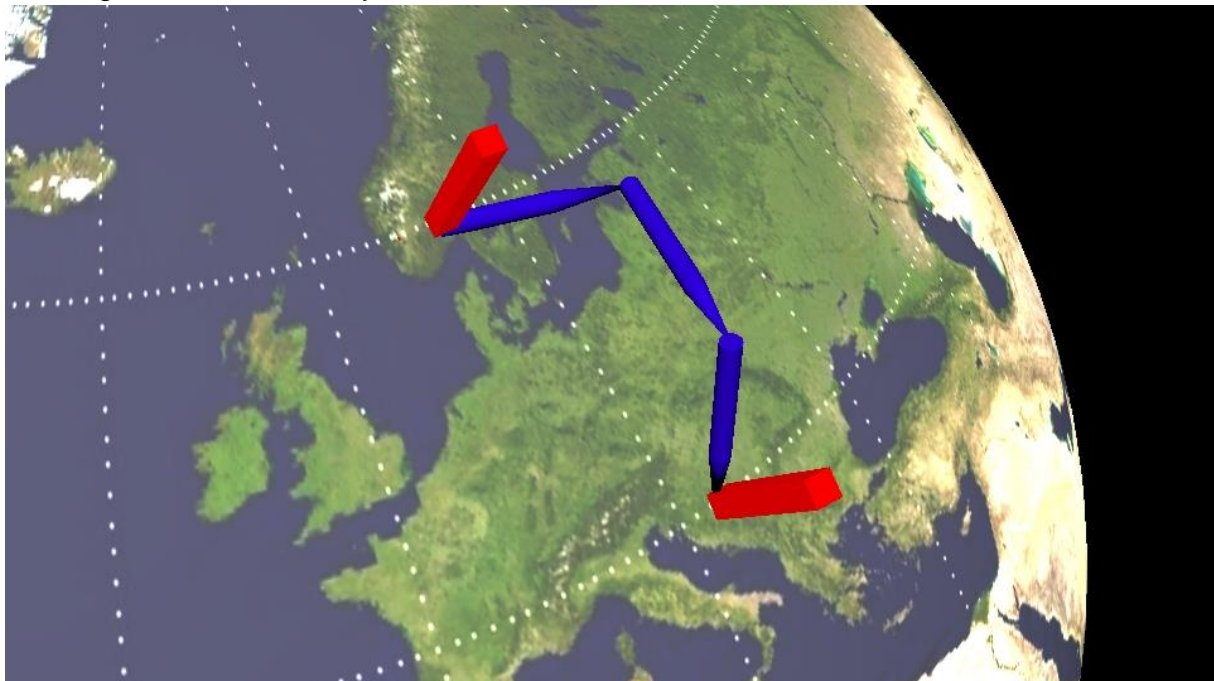


Figure 42: Unit vectors from one coordinate to another

To account for this, we need to rotate the vectors so that the vector points up in the perspective of the system by pretending that the systems location is at the north pole as illustrated in the left image of

Figure 43 below. This is done by finding the quaternion between the vectors of the systems geocentric coordinates and the geocentric coordinates of the north pole and then multiplying this quaternion with the target vector and the system vector. This shortens the unit vector on the z-axis and stretches the two others as shown in the right image of Figure 43 below.

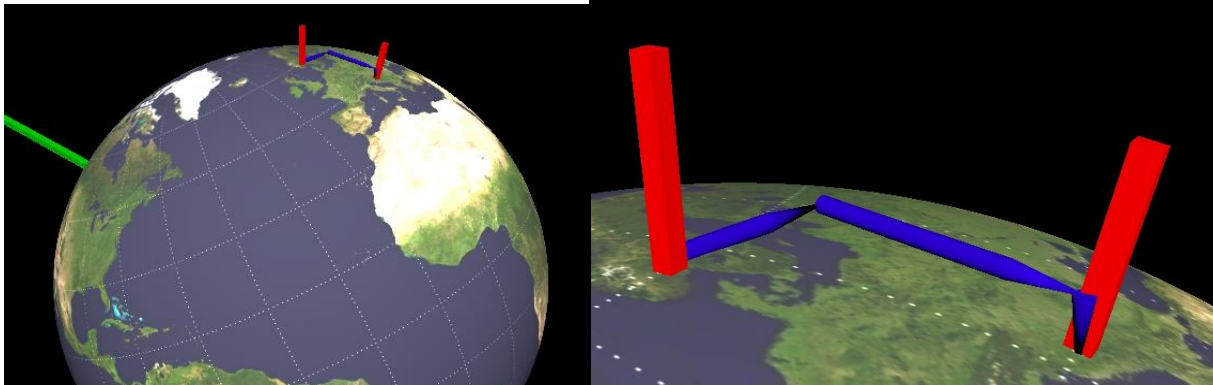


Figure 43: Left image shows earth rotated with the same coordinates now on top. Right image shows new unit vectors between the two coordinates.

The new vector between the two is more representative of their relative position to each other, however it's not good enough if you need it to fit with a compass direction. The left marker in both images in

Figure 43 above is roughly at the coordinates [59° north, 9° east], let's use this marker as the system in this situation:

If we track a target directly north of the system and place a marker at the relative coordinates in our VR-world it wouldn't appear directly north as 0 degrees on a compass. It would appear at 351 degrees north (or -9 degrees) which is the same as our longitude. This happens because north on the earth is towards a single point while north in our virtual world is in a fixed direction (positive x-direction).

To fix the rotation we no longer need the geocentric coordinates. Therefore, we subtract our new system coordinates at the "north pole" from the target's coordinates to get the vector between them. Finally, we need to rotate the vector around the z-axis (positive up) with the longitudinal degree of the system (in this case 9 degrees). This gives us the correct vector to use to place a marker (see Figure 44)

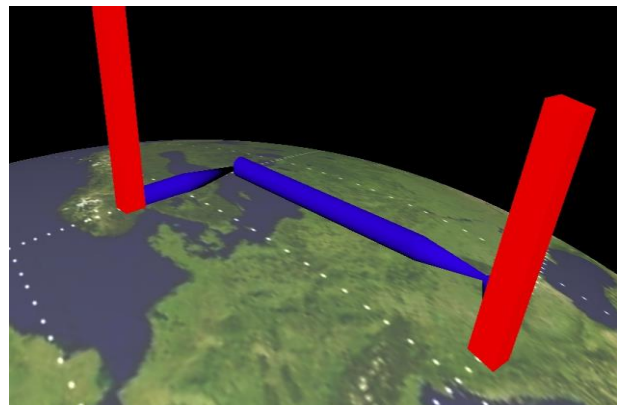


Figure 44: Final relative unit vectors



8.5.2.3.2 Calculating distance

Due to rounding errors in floating point numbers, which will be discussed later. The system uses Pythagoras' theorem for calculating the length of the relative vector between the two coordinates:

$$x^2 + y^2 + z^2 = d^2$$

8.5.2.3.3 Calculating bearing

For calculating bearing we've used the formula for "initial bearing" or "forward azimuth" which gives the forward direction to follow between two coordinates (Math formula from [46]):

$$\theta = \text{atan2}(\sin(\Delta\text{long}) * \cos(\text{lat2}), \cos(\text{lat1}) * \sin(\text{lat2}) - \sin(\text{lat1}) * \cos(\text{lat2}) * \cos(\Delta\text{long}))$$

Note: this bearing is not constant while following it, but does return the shortest path to the destination

In the formula Δlong is the difference in longitude between two coordinates, lat1 is the latitude of the system and lat2 is the latitude of the target. This is used mainly for informational purposes but can later be used to display markers on the compass of Argos.

8.5.2.3.4 The Tracker Actor

The Argos tracker is instantiated on an Actor that is placed in the virtual world inside Unreal engine. This actor is responsible for spawning in the markers on the coordinates stored in the buffer. It fetches the relative coordinates from the buffer and places markers at the positions equivalent to the normalized vectors. This essentially places the markers on the surface of a sphere around the viewpoint of the player. This is done to keep the markers at a constant size and inside the sphere that the live video is streamed on.

8.5.2.4 Challenges and solutions

During the implementation of tracking there has been a few challenges mostly caused by the limitations of, and lacking documentation of Unreal Engine. There was a long time spent sorting out the formulas needed and the rotation of vectors as well as sorting out the best method for implementing our ideas.

8.5.2.4.1 Sorting out implementation

We had an idea from the beginning of the project that we wanted to implement a heads-up display like that of first person shooter games, which usually has a compass at the top, a map, useful information in the corners and marker diamonds that mark out objects of interest or enemies. The problem was that we didn't know



how to do this. We had hopes in the beginning of implementing a deep learning image recognition algorithm. This turned to be quite difficult as most deep learning algorithms require internet or huge databases, neither of which we had access to.

We decided to start with the simpler way of tracking by only tracking static objects in the world, as this only needed the coordinates of the system and the objects. Then came the problem of how to implement this as none of us had any experience with working with coordinates. The first thing to look at was how did we wanted the markers to behave in the VR world, and what would be easiest to implement. After a few iterations over about 2 weeks we found the easiest implementation that required the least amount of processing power while still giving the wanted result. This method was to keep the virtual north in a constant direction and rotate the screen and camera in the driving direction of the system. This did however lead to a few problems later.

8.5.2.4.2 Formulas

With the method set, the next problem was find the formula which seemed easier then it turned out to be. Geodetic coordinates have been used since the 3rd century and with old math comes a lot of conversion formulas for more modern use, with several solutions for calculating the same thing. While searching through implementations for calculating the relative position of a target we found several formulas that looked like they would work, but when testing them never seemed to give the desired result or accuracy.

We landed on using the formulas for polar to cartesian conversion in 3-dimensional space and decided to use vector rotation to get the desired result for relative coordinates [45]:

$$r = 6378137 + \text{alt}$$

$$x = r * \cos(\text{lat}) * \cos(\text{long})$$

$$y = r * \cos(\text{lat}) * \sin(\text{long})$$

$$z = r * \sin(\text{lat}) * (1 - 1/298.257223563)$$

For the distance calculation, we decided we wanted to take the curvature of the earth into consideration where we quickly settled finding the angle between the vectors in radians and multiplying it with the radius of the earth to find the arc length:

$$d = \text{acos}\left(\frac{\text{pos1} * \text{pos2}}{|\text{pos1}| * |\text{pos2}|}\right) * R$$

Here R is the radius of the earth, pos1 and pos2 are the geocentric coordinates. This formula led to a few problems that will be discussed later.



For bearing, finding the correct formula was not easy since documentation on navigational terminology only describes terms using other navigational terminology. An example of this is the formula for finding “initial bearing”. This is described as finding the “forward azimuth”. A better example is the description for “final bearing”, which is described as the mirrored initial bearing. After a day of reading up on terminology we settled on the formula for initial bearing. This is because it shows the direction of the shortest path and on smaller distances gives about the same value as the actual compass direction [46]:

$$\theta = \text{atan2}(\sin(\Delta\text{long}) * \cos(\text{lat}2), \cos(\text{lat}1) * \sin(\text{lat}2) - \sin(\text{lat}1) * \cos(\text{lat}2) * \cos(\Delta\text{long}))$$

8.5.2.4.3 The problem with Unreal

Unreal has been a good help for many parts of the project, however Unreal has its fair share of problems. First of all, relative positioning. Due to our familiarity with the math functions in Unity we decided to test the formulas there first just to get a visual representation of our formulas, and they worked perfectly. But when it came to implement the relative positioning in Unreal it never seemed to return the desired result or anything close to it.

While debugging, the log was full of NULLs and NaNs. We spent three days trying to figure out why the quaternion was set to infinitely small values when trying to set it between two vectors without any luck. We read the entire documentation on Unreal's quaternion class. And it seemed that it was just broken. We finally decided to test out what would happen if we used the function called `FindBetweenNormals` instead of `FindBetweenVectors`, which according to Unreal's documentation “Generates the 'smallest' (geodesic) rotation between two vectors of arbitrary length” [47]. This for some reason worked perfectly with being the only thing changed. We still have no idea what `FindBetweenVectors` actually does.

Next problem on the list was the distance formula. For some reason the distance formula returned a value 3 times higher than it was supposed to do. After two more days of debugging searching through Unreal's math code, we concluded that this was caused by a critical rounding error due to Unreal's implementation of floating point values in all functions. This causes problems when dealing with high accuracy trigonometric formulas.

Because of this problem we decided to try to rewrite Unreal's math functions to using long doubles. This showed to be quite impossible as most of the math functions just inherited from precompiled headers and Unreal restricts access to the Standard Template Library. So instead we tried to use the Haversine formula, hoping that it would be more accurate, but it suffered from the same problem. Being at the fourth



iteration of this, we decided to ignore the curvature of the earth. This is when we decided on using Pythagoras on the vector for the targets relative position.

Before testing the bearing, we decided to try to spawn in our markers just to see what it looked like. We normalized the position so that markers would all keep the same size, and it looked exactly how we wanted, except they were in the wrong place. When testing the angle between the viewing angle and the vector the target was supposed to be at, we found the targets were shifted.

They were perfectly placed the closer north or south they were, but the closer they got to an east-/westward position the larger the error got, and it got weirder. We decided to use an old function we had written for drawing a debug sphere at the position of the targets. And they were drawn in a different area than the markers, and still, even they weren't at the right position, but they were at least closer. After looking over the code we noticed that the normalized vector that the debug sphere used was multiplied by 200 to increase their distance, so we decided to increase it to 2000 just to see what happened. To our great surprise, this worked, and was dead on. Meaning for some reason, Unreal changes the direction of vectors when normalizing and converges them back to the original direction of the vector when multiplying it with large numbers.

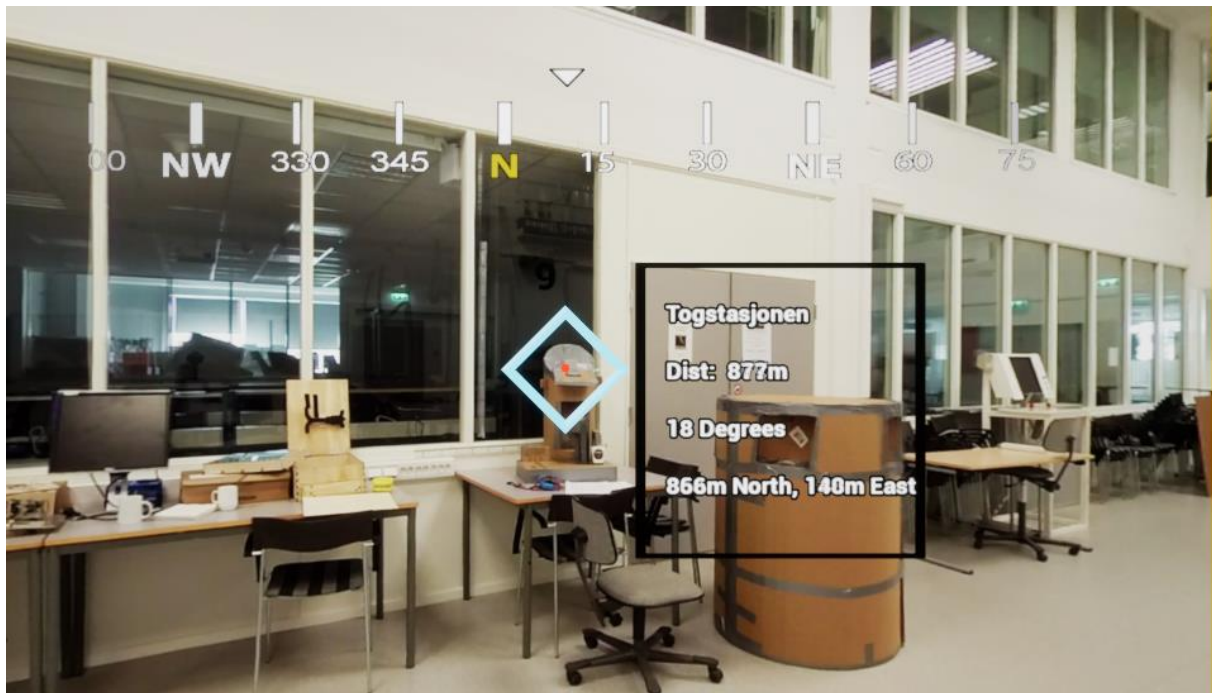


Figure 45: Text displaying information about the target



The final calculation challenge happened at the very end of the project when implementing the rotation of the camera and video sphere to the heading from the GPS. This seemed straight forward as we only needed to add the heading to the rotation of the camera. And again, Unreal caused problems because they didn't rotate at the same speed, or the same angle when just testing with a random value. This could lead to disorientation for the user, so we tried dividing the rotation input of the camera by 2 and it improved a little, but still wasn't correct, so we tested with 2,5 instead. This worked perfectly, the annoying thing was that we didn't know if this was constant or not. Therefore, we had to search the source code for an answer and it turns out that the rotation of cameras in Unreal, are as a default multiplied by 2,5.

8.5.4 Future ideas

Tracking objects has a big potential, with deep learning algorithms becoming smarter, faster and more available. Implementing tracking of dynamic objects in the might require some hardware upgrades as having for example an FPGA splitting the video feed and sending it to two different processors, one handling displaying the image and one handling image search, to not lose any performance on the latency. The markers can be further developed to include different colors for friendly and enemy targets.

8.5.4.1 Known errors and flaws that might need attention

There are a few known flaws that might need upgrading in the system, such as hardware efficiency, TargetBuffer missing certain information, the fact that coordinates of targets are currently hardcoded in the TrackerActor.cpp file and there is currently no function for adding removing markers through the user interface.

8.5.4.1.1 Hardware efficiency

Currently the update buffer function runs once on every tick of the game. This is rather unnecessary because it depends on the new position from the GPS for the values to change. The GPS only sends a signal once every second, which means that the UpdateBuffer function only need to run once every second as well.

8.5.4.1.2 TargetBuffer missing information

The TargetBuffer currently doesn't contain the information on the coordinates used by the marking class. As the objects are static this only need to be calculated once, however it could be useful to store this information in the buffer along with the rest of the information. This would have to be implemented in the Target class with functions in the TargetBuffer class to set and get this value.



8.5.4.1.3 Adding targets though file or external datalink

At the time of writing the only way to add targets to the buffer is through code. This was mainly just used for testing purposes, but due to lack of time, this stayed. A good idea would be to write a file with coordinates and make the system read from said file. Additionally, a one-way communication link could send information to the system.

8.5.4.1.4 Adding/Removing targets from the buffer through user interface

Implementing functionality to allow the user of the system add/remove targets could be useful as the user has a better overview of what has been taken out or is no longer of any interest. This can be implemented through the ArgosUI class by calling the add/remove functions of the tracker.

8.5.4.2 Ideas for other uses

Since the tracker already calculates bearing this can be used to implement markers on the compass in the HUD showing which direction the user has to turn to see other markers in their surroundings as there is currently no way for the user to know about all targets other than turning a full 360 degrees around.





8.6 Marking (TerraLens)

The purpose of this chapter is to give the reader a clear understanding of the TerraLens software component. This chapter describes the interaction between TerraLens and Unreal Engine, and how the component is implemented into the solution. Furthermore, it explains limitations, bugs, and other problems the Argos 2.0 group encountered during this bachelor thesis.

8.6.1 TerraLens Component	155
8.6.1.1 <i>Creating an interface between TerraLens and Unreal engine</i>	156
8.6.1.2 <i>Integration approach</i>.....	156
8.6.1.3 <i>Rendering (TerraLens Vs Unreal Engine)</i>.....	157
8.6.1.4 <i>Approach to creating a mini map with TerraLens</i>.....	157
8.6.1.5 <i>Conclusion</i>.....	157



8.6.1 TerraLens Component

TerraLens is a geospatial platform proprietary software made by Kongsberg Defence and Aerospace, which provide geographical information to the operator such as maps and road to the Argos Application. TerraLens is implemented into the Argos solution through a plugin made by the Argos group for the Argos project.

The TerraLens component implemented to fulfill the requirement [C.5] which derives from the user story 'Marking' from chapter 4 on Requirements. This user story was implemented into a use case (*UC3: Mark targets*) and included in the Argos 2.0 use case diagram.

UC3 led to the implementation of the *Marking sequence diagram*, which is the data flow model the OpenStreetMap component will follow and apply during its implementation.

The group was given access to the TerraLens libraries as well as a prototype example project from earlier iterations of Argos. This chapter we will uncover issues which led to a stalemate in the development of the TerraLens component, which is why the group choose to implement the OpenStreetMap component as a substitute for TerraLens.

TerraLens is a standalone component and does not contribute to other components functionality.

A Glossary for terms and abbreviation can be found at the Appendix section of the documentation.

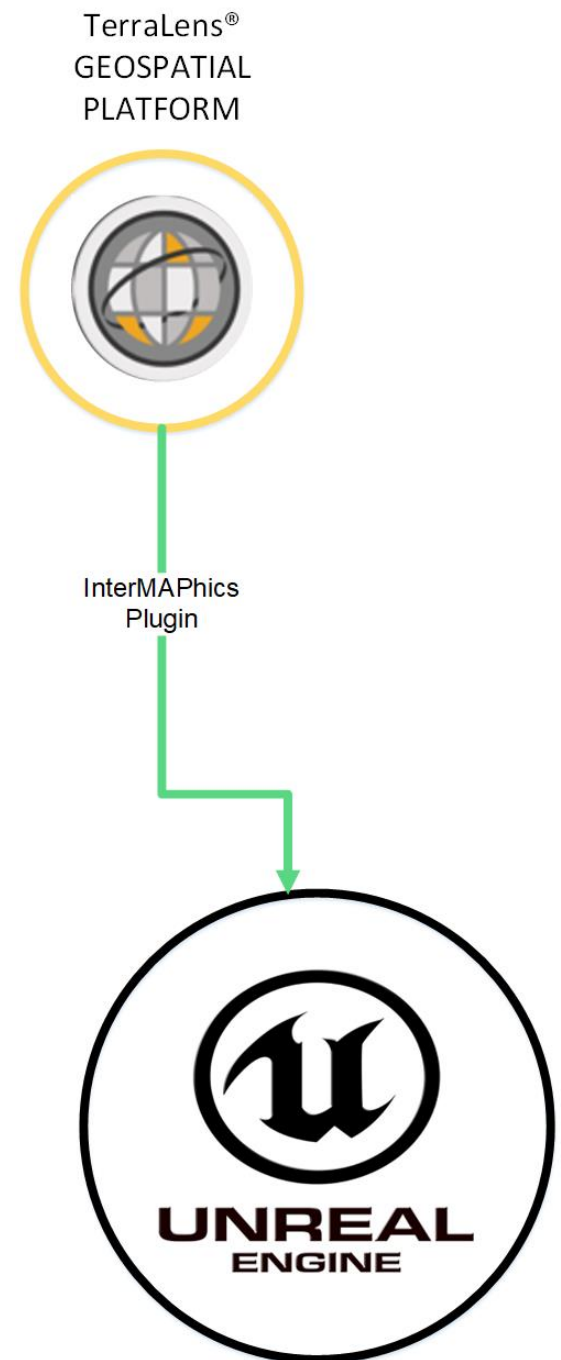


Figure 46: Architecture view of TerraLens



8.6.1.1 Creating an interface between TerraLens and Unreal engine

TerraLens is a third-party software and must therefore be incorporated into the Unreal Engine framework. We linked static dependencies TerraLens needs into the Unreal Engines build file. (This is explained in chapter 8.1) This ensures that we can access the libraries and header files in classes within the application solution.

8.6.1.2 Integration approach

TerraLens is a big library, which contains components for multipurpose functionality. The goal for the Argos 2.0 project was to integrate the same functionality which existed in earlier iterations of Argos; A basic 2D map and GPS tracking.

Now that we had access to the libraries, we started reading about the classes which included TerraLens functionality from the earlier iteration of Argos. This provided us with an understanding of which classes makes up a basic TerraLens application. We then started to implement some of these classes into the Argos 2.0 project, Unrealifying the C++ code provided from the demo application (for more information on Unrealifying, see chapter 8.1) This worked great, and we could build the Argos 2.0 application without errors.

- **ArgosMapElement**
An initializing class for the map functionality of TerraLens. This class uses the following classes to instantiate map. The map is drawn using OpenGL
- **ArgosContextManager**
A class which manages OpenGL contexts. Used to distribute resources to OpenGL components.
- **ArgosTrackLayer2D**
A container of 2d presentation layers. Multiple presentation layers can be attached to the Map Element. These layers can contain roads, terrain data and other relevant information. The ArgosTrackLayer2D class managed the visual priority of the contained presentation and layers.
- **ArgosTrack**
Base class for an object in the map which has vertical position data. Contains data such as velocity and geographic coordinates. Used to represent the Argos Operator
- **ArgosTrackManager**
Initializes, updates or delete tracks



- ArgosTimer
A simple time class which managed intervals between updates. This class was to be removed due to the tick () function UE provides.
- ArgosInitializer
Initializes a track and fill in values for demonstration purposes

8.6.1.3 Rendering (TerraLens Vs Unreal Engine)

TerraLens uses OpenGL API to distribute resources and render graphics. This proved to be a challenge when trying to render to textures in Unreal engine. Simply because the engine does not support direct OpenGL calls to hardware.

This did not seem too big of a deal, and we thought we could just rewrite functions which handled rendering to textures to use Unreal engine specific calls instead of OpenGL. We started replacing OpenGL dependencies, but soon figured out that OpenGL was too deeply incorporated in most of the fundamental TerraLens software. We did a lot of research on how we should go forth but found little to no information other than rewriting all the OpenGL dependencies. There are other developers that deal with this very same issue, so there is to be expected that a conversion plugin will be available sometime in the future.

We called in a meeting with our supervisor to discuss which options we had and concluded that we would stop development of TerraLens integration, to focus on other solutions which could provide somewhat of the same functionality. We ended up incorporation OpenStreetMap into the project as a substitute for TerraLens.

8.6.1.4 Approach to creating a mini map with TerraLens

OpenStreetMap is as said a substitute for TerraLens in the Argos 2.0 application, but it uses the same approach to draw textures into the world scene. This means that if one would get TerraLens to draw to a texture in Unreal Engine, one could just swop out the texture which today renders OpenStreetMap. More information on this can be found in the OpenStreetMap component subchapter in the documentation.

8.6.1.5 Conclusion

There might be a way of incorporating TerraLens into Unreal engine without removing dependencies from OpenGL, but we struggled to find a solution to this problem. We therefore chose not to spend more time developing a component we were unsure we would be able to finish. It's recommended for the next iteration of



Argos to have another go at this, when there is more time and might be more information on this subject.

It's also recommended to have a dialogue with developers of TerraLens to see if there is a way to remove rendering functionality within TerraLens.





8.7 Marking (OpenStreetMap)

The purpose of this chapter is to give the reader a clear understanding of the OpenStreetMap (OSM) component from the system architecture. The chapter will explain how the component is linked to the software architecture, and how the component was implemented into the Argos 2.0 solution. This includes communication between OpenStreetMap and Unreal Engine, and how we used OpenStreetMap in the application. Furthermore, the chapter explains limitations, bugs, and other challenges the Argos 2.0 group encountered during this bachelor thesis developing this component.

8.7.1 OpenStreetMap Component	160
8.7.1.1 File format	161
8.7.1.2 OSM plugin for Unreal Engine 4	162
8.7.1.3 StreetMapActor and Marking	163
8.7.2 Implementation	164
8.7.2.1 Argos Marker	164
8.7.2.2 Algorithm for converting geodetic to UE coordinates	165
8.7.2.3 Actor-placement in the OSM map.....	166
8.7.2.3.1 Insert actor and update it dynamically in the map	166
8.7.2.3.2 Insert static actors into the map.....	167
8.7.2.4 Integrating a mini map for the Argos application	167
8.7.2.4.1 Constructing the mini map	168
8.7.2.4.2 2D vs 3D and map rotation.....	170
8.7.3 Challenges	171



8.7.1 OpenStreetMap Component

OpenStreetMap is a third-party open source software component which provides geographical data such as maps and roads (OSM file) to the Argos Application. The component is implemented into our solution through a third-party plugin for UE4. The plugin is not restricted by license for commercial use and can therefore be included in an application which can be sold.

The OpenStreetMap component is developed to fulfill the requirement C.5 (Table 14: Constraints) which derives from the user story 'Marking' from chapter 4.1.1 on Requirements. This user story was implemented into a use case (*UC.3: Mark targets*) and included in the Argos 2.0 use case diagram (Figure 11).

UC3 led to the implementation of the *Marking sequence diagram* (Figure 15), which is the data flow model the OpenStreetMap component will follow and apply during its implementation.

Initially the group pushed towards TerraLens integration to fulfill *UC3* and requirements related to this component, but for reasons explained in *TerraLens Technical Document* (chapter 8.6.1.5), this proved too complicated for this bachelor thesis. As a substitute for TerraLens, OpenStreetMap was integrated into Unreal Engine.

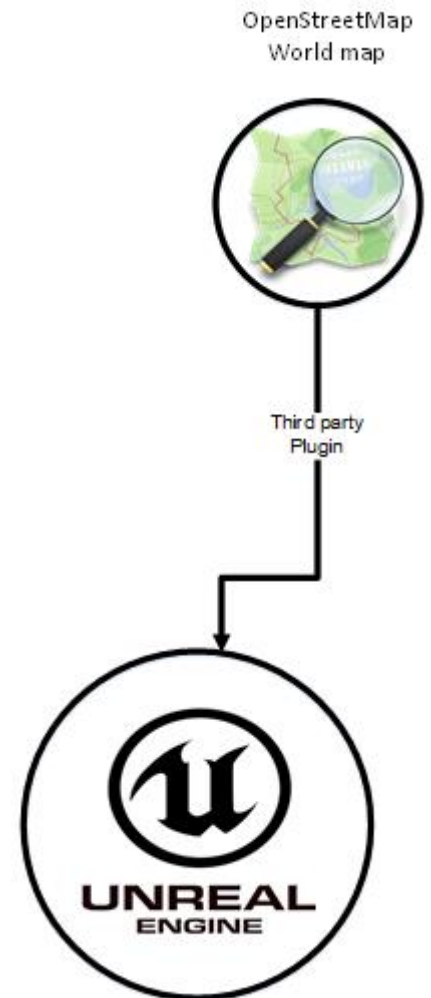


Figure 47: Architecture View of component

8.7.1.1 File format

openstreetmap.org [48] lets us extract data from its database in form of an OSM file. We select a desired area from the website, through either setting latitude and longitude range, or specifying the area with a rectangular box within the map environment (see Figure 48). When we export this data, we receive an OSM file for the marked area. These files contain various information based on registered data for that area. Some highly documented areas (such as NY) contain 3D building and shapes, while other areas contain 2d roads and building. The OSM file exported for Kongsberg, which we are using in Argos 2.0 is the latter.

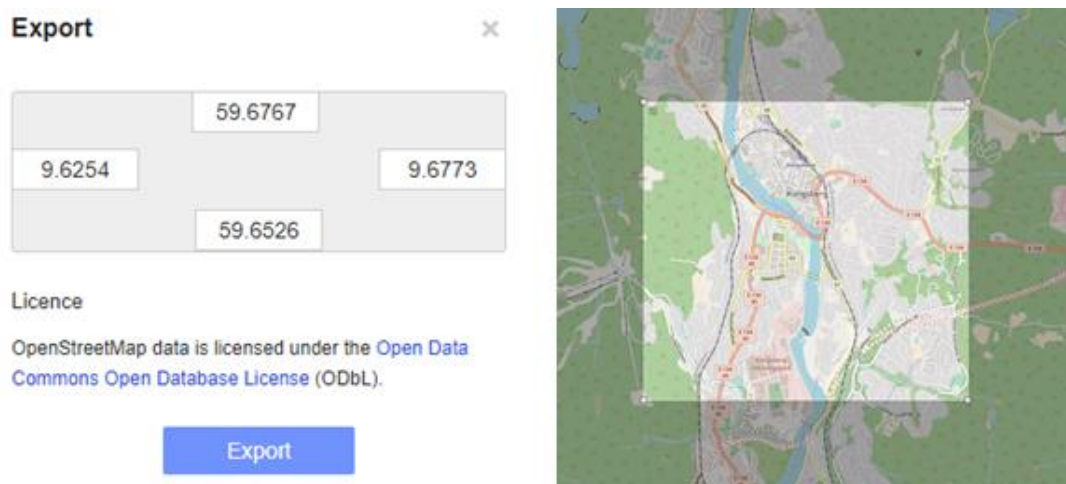


Figure 48: OSM file export example

OSM files inherit from XML files in format, and lists data for each node of information. Each node contains a unique Id and position, as well as other information not used within the Argos application. These files have a size of approximately 25MB, which is acceptable for the size of the map and the functionality it provides. Figure 49 displays how the .OSM file looks like in visual studio IDE.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <osm version="0.6" generator="Overpass API 0.7.55 579b1eec">
3 <note>The data included in this document is from www.openstreetmap.org. The data is made available under OdbL.</note>
4 <meta osm_base="2018-05-09T13:49:01Z"/>
5
6 <bounds minlat="59.6172000" minlon="9.5152000" maxlat="59.6927000" maxlon="9.7847000"/>
7
8 <node id="27042308" lat="59.6647137" lon="9.6965303" version="4" timestamp="2016-06-08T16:10:58Z" changeset="39888073" uid="715936" user="Gazer75"/>
9 <node id="27042311" lat="59.6648546" lon="9.6972411" version="4" timestamp="2016-06-08T16:10:58Z" changeset="39888073" uid="715936" user="Gazer75"/>
10 <node id="27042314" lat="59.6650259" lon="9.6975884" version="4" timestamp="2016-06-08T16:10:58Z" changeset="39888073" uid="715936" user="Gazer75"/>
11 <node id="27042318" lat="59.6662289" lon="9.6993131" version="5" timestamp="2016-06-08T16:10:58Z" changeset="39888073" uid="715936" user="Gazer75"/>
12 <node id="27042320" lat="59.6665723" lon="9.7000534" version="5" timestamp="2016-06-08T16:10:58Z" changeset="39888073" uid="715936" user="Gazer75"/>

```

Figure 49: .OSM file opened in VS2017 IDE



8.7.1.2 OSM plugin for Unreal Engine 4

To be able to use the data from the OSM file, we use a third-party plugin for converting OSM files to a format we can use within UE4. This format is named an “StreetMapActor”, which inherits from the “Actor” class. The StreetMapActor is a properly scaled, moveable game objects that we can drag into the worldspace within the application, which then becomes instantiated. Figure 50 displays conversion from OSM file to a StreetMapActor.

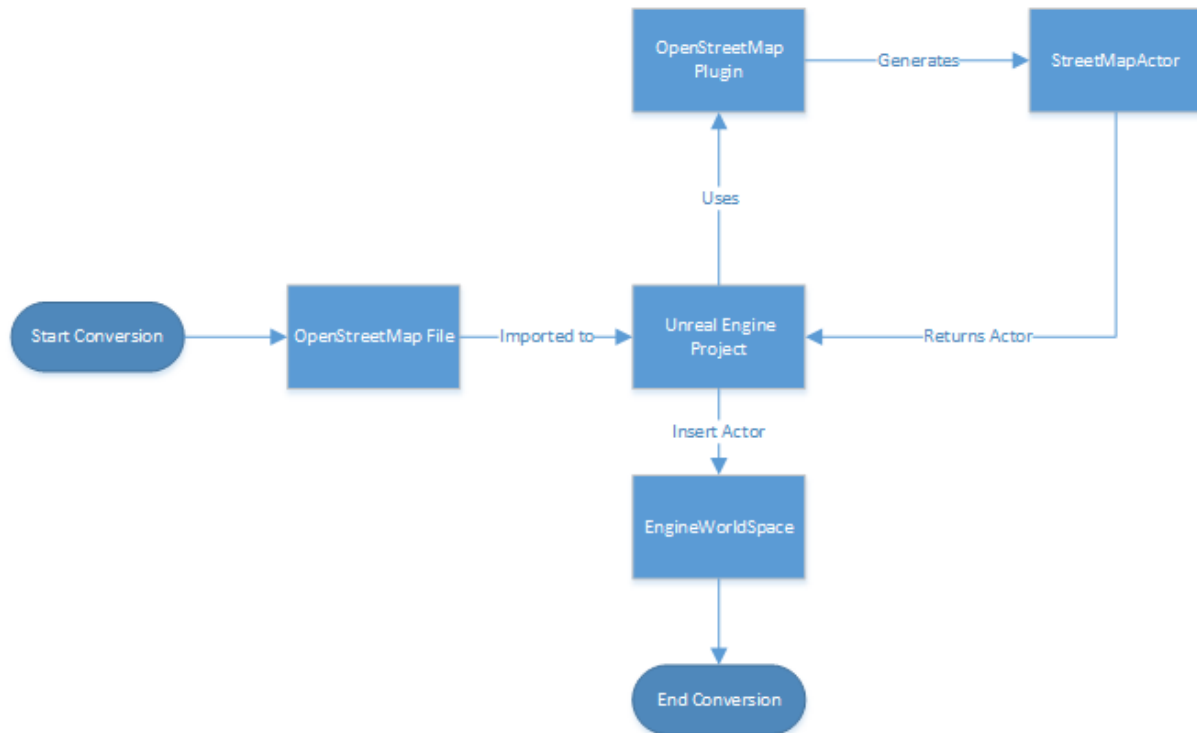


Figure 50: OSM conversion flow chart

The plugin is provided by Unreal Engine Developer Mike Fricker, which has licensed the plugin according to the *MIT License (MIT)*. This allow Argos to use, copy, modify, merge, publish, distribute, sublicense and/or sell copies of software which uses this plugin, and will therefore not provide our stakeholders with any legal issues related to this plugin. Following is the full license for the plugin used, written by Mike Fricker.

Copyright (c) 2017 Mike Fricker

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all



copies or substantial portions of the Software. [49]

8.7.1.3 StreetMapActor and Marking

When the plugin returns a StreetMapActor it can be applied to the scene by dragging it into the workspace. Argos then has a functioning map. The map by itself does not provide any functionality beyond acting as a map. We therefore need to develop additional functionality to make it display more information. This is where the “Marking” use case is being implemented. As mentioned in chapter 7 on Architecture, the marking use case inserts objects from the object buffer into the map. We will now look at how we achieved this.

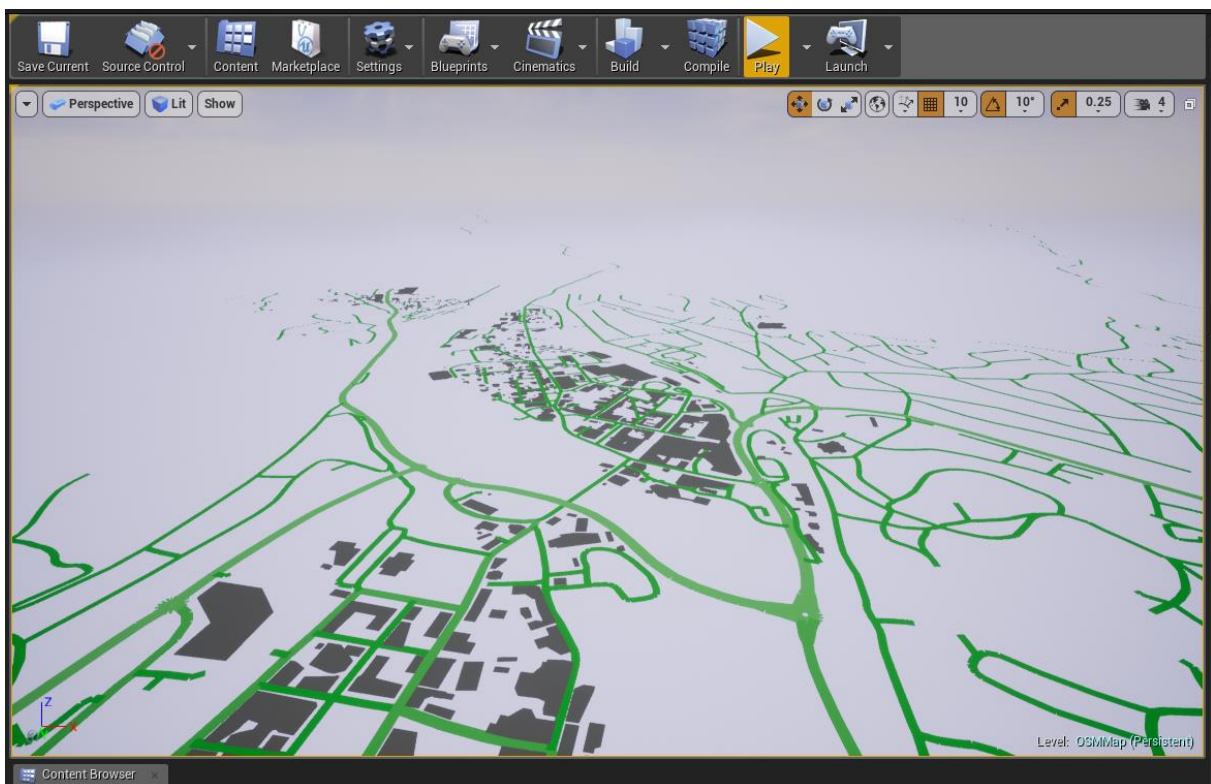


Figure 51: StreetMapActor in UE4 workspace



8.7.2 Implementation

8.7.2.1 Argos Marker

ArgosMarker is the class which will handle functionality connected to OpenStreetMap. The class has three functions which is connected to OpenStreetMap: Converting latitude and longitude coordinates to x/y coordinates, updating Argos position based on GPS input, and place targets from the target buffer into the map at the right position. The ArgosMarker class uses the GpsActor class to receive GPS coordinates, and the TargetBuffer class to fetch target positions.

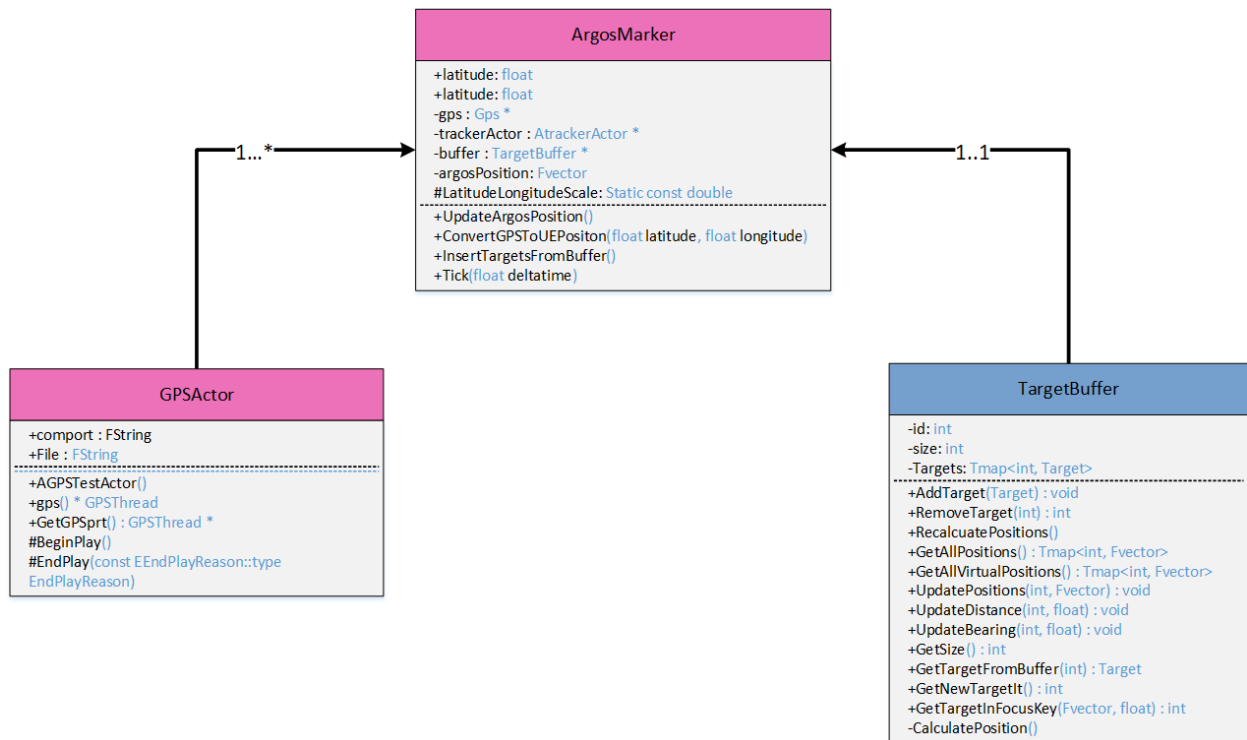


Figure 52: Class diagram for Use Case 3

8.7.2.2 Algorithm for converting geodetic to UE coordinates

Unreal engine does not understand geodetic data such a longitude, latitude, and altitude. We therefore need a way of converting these coordinates to coordinates unreal engine understands, which is metric 3D coordinates (x/y/z).

The OpenStreetMap plugin uses such an algorithm to import OSM files into Unreal engine. This algorithm uses Sanson-FlamSteed (sinusoidal) projection to determine positions of nodes relative to the center of the map. When a map is loaded into the scene, it fetches a relative longitude and latitude position. This position is the latitude and longitude values for the center of the map. It then calculates the distance from the center position to the node's longitude/latitude position and converts this into x/y/z coordinates. Figure 53 explains this algorithm using a flow chart diagram.

It is important to note that all insertion/updating of positions must use the same algorithm to ensure accurate positioning.

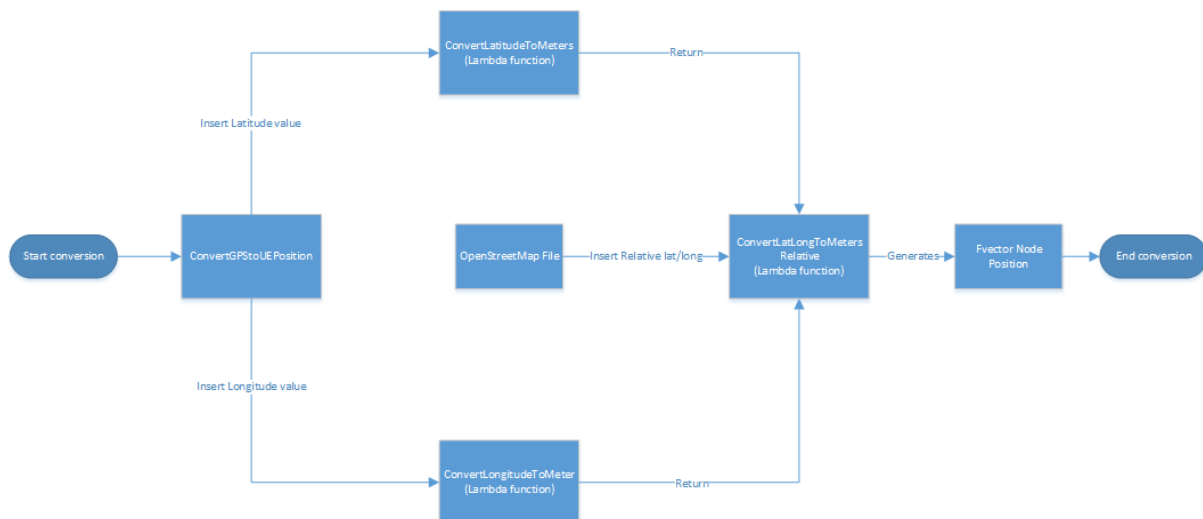


Figure 53: Flow chaw for coordinate conversion between OSM/UE4



8.7.2.3 Actor-placement in the OSM map

To understand how we could insert actors correctly into the world map, we had to take inspiration from the conversion algorithm the plugin used to insert nodes and implement our version of it. We started off by inserting an actor into the scene, which would represent our vehicle within the world-space. This actors position would change based on the latitude/longitude input it received from the GPS component

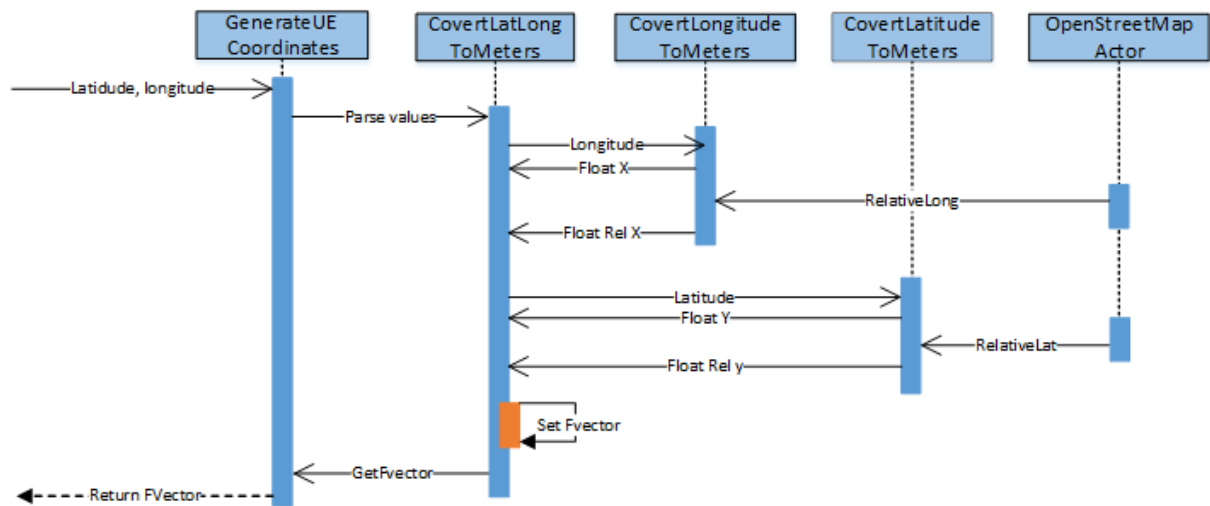


Figure 54: Sequence diagram for the GenerateUECoordinates function

8.7.2.3.1 Insert actor and update it dynamically in the map

Unreal engine provided us with a method which changed an actors position (*SetActorLocation*), so we implemented a function that fetched the data from the GPS, converted it into x/y coordinates, stored the data in a Fvector and updated the position within the map each update we received from the GPS. This functionality was implemented into the “ArgosMapMarker” class, which serves as an unreal engine component (script) that can be attached to any actors in the scene.

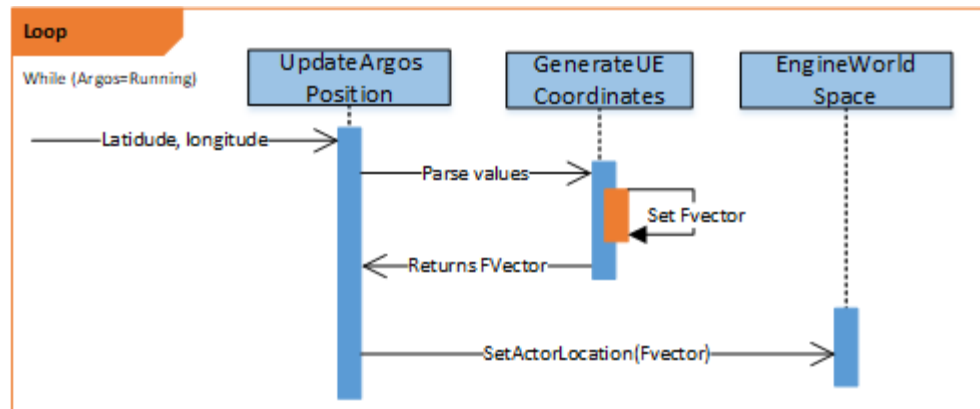


Figure 55: Sequence diagram for the UpdateArgosPosition function

8.7.2.3.2 Insert static actors into the map

To fulfill Use case UC.3: Marking, we needed a way of placing targets in the map at the accurate position. This function must fetch targets position from the Targetbuffer, convert their positions into x/y coordinates, and insert them into workspace. We named this function `InsertTargetsFromBuffer`.

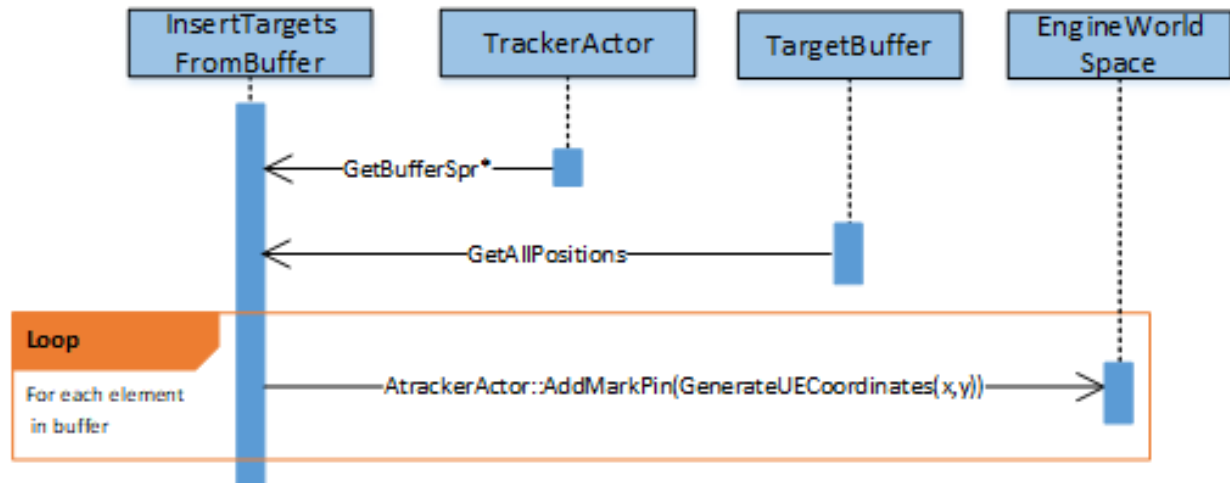


Figure 56: Sequence diagram for the `InsertTargetsFromBuffer` function

8.7.2.4 Integrating a mini map for the Argos application

The purpose of the functionality we've talked about in this document, is to project the map, actors position and more onto a surface which will act as a 3D mini map in the Argos application. This mini map will provide visual information about actor's position within the virtual real-world to the operator. The Argos mini map is implemented using the ***Unreal Motion Graphics UI Designer*** [47] developed by Unreal Engine 4 (referenced in GUI chapter).



8.7.2.4.1 Constructing the mini map

The mini map in the Argos application is made up of these unreal specific components:

- **ArgosVehicleActor**
The “parent” for all the following component. The Actor is a game object in the world space.
- **StaticMesh**
Piece of geometry that consists of a static set of polygons
- **Spring arm (MiniMapSpringArm)**
A line that tries to maintain its children at a fixed distance from a parent. The parent in this case is the ArgosVehicleActor, and the children are the camera and the SceneCaptureComponent2D
- **Camera (MiniMapCamera)**
Represents a camera viewpoint. This is attached to the spring arm which will make sure the distance from the actor to the camera is always relative
- **SceneCaptureComponent2D (MiniMapSceneCaptureComponent)**
Used to capture a “snapshot” of the scene from a single plane and feed it to a render target
- **Render target**
Renders to a regular 2d texture resource
- **Render target material**
A material that can be applied to a mesh, used to render the picture from the render target to the mesh inside the widget
- **Widget**
A user interface which can be dragged into the scene.

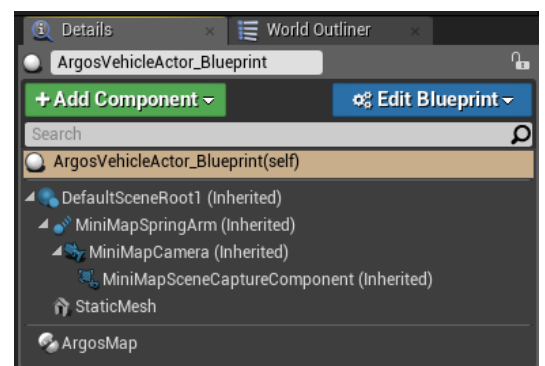


Figure 57: UE4 editor for ArgosVehicleActor

The `ArgosVehicleActor` is just an empty game object placed in the worldspace inside the Argos application. To give it a visual shape we add a `StaticMesh` component to it, which allow us to give it a mesh and a material.

Next, we add a *spring arm* component, which inherit from `ArgosVehicleActor`. At the end of the *spring arm* we attach a *camera*. This *camera* contains a *SceneCaptureComponent*, which capture a picture of the scene at the cameras x/y/z position. We will capture a picture each tick, which will render a seamless view for the mini map. We then create a *Render target* asset, which takes input from the *SceneCaptureComponent*. This ensures that the target for the renderer is the picture taken from the *SceneCaptureComponent*.

To place this picture onto a surface, we need to convert it into a material, which is called a "*RenderTargetMaterial*". This material is applied to a texture inside the widget UI element, which is instantiated and applied to the world space when the program is executed. Figure 58 displays how the mini map looks when running the Argos Application

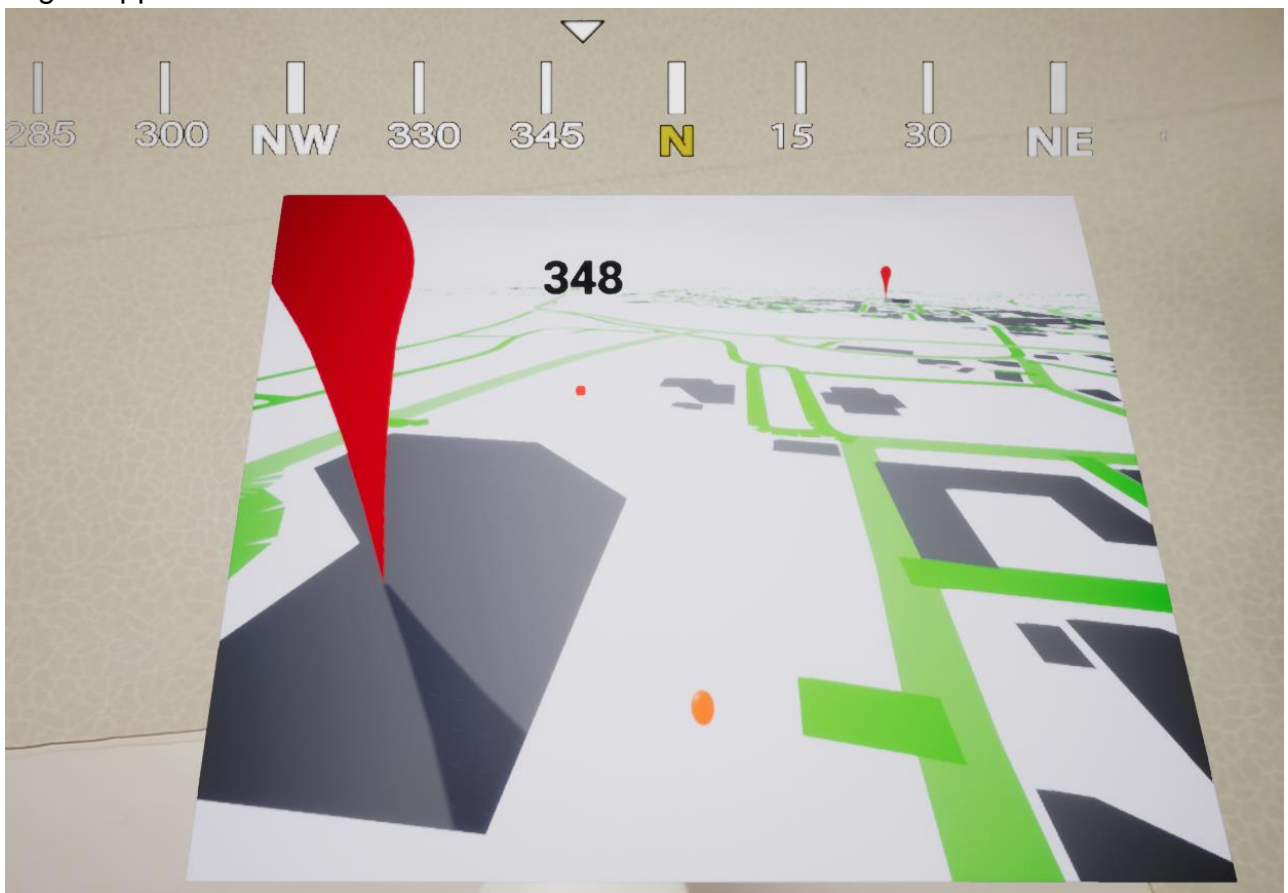


Figure 58: Minimap in VR scene



8.7.2.4.2 2D vs 3D and map rotation

The way we implemented the map lets us choose between top view (2D) and vertical view (3D). We decided to go for a 3D view, simply because it gives a more accurate representation of the surroundings than a regular 2D map. This can easily be changed by changing the angle and view of the camera component.

3D view provides more information, but also requires more implementation. One of the challenges we met was rotation synchronization. We wanted to synchronize the direction the operator looked in the real world, with the direction of the map. We discovered that we could rotate the **ArgosVehicleActor with the same input as the headset rotation, which would automatically rotate the camera because of the *spring arm* component which is attached to the actor. A great solution to a complicated problem.**



8.7.3 Challenges

Some parts of the integration / implementation of the OpenStreetMap component had challenges which we overcome to make sure that we accomplished the desired functionality. All these challenges are related to the ArgosMapMarker class.

- **Relative latitude / longitude**

The plugin has a variable for relative latitude / longitude, which it sets when converting an OSM file to a StreetMapActor. This variable is not 100% accurate to the center of the map. This is not a problem, but we have a hard time fetching the data stored in this variable, because it's only instantiated when converting the map and not at runtime. This means that we today fetch the data that is stored in these variables while debugging and setting it as a local variable in our relative longitude / latitude variables. This causes problems when trying to change the OSM map to another one. One of the tasks for future implementation of the Argos 2.0 project should therefore be to fetch this variable when loading a new map.

- **Weird constant**

When converting longitude to x coordinates, we get an output from the conversion algorithm which is the wrong position. For some reason, we get the right position by dividing our converted longitude value with 1.87. This is accurate for every position within the Kongsberg OSM map, but research for why this constant is needed is not yet conducted because of lack of time. It is also suggested to look at why this constant is needed in the next iteration of the Argos 2.0 Project.





8.8 GUI-overlays

This chapter describes the GUI-overlays designed in Unreal Engine for the final prototype of Argos 2.0.

8.8.1 Graphic overlays component	173
8.8.2 Unreal Engine	173
8.8.3 First GUI-prototype.....	175
8.8.4 GUI-overlays in VR	178
8.8.5 Classes connected to GUI-overlays.....	180
8.8.6 GUI Elements	183
8.8.7 Menu system.....	187
8.8.8 Interaction Method	192
8.8.9 Further development.....	194



8.8.1 Graphic overlays component

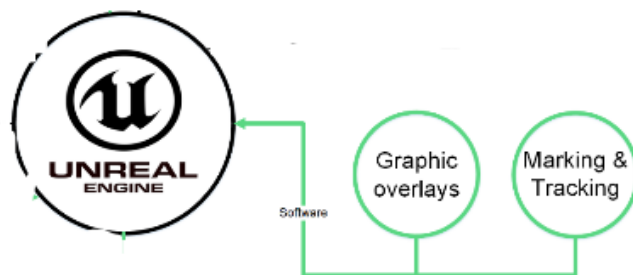


Figure 59: System architecture GUI

The graphical overlays represent the software displayed over the video feed in Unreal Engine. The marking and tracking is displayed through these overlays to show information about the tracked and marked targets.

According to our Use Case Diagram and UC:5 one of the main functions and requirements of the project is to have GUI overlays. GUI overlays is graphics, like text and images that are laid on top of the camera feed to display useful information to the user.

This component fulfils the functional requirement F.1, “The display must show information on a HUD”, that derives from the user story *GUI overlays* and the use case UC:5. (see Requirements chapter) This project contains GUI-overlays in the form of a compass, information box for targets, mini map and menus to choose features.

8.8.2 Unreal Engine

One of the main reasons Unreal Engine is the best option of framework for Argos 2.0 is because of the flexible and easy way to add *GUI-elements*. The easiest way to add GUI-overlays in Unreal Engine is using *UMG UI Designer*. For the prototype of Argos 2.0 we used UMG UI Designer (see chapter 8.8.2.1) and *blueprints* (see chapter 8.8.2.2) combined with C++ code (see chapter 8.8.2.3)



8.8.2.1 UMG UI Designer

Unreal Motion Graphics UI Designer (UMG) is a tool in Unreal Engine which is used specifically to create GUI elements. There are many built in functions in UMG, such as text-boxes, bars, buttons and images. These functions are called *Widget elements* and are edited by a *Widget Blueprint*.

In the Widget Blueprint there are two tabs, Designer and Graph. In the Designer tab the user may construct and design the visuals of the User Interface. While the Graph tab constructs the functionality of the Widgets. [47]

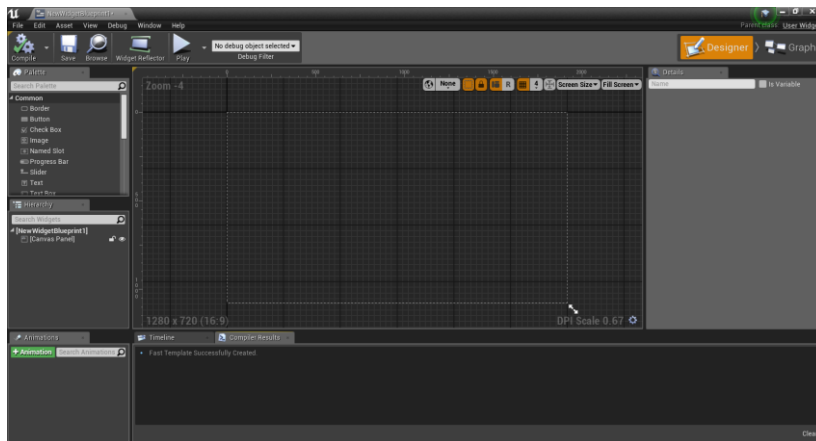


Figure 60: UMG UI Designer

8.8.2.2 Blueprints

For the first GUI-tests for Argos 2.0 we chose to use blueprints for testing purposes. The finished Argos 2.0 project will contain more C++ than Blueprints.

For the final prototype the project contains blueprints for graphical use and to move between maps (See chapter 8.8.7.1).

8.8.2.3 C++ Script

In Unreal Engine there is possible to create C++ classes to handle and store data that will be used in the Unreal Editor. In this project C++ scripts are used to handle all types of data.

8.8.3 First GUI-prototype

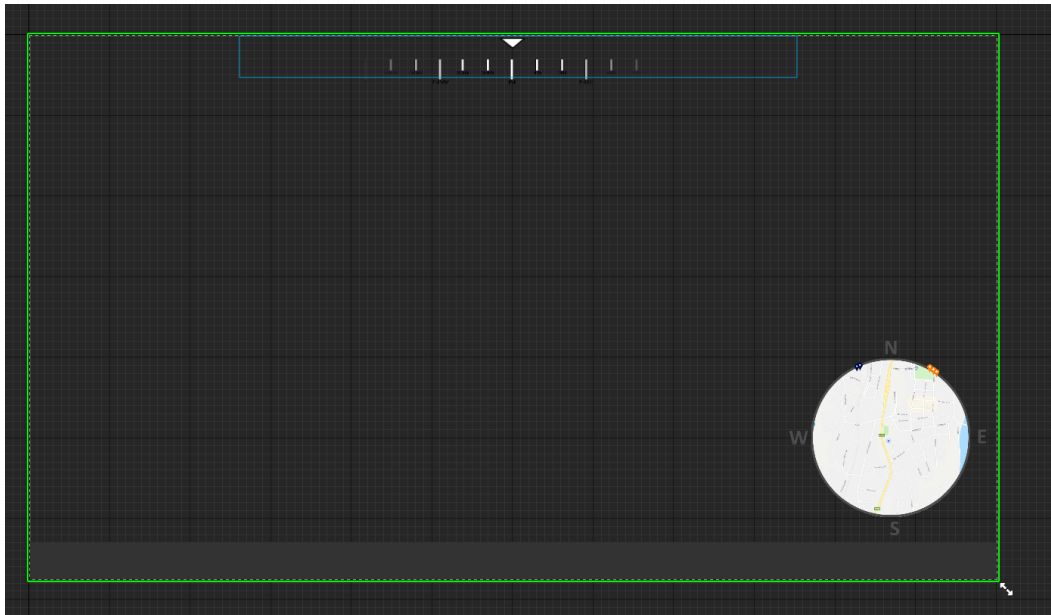


Figure 61: GUI-design

The purpose of the first GUI-prototype was to learn what possibilities Unreal Engine gives in terms of design and functionality and how to implement and create GUI-elements. The goal was also to produce some GUI-elements solely for the purpose of demonstration.

The elements added is a compass bar, a mini-map and a footer.



8.8.3.1 Compass bar

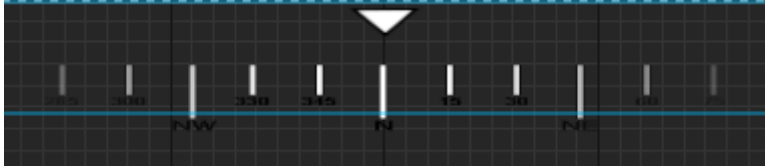


Figure 62: Compass bar

The compass bar is a scrollable compass which shows the users direction of view dependent to the four cardinal directions: north, south, east and west. As the user looks around the compass bar will rotate with the user to always indicate which direction the user is currently facing.

The compass bar consists of a slim picture indicating the initials of the cardinal directions and the angular degree the user is turning according to North. In Unreal Engine this picture is turned into a material to make it easier to edit using a Blueprint. The material is set to always fade out in the edges so that only the relevant directions is displayed.

There is a pointer anchored to the middle-top of the screen to indicates the direction forward of the user.

8.8.3.2 Mini-map

The mini-map shows the current location of the user on a map displayed in the GUI. This lets the user always know the current location and the area around. The map rotates with the user's rotation.

The mini-map is currently just an image to demonstrate the layout and idea of a GPS driven mini-map.

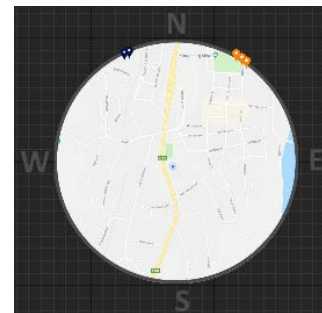


Figure 63: Mini-map Image

8.8.3.3 Footer

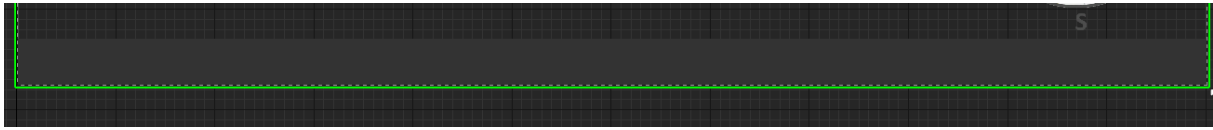


Figure 64: Footer

The Footer is the area of the GUI where the desired information is displayed in text and numbers. This area is currently in the bottom of the layout. Some ideas of information to be displayed here are: Coordinates, time, place, errors and so on.

8.8.3.4 The final prototype

From the first prototype the compass has a new look while the rotation has been converted from blueprint to C++ script.

VR-tests concluded that the footer does not work as intended. If it is located at the bottom of the layout it is not possible to see it in VR-view. If the footer is placed higher up in the layout it looks misplaced. The decision is to not use the footer. For future work there should be developed a different and user-friendly way to view constant information.

The mini map has been updated with live update from the GPS so that there is possible to see the map of the area the operator is currently in as the operator moves.

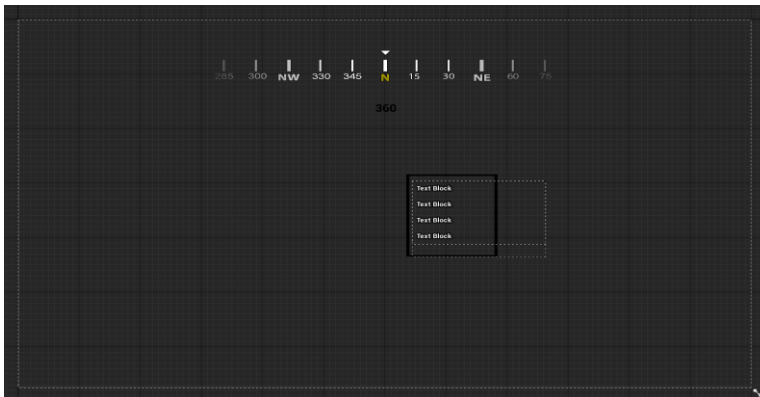


Figure 65: Final prototype widget



8.8.4 GUI-overlays in VR

There are two ways to display the canvases containing GUI-elements, called Widgets. The first is in the viewport. In the viewport, the widgets are set to the screen, completely two dimensional. This makes it easy to show the UI-elements as the project is run through the computer screen.

Since this project is using VR-headsets for controlling and using the program it is not possible to use the viewport. In VR, there is only possible to use three dimensional elements. 2D elements are reserved for the viewport and the VR-headset does not use the viewport.

The solution is to put the widget components in the world space, relative to the ArgosOperator pawn. By doing this, the widget appears as 3D components and exists in 3D space.

8.8.4.1 Argos Operator

The Argos Operator spawns in world space as the program starts running. The Argos Operator is a blueprint inheriting from the ArgosOperator class, which has been developed in C++ for this project. As the program is started the user of the VR-headset will be in control of the Argos Operator. The Widgets wants to follow the field of view of the user. To do this, the Argos Operator needs *components*.

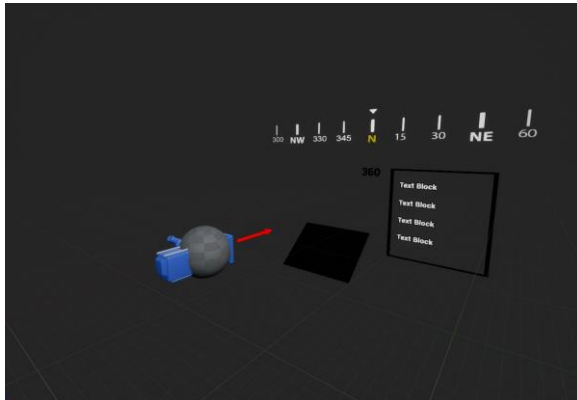


Figure 66: Operator in world space

8.8.4.2 Components

The Unreal Editor has a hierarchy for each actor. In this hierarchy there is possible to add components that are children of the base actor. In this case the base actor is Argos Operator. To display the GUI and make it follow the line of sight, the Operator needs to have a camera component. Since Argos Operator is the controllable actor, this camera component will automatically be the line of sight for the user of the program. This camera component has a child which is a Widget component. This widget component is the GUI that wants to follow the line of sight. Since it is a child of the camera component, it follows its movement.

In the Argos Operators viewport, the components are set relevant to the actors' location in the world space. As the Operator moves these components follow.

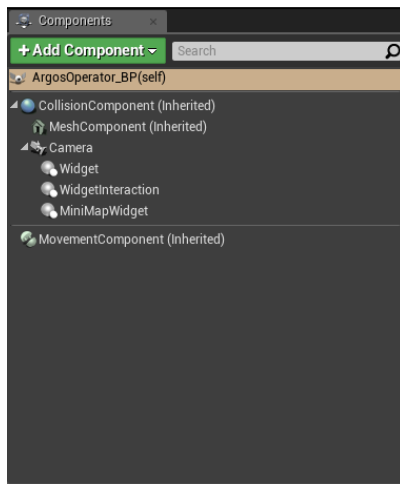


Figure 67: Component hierarchy of ArgosOperator



8.8.5 Classes connected to GUI-overlays

To handle the GUI-overlays there are created two base classes, here in pink. There is needed a class to send data to the GUI and one to process the data. To make these classes appear in world space it must be generated a blueprint that inherits from the corresponding C++ class.

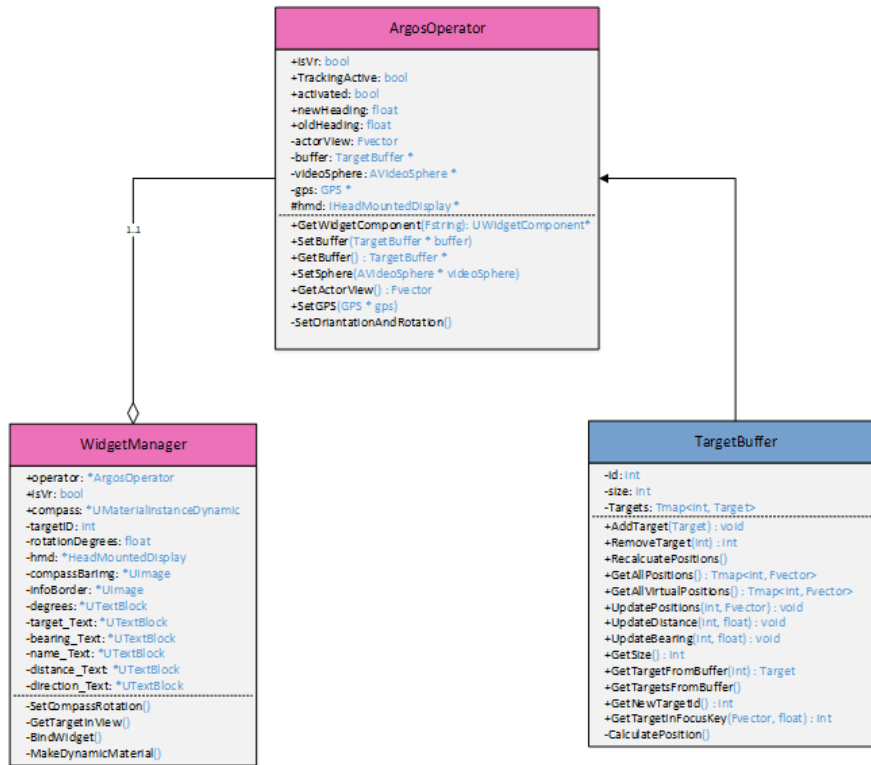


Figure 68: GUI-overlays class diagram

8.8.5.1 ArgosOperator class

This is the base class for the Argos Operator (see 8.8.5.1). This class' basic functionality is to represent the Argos Operator, the playing actor. It collects movement data from the VR-headset and mouse, it has functions that help connect the Operator in world space to other classes and handles interactions with other actors.

This class gets data from the target buffer, to decide which target the Operator is currently looking at.

8.8.5.2 WidgetManager class



The WidgetManager class handles the manipulation of data connected to the Widgets in Unreal. This class binds the GUI Elements from the UMG Designer's Widget to code variables that can be handled and manipulated in C++.

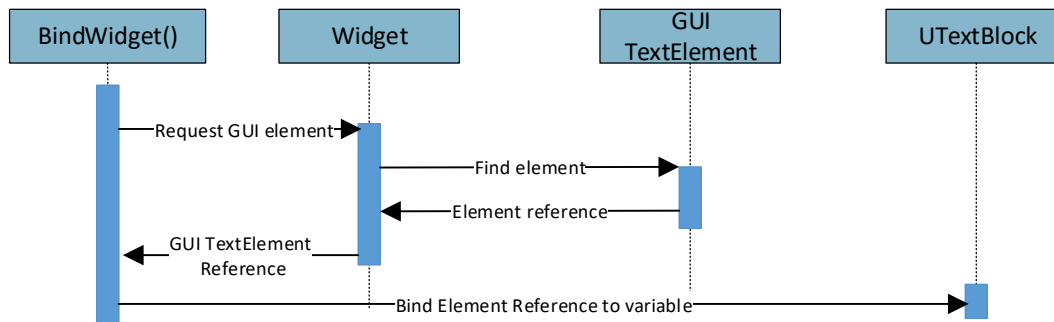


Figure 69: Sequence diagram BindWidget

This class gets and displays the information of targets from the target buffer. The class gets rotation data and target buffer data from the ArgosOperator class. The rotation data is set to the compass rotation and the target buffer data is used to see which target the Operator is looking at and gets the information from this target. All information displayed through the elements connected by BindWidget().

8.8.5.3 ArgosUI class

Decides the function of the buttons in the Argos UI menu.

This class has the main responsibility to turn on and off functions such as, recording, tracking, setting visibility of GUI-overlays and quit the current recording.

This class gets widget components from the ArgosOperator class to decide which widget that will toggle its visibility

8.8.5.4 MenuWidget class

Decides what happens as the menu widget is set up and tears down. Here the overlays toggle off as the menu prompts and toggles on when the menu tears down.

8.8.5.5 MainMenu class

Decides the function of the buttons in the Main menu. This class initiates the choice of recorded video and initiates live stream.



8.8.5.6 Cooperation between classes

The Widgets that is being managed by the WidgetManager is components of the Argos Operator. When toggling the visibility of GUI in Argos UI, we get a reference to ArgosOperator and gets the desired widget components, which toggles the visibility in ArgosUI as the button is pushed.



8.8.5.7 VR vs Viewport in code

There is different code required to get the same user data, such as location and rotation of the users' line of view, as you use a VR-headset or the viewport. In this case, there is an editable bool "isVR" in both ArgosOperator and WidgetManager classes. This needs to be set true if a VR-headset is connected, false if not. If isVR is true, we extract data from a HeadMountedDisplay which represents the location and rotation of the VR-headset. If *isVR* is false, we get rotation and location from the Argos Operator in the world space.



Figure 70: isVR boolean

8.8.6 GUI Elements

The GUI elements are the different overlays that are visible to the Operator during runtime. This consists of the compass, information boxes and mini map.



8.8.6.1 Compass



Figure 71: Final prototype Compass

The compass for the final prototype of Argos 2.0 works the same way as the first prototype (see chapter 8.8.3.1). The difference is another visual model and the compass rotation is handled by C++ code and not blueprint. The WidgetManager class binds the compass model in UMG to code.

The SetCompassRotation function sets the rotation of compass image to follow the rotation of the Argos operators' field of view's yaw rotation.

Below the compass there is a number, DegreeText, which indicates the degrees from North relative to the direction the operator is currently looking. The function makes this happen in the GUI.

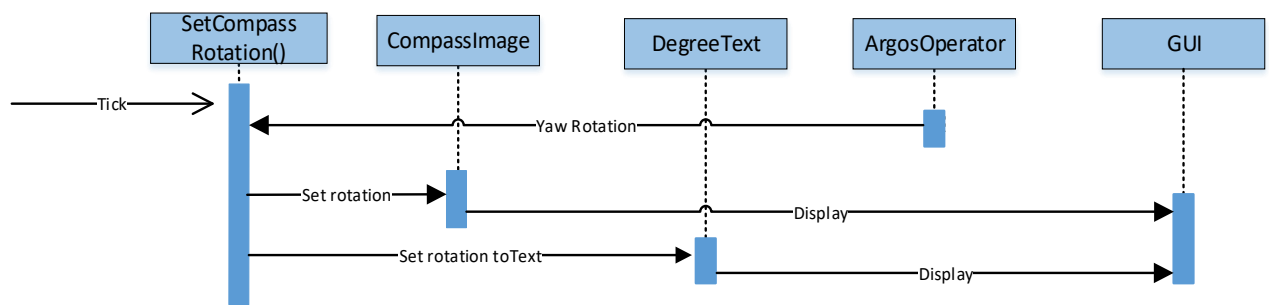


Figure 72: Sequence Diagram SetCompassRotation

8.8.6.2 Target information boxes

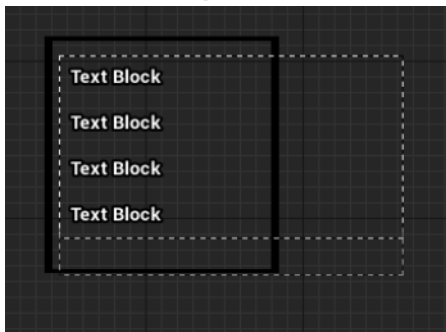


Figure 73: Target information box

These boxes show the information of the target the Operator looks at, as it is being looked at. The box disappears as the Operator looks away from a target. The text is bound to the target buffer and extracts information from the targets in the target buffer. The information shown is:

- Target name: the type or name of current target.
- Distance: The distance between the operator and the current target.
- Bearing: The angle between the direction the operator is looking and the target.
- Direction: The compass direction the target is located in.

The function `GetTargetInView` make this happen. The `ArgosOperator` gets to know the targets in the target buffer and indicates which of these targets are in the line of view. The function then extracts the information from the target in view and sets these to the GUI elements that are displayed in the GUI-overlays.

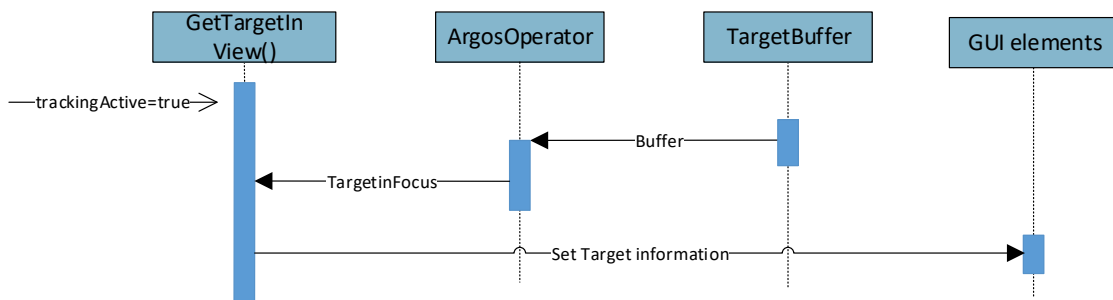


Figure 74: GetTargetInView Sequence Diagram



8.8.6.3 Mini map

The mini map is a widget for displaying the map of the current real-world location of the Operator. The Operator will then always know the location and the environment around. More information about this in the Mini Map Chapter.

The Widget has an actor that is placed in world space. It is placed down “On the floor” of the Operator’s view. The reason for this placement is to not take unnecessary attention from the operators view.

The mini map widget follows the players forward vector. Which means that its position is always relative to the operator’s field of view.

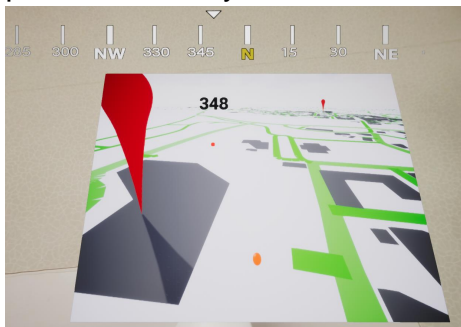


Figure 75: Mini map widget

8.8.7 Menu system

The menu system of Argos is made to let the operator choose what should be shown. It is possible to show different maps and toggle functionality such as tracking and marking. There is two menus:

The main menu

This is the menu that prompts as the Argos application starts running. Here the operator may choose either to watch recorded streams or to view the live stream. The operator may also quit the application.



Figure 76: Main Meny widget

The Argos menu

This menu is prompted as the button “M” is pressed on the keyboard. This menu lets the operator choose:

- If marking is enabled or not
- If tracking is enabled or not
- If GUI-overlays is visible or not.
- Quit the current stream and return to main menu.

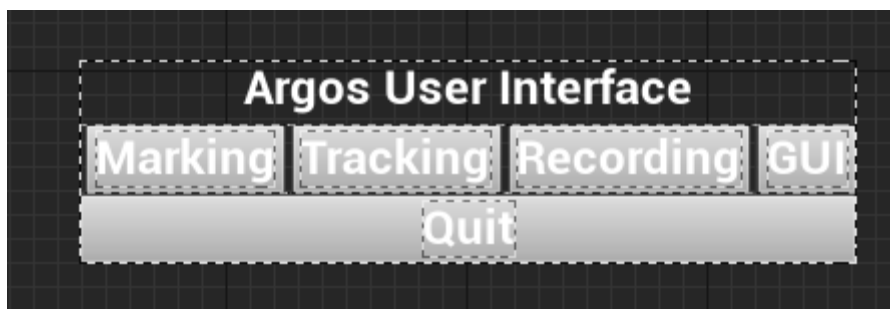


Figure 77: Argos UI Widget



8.8.7.1 Final menu system

The first menu system did not survive the transition from viewport, 2D to VR, 3D. It is unexplainable to the group at the current state, but the programmed menu interface class does not work as the menus is in world space. Lack of documentation and time forced another solution to be developed for moving between maps.

Menus in world space

To make the menu widgets interactable we need to put them on an actor in world space. The actors have a widget component which contains the menu. The menu actors' location is set relative to the Operators location in world space, so it appears at a comfortable length away from the operators' eyes.

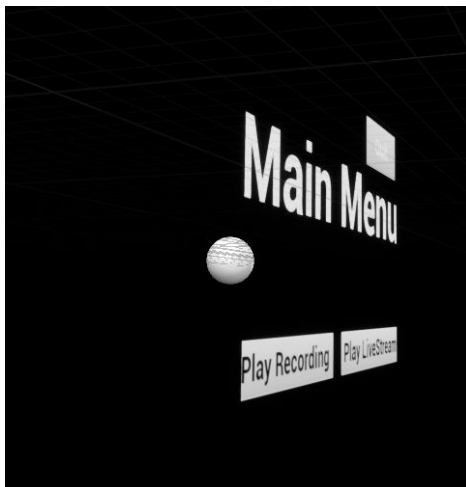


Figure 78: Menu Actor in World Space

Transition between maps

The main menu is about choosing the map that is going to be viewed. The buttons for changing maps does not work with use of the old system because the menu interface class is never reached for unknown reasons. The fix was to connect the buttons to blueprints using an “Open map” function at button clicked.

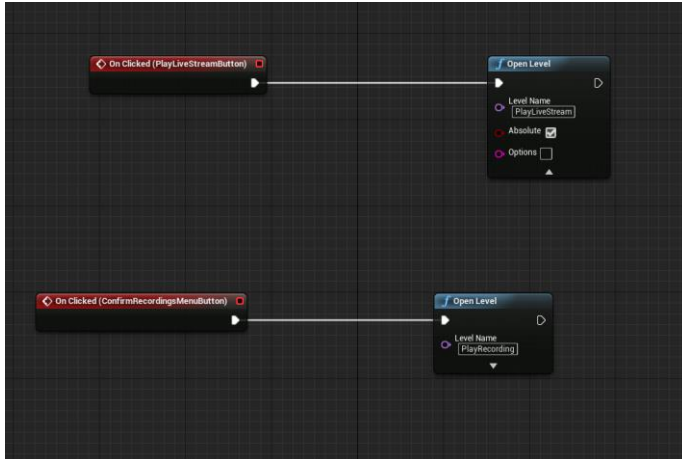


Figure 79: Transition between maps



MainMenu

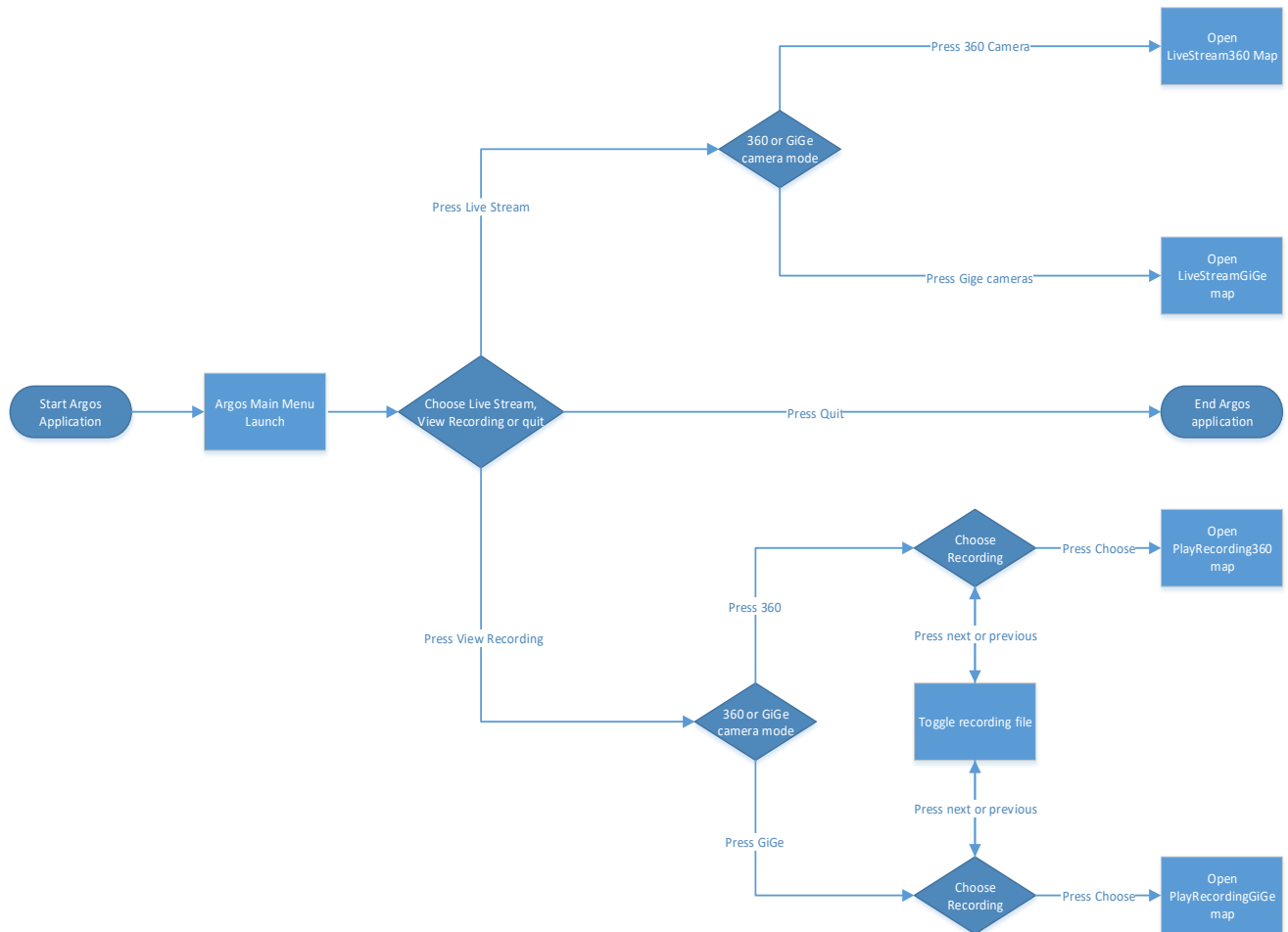


Figure 80: Menu system flowchart

This diagram shows the flow of operation to the main menu from the start to the end of the application. As the Argos application launches the main menu is launched. Here there is a choice to view live stream or recorded video with either the Gige cameras or the 360° camera.

As one of these “View Recording” is chosen, there is the last option of choosing which video that will be shown. As a video is chosen and “Choose” is pressed, the map dependent to the earlier choices will be opened.

In the flowchart, the round boxes are start and end. The rectangular boxes are functions while the diamond shaped boxes are choices.



ArgosMenu

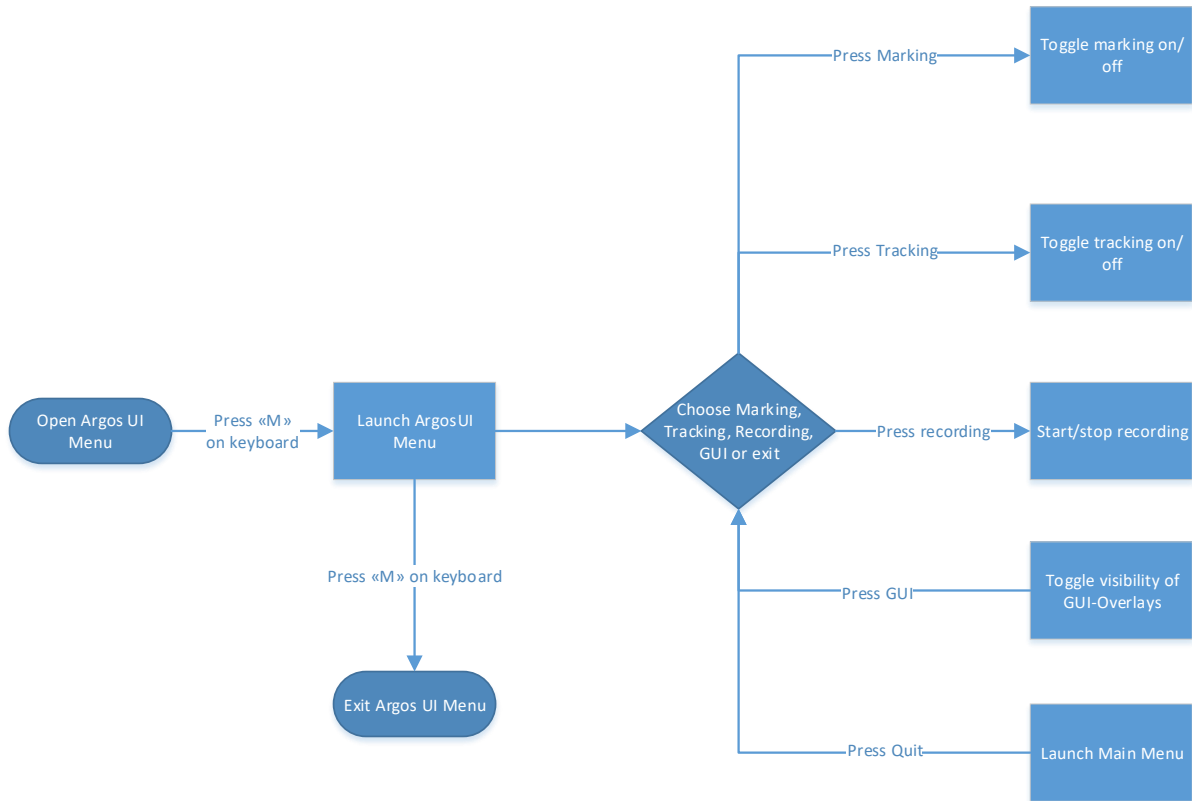


Figure 81: Argos GUI flow chart

The ArgosMenu is a popup menu that appears as “M” is pressed on the keyboard. At the first test “M” toggled the visibility of the ArgosMenu. This function only toggled the visibility of the Menu, but it was still there in world space, taking up space and overlaps the widget interactions line trace (see 8.8.8.2). The second test had the purpose to solve this problem and the result is to shrink the scale of the menu to 0 at begin play. When “M” is pressed the scale of the menu is set to 1.

The effect is the same and it does not take up world space when it is not in use.

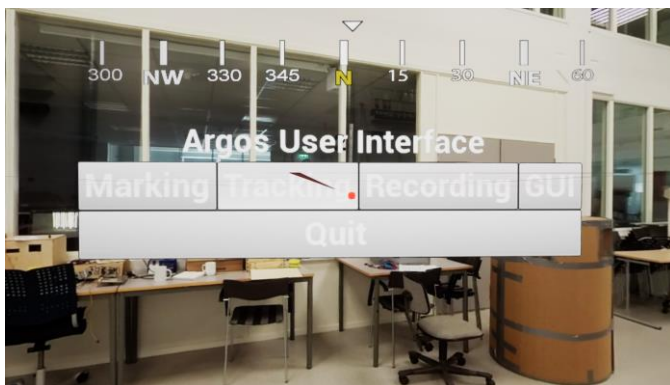


Figure 82: Argos UI in Runtime



8.8.8 Interaction Method

The operator shall be able to interact with the simulation world to use menu buttons and look at desired targets. This works differently as the program runs in VR or in the viewport. In the viewport the operator may interact with simply using the mouse cursor. In VR it is not desired to use the mouse cursor, but rather the line of sight to determine what the operator is actually looking at. There has been tested two ways of doing this in this project:

8.8.8.1 Raycasting

The first idea was to raycast a line that traces the forward vector of the Operator's line of sight and get information from the actor it is currently hitting. Upon learning that Unreal has a built-in component for this exact use the raycast was put out of use.



8.8.8.2 Widget Interaction Component

Unreal Engine has newly developed a component designed to interact with widgets. This project uses this component to display a crosshair representing the line of sight. This crosshair has the same function as the mouse hovering. An example is: if the widget interaction components crosshair is hitting a button in the menu, the button recognizes this as if the mouse cursor hovers on the button. Press “Shift” on the keyboard to simulate a click of the mouse. The widget interaction components crosshair is indicated by a red dot.



Figure 83: Red Dot Widget Interaction Component

The widget interaction component is a child component of the Argos Operator Actor's camera. Which means that it follows the camera's line of sight.

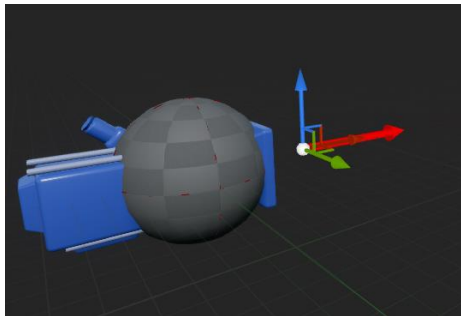


Figure 84: Widget Interaction Component



8.8.9 Further development

For further development regarding the GUI-overlays:

8.8.9.1 Reducing blueprint solutions

The blueprint solution of Unreal Engine is an easy way to program minor functionality, but there is a small drop in performance and reusability compared to the C++ solution.

The reason this project contains some blueprint solution is to test functionality and to quick fix some issues that would not have been time to fix during the time limit of the project.

There is used blueprints to make the mini map move according to the cameras' x and y directions. There is used blueprint to switch between maps as the menus are used. There is used blueprints to decide what the Widget Interaction Component will do.

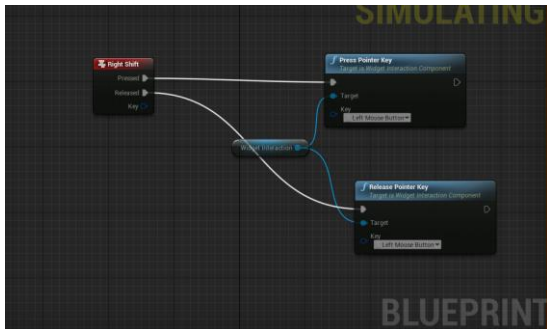


Figure 85: Blueprint for Interaction

8.8.9.2 Display constant information

The idea of a footer that constantly displays information about the vehicle, current position of the operator in the real world and time, was not manageable within the time scale of this project.

For further work, make a class that gets the information that is needed, bind this information to a widget in word space and find the best way to display this to the operator.

8.8.9.3 Add real compass data

The compass displayed in this prototype is not connected to real compass directions. It exists mainly for demonstrational purposes.

For further work, connect the rotation of the compass image to real compass directions, working in real time from the operator's perspective in the real world. So that the center of the image that displays North, will be centered as the operator looks towards north.

8.8.9.4 Make program recognize use of VR-headset

The problem regarding the operator's location and rotation data, with use of VR-headset or not (See 8.8.5.7) needs a better solution. The current solution is to enter the blueprints of WidgetManager and ArgosOperator and turn the bool "IsVR" to true, if the VR-headset is connected.

For further work, add a function that recognizes that the VR-headset is connected by itself so the developers of the future project does not have to turn two booleans on and off all the time as they are testing.



Figure 86: IsVR boolean





8.9 GPS

We inherited the old Argos project, which already had functional GPS integration. The previous projects had faced issues with GPS drift, and due to these issues, we had a requirement acquired new GPSs. During our project we integrated the GPS to work with unreal engine. This document documents how we made the new GPS, why we did it and the challenges we faced when integrating the it.

8.9.1 GPS Component	197
8.9.1.1 GPSs connection to Argos 2	197
8.9.1.2 Implementation of GPS	198
8.9.1.3 How the GPS works.....	199
8.9.1.4 Reading GPS logs	202
8.9.1.5 Challenges with programming the GPS.....	203
8.9.1.6 NMEA messages.....	204
8.9.1.7 Brief description of the GPS classes:	206



8.9.1 GPS Component

The old Argos project had a GPS but had some problems with the GPS drifting and giving very inaccurate results. Therefore we got a new requirement of buying and implementing a new GPS into the system. The new GPS uses GNSS and therefore gives us higher precision than the old one but makes it incompatible with part of the old code. This made it so that we had to write most of the functionality of the GPS from scratch while keeping a few lines of code from the old project. This is described in more depth under the implementation of the GPS.

8.9.1.1 GPSs connection to Argos 2

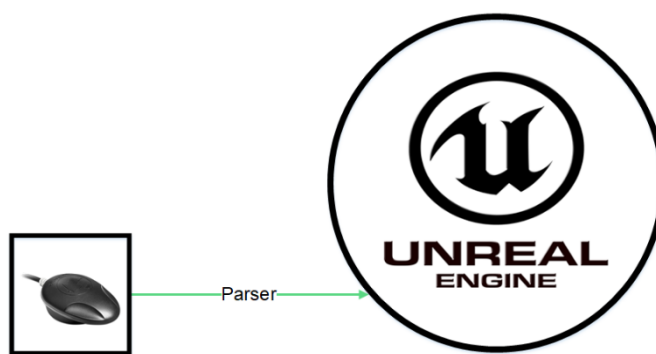


Figure 87: GPS in the system architecture

The GPS is a vital part of the system and is necessary for a lot of the functionality of the system and in fulfilling some of the requirements given to us by Kongsberg Defense and Aerospace. For requirement NF.21, the system needs a new GPS that doesn't drift for tracking the position of the system. And for finding the position of the operator in requirement F.8, The system must be able to calculate the range from an operator selected position in the image (Table 12). The GPS shall also be used to find the location of the user in the map system. For requirement C.5, the map engine shall be the Kongsberg product TerraLens (Table 14: Constraints).

The GPS main functionality is to give the system its longitude, latitude and altitude to be able to place the system on a map. And to use its location to calculate the distance and relative position to targets marked by the marking and tracking functionality of the system (see chapter 8.5 and 8.7).



8.9.1.2 Implementation of GPS

The new GPS we bought for Argos 2.0 uses GNSS, which is both GPS and GLONASS, and the old one only used GPS, so the messages sent from the new GPS were incompatible with the old parser. Messages start “GN” from the new GPS while the parser expected “GP”. Messages that start with “GN” uses GLONASS and GPS, while messages starting with “GP” only use GPS giving us higher precision with the new GPS. [50]

When starting to implement the GPS into Argos 2.0 we first tried to just put the code from the previous project directly into the Unreal Engine. This didn't work for two reasons. The code used was not written to work in Unreal Engine and the parser used by the previous projects was originally written in 2002 and does not support the message types used by our GPS. We originally intended to rewrite the original parser but found it to be overly complicated and hard to understand. We concluded that it would be easier to write our own and it would make it easier for future summer projects and bachelor projects to work with the GPS part of the system. We also looked for alternative parsers but did not find any that supported the message types we needed without needing major redesigning to work in Unreal. We also found it to be a benefit to have all the rights to the code when writing it ourselves.

We reimplemented the GlobSatBU class from the old Argos project as GPSThread where we only kept the code for serial communication and error handling with the GPS and rewrote the rest of the functionality to work with Unreals thread system and the new NMEA parser. We also kept the serial class for reading and writing data to the GPS.

The GPS uses no blueprints other than in the connection with the user interface to change the comport that the GPS is connected to, so all the functionality is written in C++ according to requirement C.2 “Programming language shall be C++” (Table 14).



8.9.1.3 How the GPS works

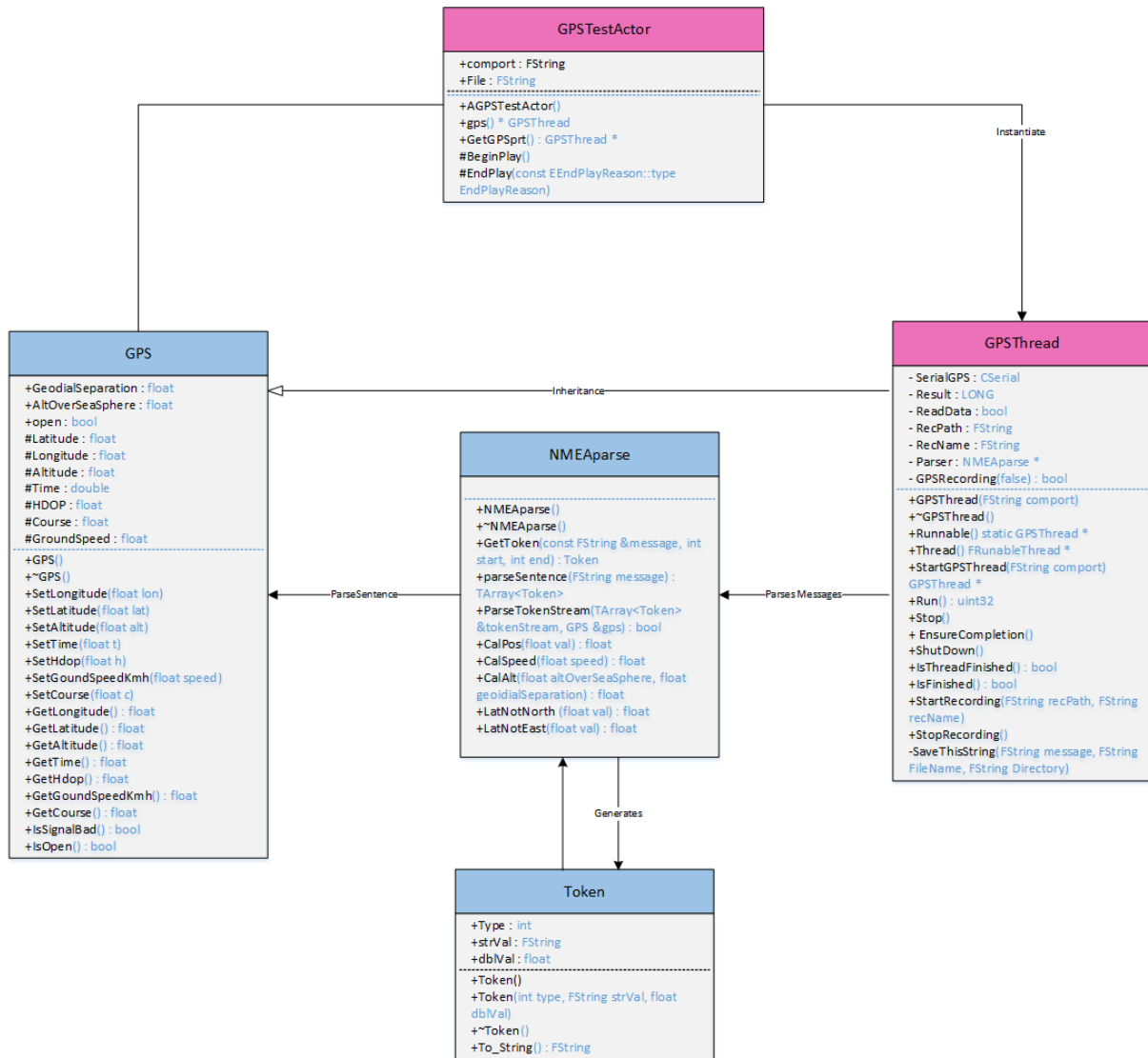


Figure 88: Class Diagram GPS

The **Serial** class first sets up serial communication with the GPS in the constructor *GPSThread* in the **GPSThread** class. If it connects, it calls a function called *run* as a thread, here messages are read from the GPS through serial communication and fills up a buffer. Messages are then extracted to a FString from this buffer. Each message is separated by a start char “\$”, and an end char “*”. We ignore the checksum after the end char. After messages are extracted from the buffer they look like this. This is also how they are stored in a text file if recording of GPS is activated.

```

$GLGSV,3,3,12,82,56,284,16,83,06,272,,88,16,088,,94,55,290,*
$GNGLL,5939.86069,N,00938.73356,E,174349.00,A,A*
$GNRMC,174350.00,A,5939.86031,N,00938.73363,E,0.154,,110518,,A*
$GNVTG,,T,,M,0.154,N,0.285,K,A*
$GNGGA,174350.00,5939.86031,N,00938.73363,E,1,09,0.89,172.1,M,40.5,M,,*
    
```



Each one of these messages is then sent to the parser by calling *parseSentence* from the **NMEAparse** class. The *parseSentence* function works by dividing the message at each comma. Each part of the message, such as UTC time, latitude and longitude are separated by a comma, so every part of the message between the commas are made into separate tokens. The start and stop chars are not made into tokens.

The **Token** class has three parameters, a type, which tells you if it's a number or a string, a string value and a float. These tokens are then put into the **TokenStream TArray**, and then parsed further by the *parseTokenStream* function from the **NMEAparse** class based on the message type. This function looks at the first token in the array to see what type of GPS message it is, "GNGGA" or "GNRMC" in the case of the example above. After identifying the message type, the data is extracted and formatted to the format we want according to how the message is built up. For each message type we pass the token values into the appropriate variables in the **GPS** class. To see how NMEA messages are built look at the tables 8.9.1.6 NMEA . The GPS receives many different messages, and just a few of them contain the information needed by our project. For that reason, our parser is only written to extract data from "GNGGA" and "GNRMC" messages from our GPS and "GPGGA" and "GPRMC" to support other GPSes. However, the *parseTokenStream* function is written to be easily expanded if any further information is needed beyond what you get from these types. Any messages not supported like "GNGLL" are ignored by the parser.

To contain the GPS information, we have an abstract class **GPS** which the classes for connecting to the GPS and for reading recorded GPS data will inherit from. The **GPS** class has a set of Get functions and is used as the interface to any parts of the program that require GPS information.

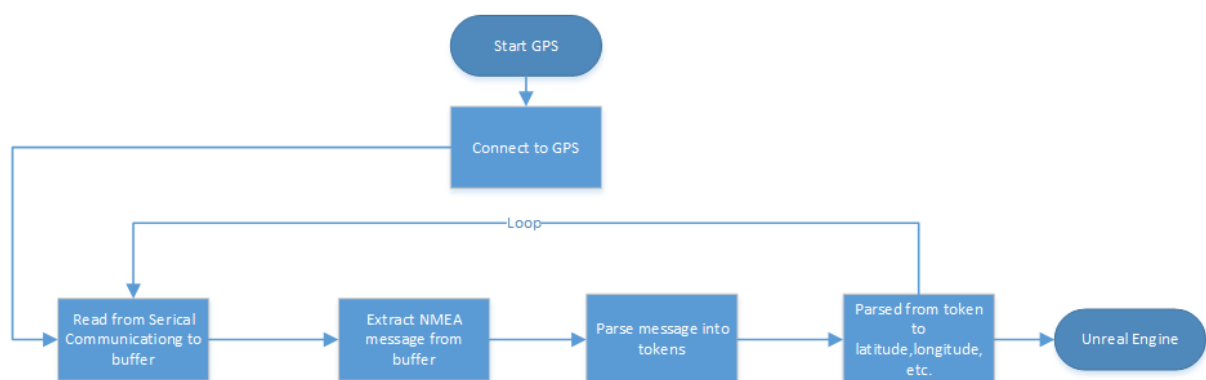


Figure 89: Show the dataflow for the GPS data



The function *run* that is responsible for constantly reading the GPS data runs on a separate thread to stop the entire system from having to wait for the GPS to send new data. Something it does approximately once every second. The loop processing the GPS messages is always running as long as the `readData` boolean is true.

Currently the system can only connect to one GPS. If there is a need for higher precision or redundancy the system could be expanded for that. Currently you also have to manually set the comport for the GPS, in the future the system could be set up to automatically detects the comport making it more user friendly.



8.9.1.4 Reading GPS logs

The GPSReadThread inherits from FRunnable and GPSFileReader. GPSFileReader inherits from GPS and overrides the set and get functions in GPS, in order to display the recorded information at the correct time.

The GPSReadThread is used the same way as the GPSThread, except it takes a file location instead of a comport.

The GPS reader stores the time of the first received GPS message and the time it was read and uses that information to read the subsequent messages at the correct time, in order to synchronize this with other information logs. This means that it is incompatible with any NMEA messages that do not include a timestamp.

If the GPS reader encounters a message it does not support, it will keep reading until it reads a valid message.

The thread stops running when it has read all messages in a file, or Stop() is called.

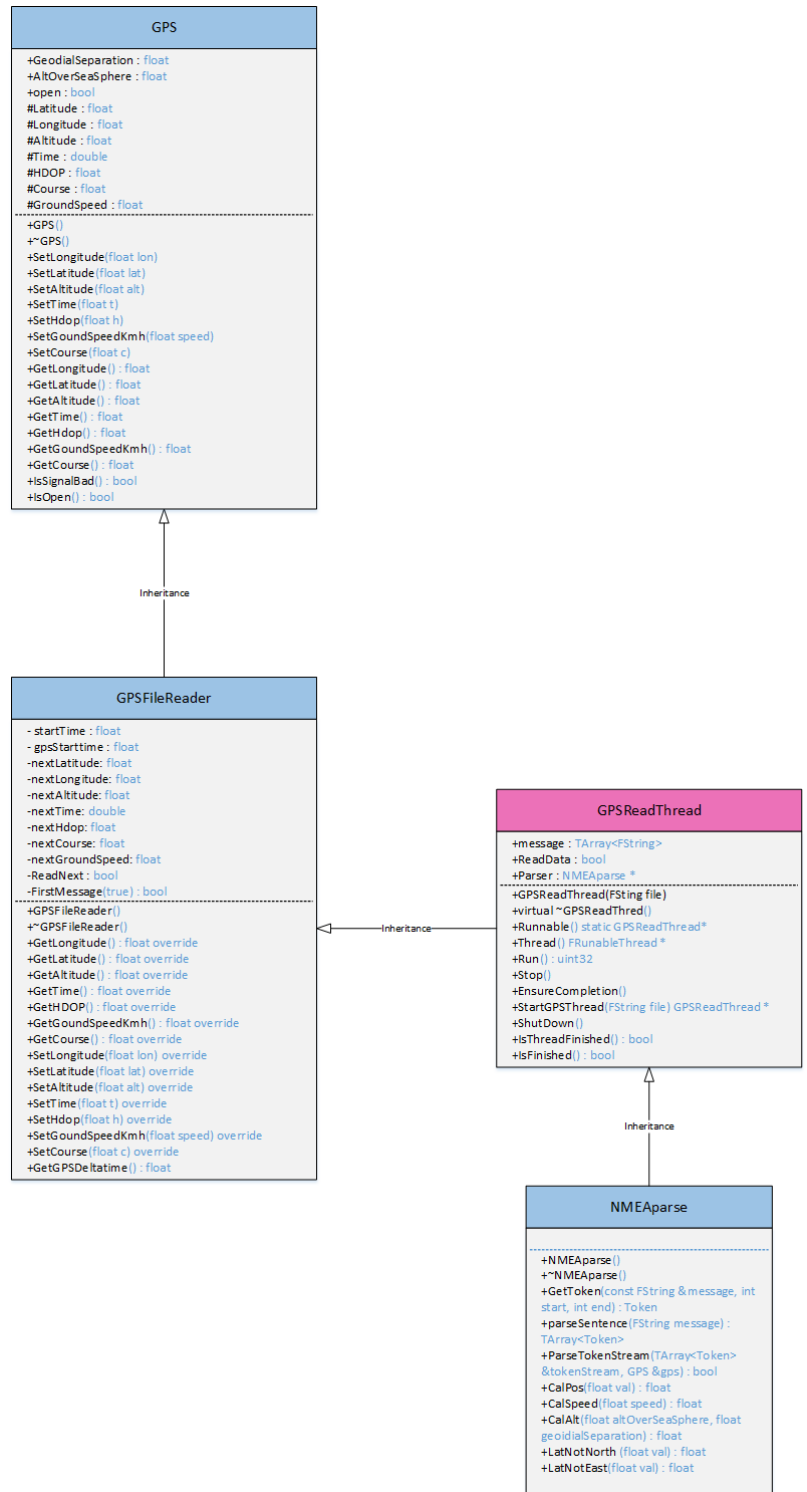


Figure 90: Class diagram for read GPS logs

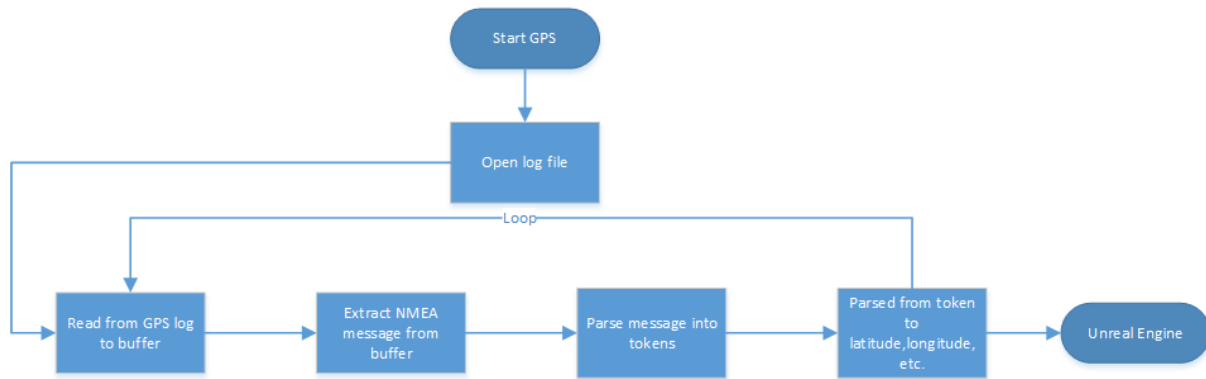


Figure 91: Dataflow while reading from a GPS log

8.9.1.5 Challenges with programming the GPS

Throughout the development of the GPS functionality we faced a few challenges:

- Simply just getting to know how Unreal Engine works and the ways you have to program differently to regular c++ programming in visual studio.
- Not being allowed to use the std
- Conflicts between Unreals source code and other libraries, especially the windows header
- The lack of documentation on certain areas of unreal.
- Making the code unreal friendly with the lack of documentation
- Understanding the thread system that unreal uses
- General errors and bugs in the code while writing it

At the beginning of the project we made a lot of errors, due to us not understanding how unreal changes the way C++ is used. This challenge was solved by us gaining an understanding over the engine.

C++ in Unreal is parsed by the engine, and macros that tell the engine how to parse the source code is frequently used. We were unused to these features and struggled to use them at first

We were unused to Unreals type system, but by referring to the documentation we found the equivalent types and functions.

The windows header is extremely large and caused conflicts with the Unreal Engine source code. It took a lot of time to identify the cause of this, and how to solve it. We had to find the correct Unreal header files to include and, in some cases, go through the windows header to find what it had redefined the functions as.



In documentation for parts of the engine used frequently for game development the documentation was good and easily accessible. We needed functionality that was not frequently used, and this functionality usually had lacking and, in some cases, wrong information in the documentation. Several times we found ourselves digging into the source code in Unreal in order to identify what a function did and how it was used.

During the development of the parser we made several small errors in the code, usually due to errors in a loop, or not checking the length of an array before accessing array elements. These errors were found and solved during development and debugging.

One of the errors we made was assuming that one NMEA message would be sent every time the GPS sent us data. We found out that the GPS would fill its buffer before sending, meaning that NMEA messages were split up between several transmissions. The messages that were split up were ignored by the parser. When we understood that this was an issue we rewrote the GPSThread class to send messages compiled from several transmissions into separate messages, which solved the issue.

We also had issues with the standard library(std), since Unreal provides its own type system instead. The greatest difficulty was converting the std threads to Unreals FRunnable thread system, since they are used differently than the std threads. First, we tried to not use threads at all, and rather read a message every Tick, or frame the unreal engine rendered. However, this led to the entire execution of the program to be halted, while we waited to receive a message.

8.9.1.6 NMEA messages

There are no types of GPS messages that contain all the information we want in a single message. Therefore we need to extract the data from two different GPS packets, GNGGA and GNRMC. Both contain the time, latitude and longitude while only GNGGA contains the altitude and only GNRMC contains the course and speed. Underneath are an example of a GNGGA message and a GNRMC message with a description of what the different part does. Each part is separated by a comma. Messages that start with "GN" uses GLONASS and GPS, while messages starting with "GP" only use GPS giving us higher precision with the new GPS. [50]



Message:	\$GNGGA,174350.00,5939.86031,N,00938.73363,E,1,09,0.89,172.1,M,40.5,M,,*	
Field No.	Value	Description
0	\$GNGGA	Message type
1	174350.00	UTC time
2	5939.86031	Latitude (Degrees, minutes)
3	N	North or South (N/S)
4	00938.73363	Longitude (Degrees, minutes)
5	E	East or West (E/W)
6	1	Quality indicator for position fix
7	09	Number of satellites used
8	0.89	HDOP – Horizontal Dilution of Precision
8	172.1	Altitude above mean sea level
9	M	Units of measurement (Meters)
10	40,5	Difference between an ellipsoid representation of earth and mean sea level
11	M	Units of measurement (Meters)

Table 83: Description for GNGGA message [50]

Message:	\$GNRMC,174350.00,A,5939.86031,N,00938.73363,E,0.154,,110518,,A*	
Field No.	Value	Description
0	\$GNRMC	Message type
1	174350.00	UTC time
2	A	A = valid data, V invalid data
3	5939.86031	Latitude (Degrees, minutes)
4	N	North or South (N/S)
5	00938.73363	Longitude (Degrees, minutes)
6	E	East or West (E/W)
7	0.154	Groundspeed (Knots)
8	0	Course (0-360, shows the direction the GPS is traveling)
9	110518	Date (DDMMYY)

Table 84: Description of GNRMC message [50]



8.9.1.7 Brief description of the GPS classes:

Underneath is a list with descriptions of the classes used for anything related to the GPS in the Argos 2.0. For a more in depth look at how the classes and functions within them work, look at the Doxygen documentation.

- GPS – Stores the information about latitude, longitude, altitude, etc. As Well as the get functions to use the data in other parts of the system.
- GPSFileReader – Overloads GPS class to synchronize GPS data with timestamp of recorded messages.
- GPSReadThread – Overloads GPS file reader to run as a thread and open and read from GPS logs.
- GPSThread – Thread that connect to GPS and receives data, all GPS data will also be recorded here if recording is active.
- NMEAParse – Parses messages from GPS into location data.
- Serial – (Class we have not written ourselves) used for reading and writing serial communication.
- Token – Class used to split NMEA message into separate data stored as the correct data types.
- GPSActor – Placed into the virtual world to connect the GPS with the maps.



8.10 Software implementation diagram

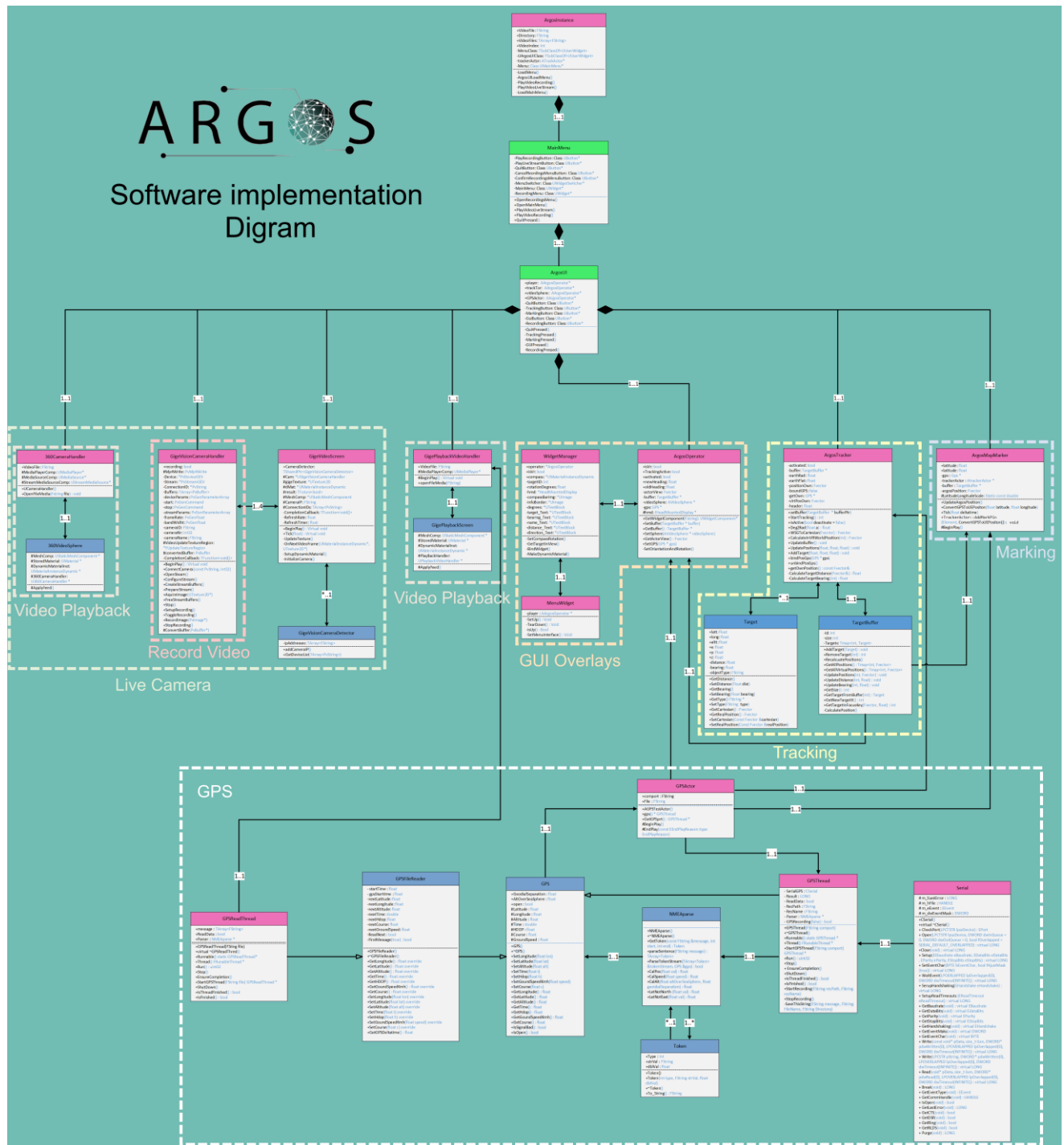


Figure 92: Software implementation diagram. Full image is added as attachment





9 Conclusion

This chapter describes the conclusion of the project and the work that has been done to accomplish the Argos 2.0 functionality. It also describes related projects and suggestions for further development.

Describes

- Contribution
- Related projects
- Further work



9.1 Overview

In this bachelor project the bachelor group recreated the existing Argos system to take advantage of cutting-edge technology, and to be extensible to future technology in both the VR and camera fields.

Using a powerful framework such as Unreal has saved this project a lot of time that would otherwise be spent implementing features already existing within Unreal Engine. The previous projects used a self-developed engine to run Argos. This engine suffered from being difficult to extend with new features. For example, UI elements were troublesome to develop, since they had to be programmed from scratch. Unreal Engine on the other hand has a powerful and flexible UI system that significantly shortens development time of new UI elements.

Unreal Engine also supports the use of plugins which extend the functionality of the engine. Integrating the Decklink capture card was a challenging task that would have taken a long time, but there already existed a plugin for this which let us spend more time on other parts of the system.

Plugins were also an immense help in integrating VR headsets in our application. To make both the OSVR headset and the HTC Vive Pro to function in our system all that was required was to install the required software and activate their respective plugins. This let us focus on making Argos work well in VR.

Our project has created a new version of Argos, that is more extensible than the previous system, allowing future project groups to focus on new features. The project has also brought new technology to Argos, using a 360-degree camera to improve the users field of view seamlessly.



9.2 Contribution

This project group has made great contributions to the Argos project. Most importantly is making the system more extensible, thereby allowing future project groups to achieve more progress.

The group has also accumulated research on using dedicated 360-degree cameras to improve the users field of view. The system is now compatible with most consumer VR headsets and can use any camera able to stream video over HDMI. Marking and Tracking software has been added to allow the Argos operator to have detailed information about his or hers surrounding, as well as targets position and information at any given time.

Novelty

Currently there is no system known to the group using VR and 360-degree cameras to improve the situational awareness of the driver of a vehicle. Argos is breaking new ground in the mixed reality field.

9.3 Related Work

In high end consumer vehicles today 360-degree cameras are used to improve the driver's situational awareness. However, unlike Argos, these systems are designed to be used at low speed to avoid obstacles close to the vehicle when parking. These systems usually present a top down view of the vehicle to the driver using a traditional screen and can be seen as an evolution of back up cameras.

At GTC 2018 NVIDIA presented a system using VR to remote control a vehicle. The driver was equipped with a VR headset and a gaming wheel and pedals controller. Several cameras were mounted to the vehicle to give the user a 360-degree view outside the vehicle. The interior of the car was reconstructed virtually to let the user see their own arms in VR in relation to the controls. [51]



9.4 Future Work

The NVIDIA VRWorks SDK has support for real time stitching of 360° videos [52]. This SDK could improve the seams between the images received from the GigE Vision cameras. NVIDIA offers custom Unreal Engine branches with VRWorks integrated. [53]

The project has suffered from performance issues using the eBus converter to convert GigE Vision images from BayerG8 to RGBA and alternatives to this converter should be researched. NVIDIA CUDA offers images conversion using the GPU, which could improve the performance of Argos and lower the latency.

Currently the only sensors used by the system is a GPS, future project should research and integrate other sensors, to improve the situational awareness of the user.

The latency from the GigE Vision cameras is higher than would be preferred. Some latency can be eliminated by not using a switch. Currently a switch is needed to provide power over ethernet(POE) to the cameras. By switching to a new network card, or by powering the cameras by the Hirose I/O port the latency should be reduced.

Using computer vision to analyse incoming images and marking regions of interest could improve the user's situational awareness, especially since the user is not able to look in all directions at once.

Currently Argos uses OpenStreetMap as its map system. The preferred map system is TerraLens, but this was not integrated due to issues with OpenGL and Unreal Engine. Future project groups should research how to implement TerraLens with Argos.





10 Sources

- [1] K. D. & Aerospace, "Argos Requirement Specification," Kongsber Defence & Aerospace, Kongsberg, 2018.
- [2] Microsoft, "Microsoft Support," Microsoft, 2018. [Online]. Available: <https://support.microsoft.com/en-us/help/13853/windows-lifecycle-fact-sheet>. [Accessed 16 01 2018].
- [3] OSVR, «OSVR,» OSVR, [Internett]. Available: <http://www.osvr.org/>. [Funnet 15 May 2018].
- [4] Argos Bachelor Group, "Argos Web-page," Argos Bachelor Groups, 2018. [Online]. Available: <http://projectargos.net/>. [Accessed 18 1 2018].
- [5] A. Srivastava, S. Bhardwaj and S. Saraswat, "SCRUM model for agile methodology," 21 12 2017. [Online]. Available: <http://ieeexplore.ieee.org/document/8229928/>. [Accessed 1 2018].
- [6] S. T. Veethil, "Scrum Alliance," 3 5 2013. [Online]. Available: <https://www.scrumalliance.org/community/articles/2013/2013-may/risk-management-in-agile>. [Accessed 25 1 2018].
- [7] Agile Alliance, "Agile Alliance," 2018. [Online]. Available: <https://www.agilealliance.org/glossary/invest/>. [Accessed 1 2018].
- [8] Agile Alliance, "Agile Alliance - The Three C's," 2018. [Online]. Available: <https://www.agilealliance.org/glossary/three-cs/>.. [Accessed 1 2018].
- [9] H. Zahraoui and M. A. J. Idrissi, "Adjusting story points calculation in scrum effort & time estimation," 17 12 2015. [Online]. Available: <http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7358400>. [Accessed 1 2018].
- [10] «Innolution, Task,» 2018. [Internett]. Available: <http://www.innolution.com/resources/glossary/task..> [Funnet 04 02 2018].
- [11] Scrum.org, "Scrum.org," 2018. [Online]. Available: <https://www.scrum.org/resources/what-is-a-sprint-review..> [Accessed 1 2018].
- [12] Software Testing Fundamental, "Software Testing Fundamental," 2018. [Online]. Available: <http://softwaretestingfundamentals.com/integration-testing/>. [Accessed 1 2018].



- [13] «Scrum Alliance, "Done",» 2018. [Internet]. Available: <https://www.scrumalliance.org/community/articles/2008/september/what-is-definition-of->. [Funnet 04 02 2018].
- [14] Office of the Deputy Assistant Secretary of Defense for Systems Engineering,, "Office of the Deputy Assistant Secretary of Defense for Systems Engineering,," 1 2017. [Online]. Available: <https://www.acq.osd.mil/se/docs/2017-RIO.pdf>. [Accessed 19 1 2018].
- [15] Epic Games Inc. , «Unreal Engine, About,» 2018. [Internet]. Available: <https://www.unrealengine.com/en-US/what-is-unreal-engine-4>. [Funnet 20 05 2018].
- [16] Epic Games Inc., «Unreal Engine, UPROPERTY,» 2018. [Internet]. Available: <https://wiki.unrealengine.com/UPROPERTY> . [Funnet 16 05 2018].
- [17] Epic Games Inc, "Unreal Engine - Blueprints," Epic Games Inc, [Online]. Available: <https://docs.unrealengine.com/latest/INT/Engine/Blueprints/>. [Accessed 8 3 2018].
- [18] Epic Games Inc., «Unreal Engine, Levels,» 2018. [Internet]. Available: <https://docs.unrealengine.com/en-us/Engine/Levels>. [Funnet 16 05 2018].
- [19] Blackmagic Design, «DeckLink Studio 4K Technical Specifications,» Blackmagic design, [Internet]. Available: <https://www.blackmagicdesign.com/products/decklink/techspecs/W-DLK-12>. [Funnet 14 May 2018].
- [20] Elgato, «Elgato Game Capture 4K60 Pro — Supported Resolutions and Frame Rates,» Elgato, 27 December 2017. [Internet]. Available: <https://gaming.help.elgato.com/customer/en/portal/articles/2897373-elgato-game-capture-4k60-pro—supported-resolutions-and-frame-rates>. [Funnet 14 May 2018].
- [21] Foto.no, «Blackmagic Decklink Studio 4K,» Foto.no, [Internet]. Available: https://butikk.foto.no/blackmagic/110692/blackmagic-decklink-studio-4k-pcie-opptaker-og-monitorering?gclid=CjwKCAjwiurXBRAnEiwAk2GFZqYFoOpuN-HRFkllm_aphK0CmO8v5T3yGsWL3Nw38cklR-CYjes9jxoC1_UQAvD_BwE. [Funnet 15 May 2018].
- [22] Komplet.no, «Elgato 4K60 Pro,» Komplet.no, [Internet]. Available: <https://www.komplet.no/product/971571/gaming/streaming/capture-streaming/elgato-game-capture-4k-60-pro?offerId=KOMPLETT-310-971571#>. [Funnet 15 May 2018].
- [23] Elgato, «Elgato Game Capture 4K60 Pro — Technical Specifications,» Elgato, 27 December 2017. [Internet]. Available: <https://gaming.help.elgato.com/customer/en/portal/articles/2896946-elgato-game-capture-4k60-pro—technical-specifications>. [Funnet 14 May 2018].
- [24] Audioholics, «Understanding the Different HDMI Versions (1.0 to 2.0),» Audioholics, 11 September 2013. [Internet]. Available: <https://www.audioholics.com/hdtv-formats/understanding-difference-hdmi-versions>. [Funnet 14 May 2018].
- [25] The Mill, «themill.com,» The Mill, 14 May 2018. [Internet]. Available: <https://github.com/themill/DeckLinkMedia>. [Funnet 14 May 2018].



- [26] AltaVision, «Mako G-223,» [Internet]. Available: http://www.altavision.com.br/Arquivos/AVT/Datasheet/Mako_G-223_Datasheet.pdf. [Funnet 18 05 2018].
- [27] DALSA, «Gige Vision for Realtime,» 2010. [Internet]. Available: https://nstx.pppl.gov/nstxhome/DragNDrop/Operations/Diagnostics_&_Support_Sys/D1CCD/GigE_Vision_for_Realtime_MV_11052010.pdf. [Funnet 18 05 2018].
- [28] STEMMER IMAGING Ltd. , «Gige-Vision,» [Internet]. Available: <https://www.stemmer-imaging.co.uk/en/knowledge-base/gige-vision/>. [Funnet 18 05 2018].
- [29] STEMMER IMAGING Ltd. , «GVSP,» [Internet]. Available: <https://www.stemmer-imaging.co.uk/en/knowledge-base/gige-vision-gvsp-gige-vision-streaming-protocol/>. [Funnet 18 05 2018].
- [30] STEMMER IMAGING Ltd. , «GVCP,» [Internet]. Available: <https://www.stemmer-imaging.co.uk/en/knowledge-base/gige-vision-gvcp-gige-vision-control-protocol/>. [Funnet 18 05 2018].
- [31] Cisco, «Cisco.com,» 18 05 2018. [Internet]. Available: <https://www.cisco.com/c/en/us/solutions/small-business/resource-center/connect-employees-offices/network-switch-what.html>. [Funnet 18 05 2018].
- [32] Juniper, «Juniper,» 14 03 2018. [Internet]. Available: https://www.juniper.net/documentation/en_US/junos/topics/concept/layer-2-lacp-security-understanding.html. [Funnet 18 05 2018].
- [33] J. Becvar, «<http://www.leutron.com>,» 2009. [Internet]. Available: https://www.next-community.de/fileadmin/media/whitepaper/files/Leutron_wp_network-performance.pdf. [Funnet 18 05 2018].
- [34] cppreference, «cppreference,» 13 11 2017. [Internet]. Available: <http://en.cppreference.com/w/cpp/thread/async>. [Funnet 18 05 2018].
- [35] EPIC GAMES, «Unreal Engine Async Api,» 2017. [Internet]. Available: <http://api.unrealengine.com/INT/API/Runtime/Core/Async/index.html>. [Funnet 18 05 2018].
- [36] EPIC GAMES, «Unreal Engine FUpdateTextureRegion2D Api,» 2017. [Internet]. Available: <http://api.unrealengine.com/INT/API/Runtime/RHI/FUpdateTextureRegion2D/index.html>. [Funnet 18 05 2018].
- [37] EPIC GAMES, «Unreal Engine UTexture2D Api,» 2017. [Internet]. Available: <http://api.unrealengine.com/INT/API/Runtime/Engine/Engine/UTexture2D/index.html>. [Funnet 18 05 2018].
- [38] EPIC GAMES, «Unreal Engine Media Texture Api,» 2017. [Internet]. Available: <http://api.unrealengine.com/INT/API/Runtime/MediaAssets/UMediaTexture/index.html>. [Funnet 18 05 2018].
- [39] EPIC GAMES, «Unreal Engine Textures,» 2018. [Internet]. Available: <https://docs.unrealengine.com/en-us/Engine/Content/Types/Textures>. [Funnet 18 05 2018].



- [40] EPIC GAMES, «Unreal Engine Materials,» 2018. [Internet]. Available: <https://docs.unrealengine.com/en-us/Engine/Rendering/Materials>. [Funnet 18 05 2018].
- [41] EPIC GAMES, «Unreal Engine Dynamic Material Instances,» 2018. [Internet]. Available: <https://docs.unrealengine.com/en-us/Engine/Rendering/Materials/MaterialInstances>. [Funnet 18 05 2018].
- [42] Riftinfo, «Oculus Rift specs,» [Internet]. Available: <https://riftinfo.com/oculus-rift-specs-dk1-vs-dk2-comparison>. [Funnet 16 May 2018].
- [43] OSVR, «OSVR HDK2,» OSVR, [Internet]. Available: <http://www.osvr.org/hdk2.html>. [Funnet 16 May 2018].
- [44] HTC, «HTC Vive Pro,» [Internet]. Available: <https://www.vive.com/us/product/vive-pro/>. [Funnet 16 May 2018].
- [45] C. Shu, «Improving Algorithm to Compute Geodetic Coordinates,» i *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing*, Shanghai, China , 2008.
- [46] C. Veness, «www.movable-type.co.uk,» Chris Veness, 2017. [Internet]. Available: <https://www.movable-type.co.uk/scripts/latlong.html>. [Funnet 05 04 2018].
- [47] Epic Games Inc, «Unreal Engine - UMG,» Epic Games Inc, 2004-2017. [Internet]. Available: <https://docs.unrealengine.com/latest/INT/Engine/UMG/>. [Funnet 08 03 2018].
- [48] ODbL, «Open Street Map,» [Internet]. Available: <https://www.openstreetmap.org/export#map=14/59.6648/9.6808>. [Funnet 16 05 2018].
- [49] MikeFricker, «GitHub UE4 plugins Street Map,» 01 01 2017. [Internet]. Available: <https://github.com/ue4plugins/StreetMap/blob/master/LICENSE.txt>. [Funnet 16 05 2018].
- [50] U-Blox, «<https://www.u-blox.com>,» 06 03 2018. [Internet]. Available: https://www.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_%28UBX-13003221%29_Public.pdf. [Funnet 14 5 2018].
- [51] J. Horwitz, «Venture Beat,» 27 03 2018. [Internet]. Available: <https://developer.nvidia.com/nvidia-vrworks-and-ue4>. [Funnet 18 05 2018].
- [52] NVIDIA Corporation, «NVIDIA VRWorks,» 2018. [Internet]. Available: <https://developer.nvidia.com/vrworks/vrworks-360video>. [Funnet 18 05 2018].
- [53] NVIDIA Corporations, «NVIDIA VRWorks Unreal,» 2018. [Internet]. Available: <https://developer.nvidia.com/nvidia-vrworks-and-ue4>. [Funnet 18 05 2018].
- [54] KDA, "Kongsberg Defence & Aerospace," 2018. [Online]. Available: <http://www.kongsberg.com/en/kds>. [Accessed 16 01 2018].
- [55] IEEE, "Open Source Reality," IEEE Computer Society, 2015. [Online]. Available: <https://www.computer.org/csdl/proceedings/vr/2015/1727/00/07223456.pdf>. [Accessed 18 01 2018].



- [56] J. M. Bass, "Scrum Master Activities: Process Tailoring in Large Enterprise Projects," 2 10 2014. [Online]. Available: <http://ieeexplore.ieee.org/document/6915249/>. [Accessed 01 2018].
- [57] N. Davis, "Driving Quality Improvement and Reducing Technical Debt with the Definition of Done," 30 9 2013. [Online]. Available: <http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6612893>. [Accessed 1 2018].
- [58] J. Rasmusson, "Scrum Methodology Image," Agile Nutshell, [Online]. Available: <http://www.agilenutshell.com/assets/methods/scrum/scrum-overview.png>. [Accessed 1 2018].
- [59] Epic Games Inc, "Unreal Engine - UMG," Epic Games Inc, 2004-2017. [Online]. Available: <https://docs.unrealengine.com/latest/INT/Engine/UMG/>. [Accessed 08 03 2018].
- [60] Epic Games inc, «Building Unreal Engine from Source,» 2018. [Internet]. Available: <https://docs.unrealengine.com/en-US/Programming/Development/BuildingUnrealEngine>. [Funnet 12 03 2018].
- [61] Epic Games Inc, «Media Framework Technical Reference.,» 2018. [Internet]. Available: <https://docs.unrealengine.com/en-US/Engine/MediaFramework/TechReference>. [Funnet 12 03 2018].
- [62] Epic Games Inc, «Plugins,» 2018. [Internet]. Available: <https://docs.unrealengine.com/en-us/Programming/Plugins>. [Funnet 12 03 2018].
- [63] J. Frank, «Black / Blank Display in the HMD,» 22 04 2017. [Internet]. Available: <http://52.27.214.34:8090/pages/viewpage.action?pageId=11207862>. [Funnet 12 03 2018].
- [64] F. Jacob, «Release Notes,» 25 04 2017. [Internet]. Available: <http://wiki.osvr.org/display/DD/Release+Notes>. [Funnet 12 03 2018].
- [65] Microsoft, «Microsoft/VisualStudioUninstaller.,» 08 03 2017. [Internet]. Available: <https://github.com/Microsoft/VisualStudioUninstaller>. [Funnet 12 03 2018].
- [66] Video Interface Download, «Fixed in Release 5.0.0,» 20 02 2018. [Internet]. Available: <https://supportcenter.pleora.com/s/article/eBUS-SDK-5-0->. [Funnet 12 03 2018].
- [67] Epic Games, «Properties,» [Internet]. Available: <https://docs.unrealengine.com/en-us/Programming/UnrealArchitecture/Reference/Properties>. [Funnet 18 May 2018].
- [68] Epic Games, «Ufuntions,» [Internet]. Available: <https://docs.unrealengine.com/en-us/Programming/UnrealArchitecture/Reference/Functions>. [Funnet 18 May 2018].





Appendix – A: User Manual

This appendix contains the user manual. The user manual describes how a user may use the Argos 2.0 system. How to start, stop and navigate through the solution and program.

Bachelor project 2018 at
Faculty of Technology, Natural Sciences and Maritime Sciences
University of Southeast Norway
Project owner:
Kongsberg Defence Systems



KONGSBERG



1 How to start the application

We have had issues packaging the application into an executable, this will be fixed before the presentation. Therefore, the application has to be started through the Unreal Editor. The project is opened in the editor by double clicking the .uproject file. The project can also be started in visual studio by running debug mode. Any required software is started automatically when starting the project. If the OSVR and SteamVR software is not installed the application can only be used with a normal screen.

Open the main menu level, then press play in order to test the application.

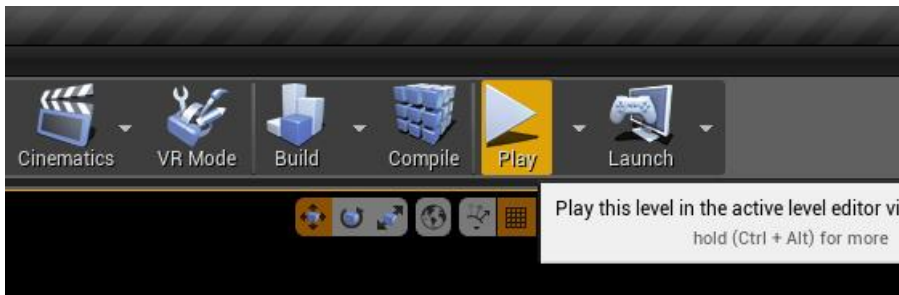


Figure 93: Play the application

2 How to control the application

Normal mode:

Use the computer mouse cursor to look around.

VR mode:

Look around with the VR headset on.

The red dot in front of you represents the mouse cursor while "Right Shift" on your keyboards represents the mouse click.

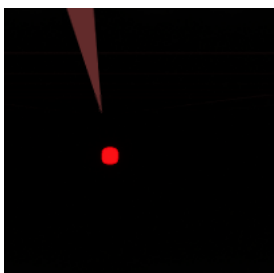


Figure 94: Red dot indicates mouse cursor



Figure 95: "Shift" functions as a mouse click

3 How to use the main menu

As the program runs the main menu appears. Here there is a choice to view live stream, recorded video or quit.



Figure 96: The Main Menu

If “Play Livestream” is pressed, there is the options of viewing through the 360° camera or the GiGe cameras, press the desired button. Now you are watching the live stream.

If “Play Recording” is pressed, there is the same options, to view video recorded by the 360° camera or the Gige cameras.

The next step, choosing which video that will be shown. Press “Next” or “Previous” to toggle throught the recorded videos. As a video is chosen and “Choose” is pressed, the map dependent to the earlier choices will be opened, playing the chosen video.



4 How the User Interface works in runtime

As you are playing a recording or watching the livestream you have the choices of toggling different functions from the Argos UI menu. You activate this menu by pressing “M” on your keyboard.



Figure 97: Press “M” on the keyboard to activate ArgosUI Menu

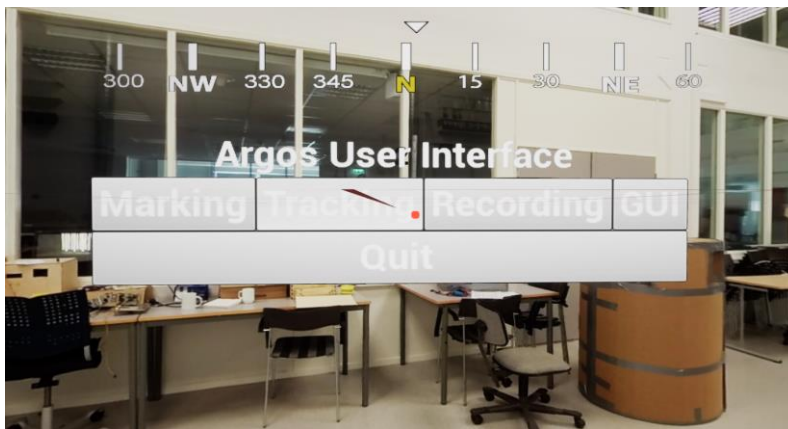


Figure 98: Argos UI Menu will appear like this

If you press “Marking” the marked targets will be marked in the mini map. The mini map is located down in the view, you will see it if you look towards the floor.

The “Tracking” button will show the targets that exists around you. They will show up like blue squares.



Figure 99: Targets tracked as blue squares

If you look at these squares an information box will pop up, displaying the information of the target you are looking at.

If you press the “Recording” button the cameras and GPS will start recording and storing the data.

The “GUI” button toggles the visibility of the GUI elements on screen.

The “Quit” button sends you back to the “Main Menu”.

5 How to exit the application

To exit the application, go to the “Main Menu” and press “Quit”.

6 Navigating the solution

The source code written by this project group can be found in the source subfolder of the project folder. Header files are found in the public subfolder, cpp files are found in the private subfolder. The visual studio solution file is found in the project folder.

6.1 Actors and Pawns

The top level of the functionality of the system is found in actors and pawns. In Unreal Engine actors are object that can be placed in a level, pawns are an extension of actors. We have several levels with the appropriate actors to provide the functionality for that level.

360VideoSphere.h

The video sphere actor has a spherical mesh attached, used to display 360-degree video. It has a camera handler responsible for opening the media source to be displayed.

TrackerActor.h

The tracker actor is responsible for the marking and tracking system. It has a TargetBuffer to contain the targets, and an ArgosTracker to perform operations on the buffer. It also fetches GPS information to update the users position.

GPSThread.h

The GPSThread is responsible for reading from a gps or a gps log file, and for recording gps information. It starts a GPSThread to read from a GPS, and a GPSReadThread to read from a log file.

MiniMapActor.h

The mini map actor has a widget component attached in order to display the map to the user.



ArgosMapMarker.h

This is an actor component. It is attached to an actor in the level. It is responsible for fetching data from the gps and moving the user on the map accordingly.

MarkPin.h

The markpin actor is placed on the minimap to mark locations.

ArgosOperator.h

The ArgosOperator is the representation of the user in the system. The ArgosOperator has a camera component. The camera is what the user of the system sees through when using the system. The ArgosOperator rotates the camera according to the rotation of the VR headset, and the heading received from the gps.

6.2 Widgets

The menuesystem is implemented using Unreal Motion Graphics. This is the recommended system for making user interfaces in Unreal. The widgets are designed visually in the Unreal editor, the behaviour of the widgets is programmed in C++.

The source code for the UI is found in the MenuSystem subfolder.

6.3 Instance and Mode

The ArgosInstance is a manager object for the system. It is created at startup and exists until shutdown. It is used to store data between levels, such as the which recording was selected in the main menu.

The ArgosMode is used for startup information of the system.

6.4 GigE Vision

The Ebus SDK is used to communicate with the GigE cameras. The library is included using the Argos2_0.Build.cs file. The GigE Vision source code is found in the GigE Vision subfolder.

GigEVisionCameraDetector.h

Finds the GigE Vision cameras.

GigEVisionCameraHandler.h

Receives images from the cameras and sends them to the screen.

GigEPlaybackScreen.h

Mesh used to display streamed and recorded video

GigEPlaybackVideoHandler.h

Starts playback of GigE Vision recordings.

GigEVideoScreen.h

Starts livestreaming from GigE Vision cameras.





Appendix – B: Project Plan

This appendix contains the project plan for this bachelor project. What the initial plan was and how the plan has been followed through the course of the project.

Bachelor project 2018 at
Faculty of Technology, Natural Sciences and Maritime Sciences
University of Southeast Norway
Project owner:
Kongsberg Defence Systems



KONGSBERG



1 Overview

Project plans is usually mapped out with the waterfall-approach, which contradicts the Scrum methodology. We would like to point out that the project plan was requested by KDA to get an overview of the project layout. To ensure we follow Scrum properly, we use the project plan to map out a timeline for which certain user stories should be finalized, but we follow the Scrum methodology listed in the Project model chapter.

This chapter presents and explains the initial plan and execution of Project Argos. It aims to give the reader an understanding of the task at hand, the planned solutions and how this was to be achieved in the given period. It also provides a comparison of estimation versus reality of phases and tasks.

2 Initial project plan description

Project Argos uses the SCRUM project model to reach the final goal. However, we found it important to section the initial plan into 5 phases. *Initiation, Foundation, Extensions, Delivery and Presentation*. The reason we do this is to ensure that our sprints are focused on the right tasks at the right time. Clear milestones in the overall project plan could also help the scrum master plan sprints based on the project deadline.



2.1 Initial Project Schedule

Figure 100: Project schedule diagram explains the initial project schedule provided to Alex from KDA.

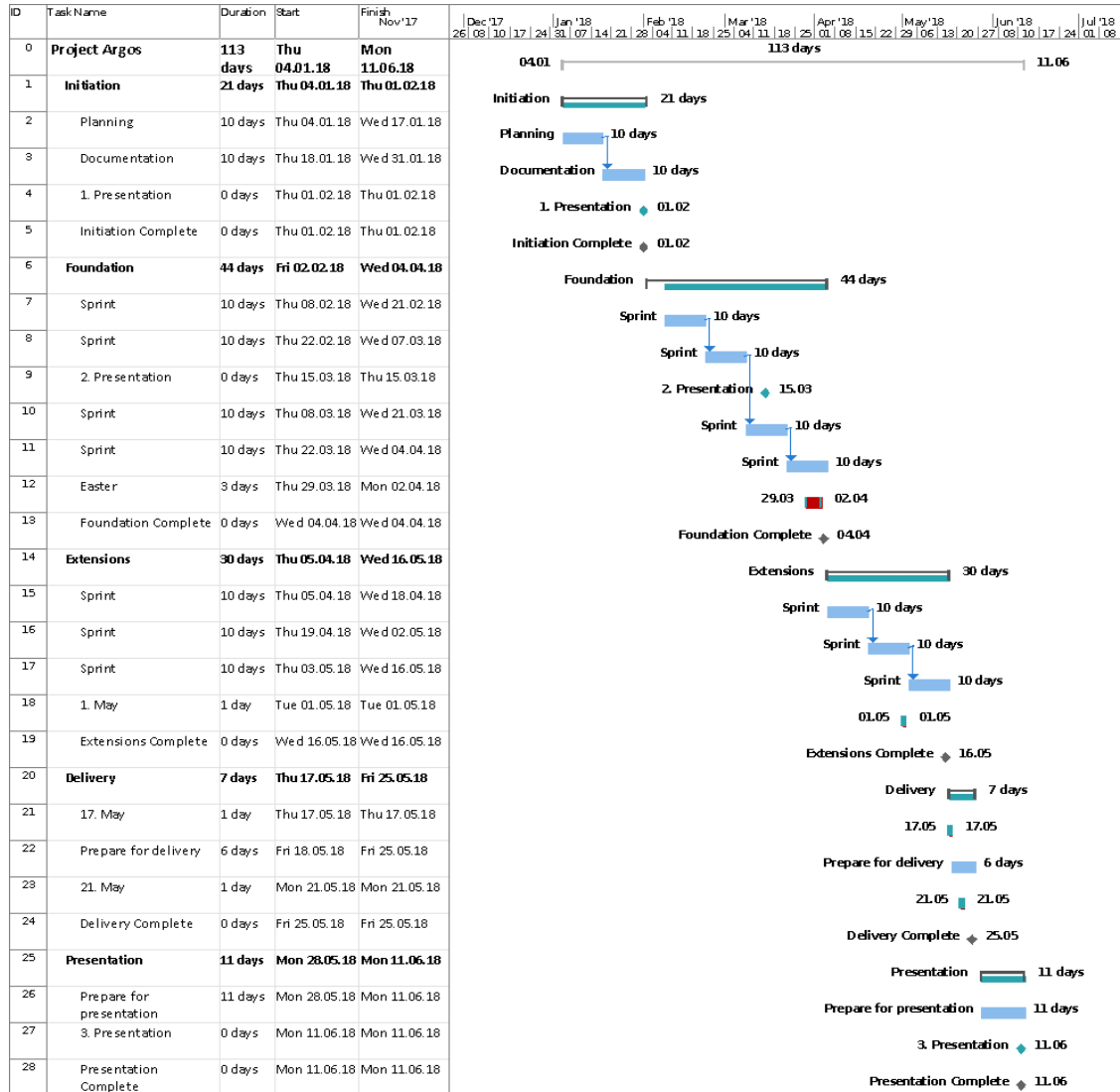


Figure 100: Project schedule diagram



Initiation

In this phase the focus is on planning how the project should progress by contemplating the question of what our product should be at the end of the project; why it is needed; and how we want to do it. (green outline indicates finished goal.)

- Project plan
- Creating document standards
- Requirements
- Risk analysis
- Our Vision for project
- Deciding team member roles
- Estimated scope of 2 sprints

Foundation

This phase will focus on constructing a solid foundation for our system. The focus will be on updating the engine, implementing support for OSVR, and updating the architecture to support x64.

- Integrating OSVR
- Upgrade to x64 architecture
- Updating the Engine
- Estimated scope of 4 sprints

Extensions

When we enter the Extensions phase, we should have a solid foundation which can be built upon and our focus will shift to create overlays and overall improve the aesthetic feel of the users' experience.

- Improve the GUI
- Develop target tracking
- Estimated scope of 3 sprints

Delivery

In this phase we prepare our documentation for delivery to stakeholders and supervisors. All documentation should be reviewed to ensure that it properly explains the progress of the project and the current state of the product.

- Review documentation
- Deliver documentation to stakeholders and supervisors
- Ensure that the product does not contain dead or unfinished code
- Estimated scope of 1 sprint



Presentation

In this phase we prepare our presentation of the product to stakeholders and supervisors. The presentations should highlight the problems we have solved and how.

- Prepare for the final presentation of the project.
- Present prototype.
- Estimated scope of 1 sprint.

2.2 Project Schedule review

We constructed a project plan to provide our project owner information on how we intended to execute the Argos 2.0 project. This plan represented a “black box” of our project execution and was used to get an overview of timespans when a certain functionality should be finalized. We divided each phase into sprints, which allowed us to work agile on user stories and tasks. We managed to structure our sprints properly, which allowed us to start new phases earlier than estimated. Documented sprint reports can be found in the Appendix section of the documentation, which includes detailed information on each sprint.



3. Initial project Gantt diagram

We wanted to create a diagram which captured the workflow during this bachelor thesis, we decided to make a “Scrumified” Gantt diagram.

Figure 101 explains the project plan in terms of sprints and time estimation for work distribution during the Argos 2.0 project.

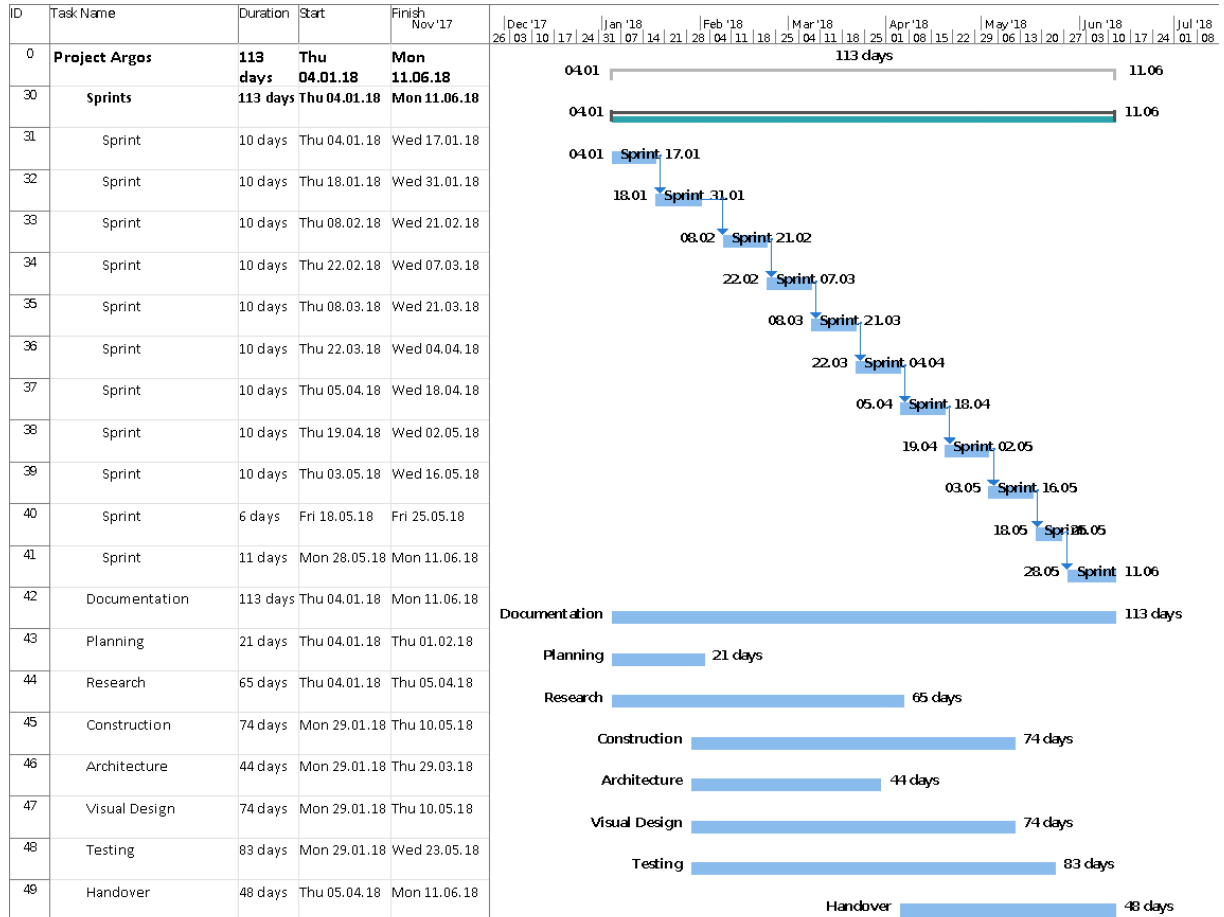


Figure 101: Initial Project Gantt Diagram

3.1 Gantt diagram task description

3.1.1 Documentation

This task estimates the time required for documentation during the project. The reason for it starting and ending with the project timeline is because we continuously create documentation and always aim to keep our documents up to date with the latest information. At certain stages of the project the documentation work will take higher priority, but it is the scrum masters task to ensure that we focus on the correct tasks during our sprints.

3.1.2 Planning

This task estimates time required for planning the project. Planning includes getting a clear understanding of the project owners desires and goal, thereby giving us a firm grasp of what the project should focused upon. Planning also includes constructing a time schedule of where and what should be done when.

3.1.3 Research

This task estimates time required for researching technologies and tools that will benefit the further development of the product. By the end of the research task we should have gathered enough knowledge about our tools to be able to construct our prototype.

3.1.4 Construction

This task estimates time required for constructing the code for our system. The construction task encapsulates both the foundation phase and the extensions phase.

3.1.5 Architecture

This task estimates time required for designing and mapping out the architecture of our system. By the end of this task we should not have to make fundamental changes to the system, but rather focus on building what we have decided upon.

3.1.6 Visual Design

This task estimates time required for the visual presentation of our product to the end user. We aim to not only make our product as best as possible from a technical point of view, but it should also be built with the end user in mind. End users experience is of high importance as our system could be in use for hours and sometimes days at a time, and therefore it should be as visually pleasing as possible as well as easily configurable to suit any changes that can enhance the end users experience.



3.1.7 Testing

This task estimates time required for testing and debugging our system.

For a user story to be completed it needs to enter a stage of testing and if it passes the test requirements it is considered done. However, there will also be further testing for documentation and research purposes. By the time this task ends, everything should be tested, and well documented reports should be available.

3.1.8 Handover

This is the time estimate for the time required to prepare our product for handover to the next stage in development. Argos is a product that has been through several development cycles and will continue to do so for an undisclosed time after our development cycle is finished. It is therefore important to ensure that what you see is what you get and not leave unfinished code in our product.



4. Result Project Gantt diagram

It's always hard to estimate the time that goes into different tasks such as documentation and implementation, but specially for a research & development project. We encountered some challenges which led re-estimation of time. We also ran parallel sprints to differentiate between technical and administrative work. This led to a reiteration of the initial Gantt diagram, which Figure 102 describes.

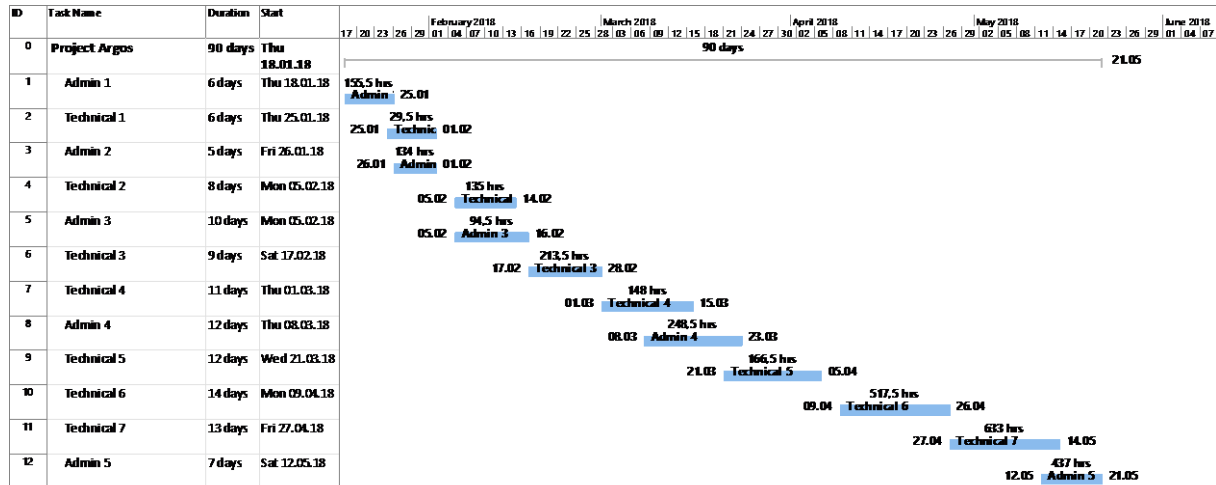


Figure 102: Final Gantt diagram for project Argos 2.0

4.1 Estimation versus Result

4.1.1 Documentation review

Argos 2.0 had ongoing and bulk documentation. We used ongoing documenting for coding and sources, and bulk documentation for presentation / project documents. We did not spend as much time on documenting as estimated, but still found our code and project documentation to deliver solid understanding of the Argos 2.0 project to the reader.

4.1.2 Planning review

Argos 2.0 follows the Scrum project model, which allow for agile development of software and functionality. Sprints were planned based on sprint reviews with stakeholders, and supervisors which ended up taking less time than estimated.

4.1.3 Research review

We did ongoing research, which is typical for a *research* and development project. It is therefore hard to calculate the actual time spent researching. Research had a direct association with implementation challenges.



4.1.4 Construction review

The Argos 2.0 project had some limitations. We we're not allowed to connect the development machines on the internet. This led to a series of challenges regarding installation of Unreal engine and its libraries. There were also challenges related to third party software incorporation with unreal engine, which can be read more about in the Marking (TerraLens) and GigE vision subchapter in the implementation chapter of the Argos 2.0 documentation.

4.1.5 Architecture review

Our system deals with a lot of third-party hardware and software, which makes it hard to model Software architecture according to UML. We had multiple revisions and went back and forth discussing with supervisors and the group, to land an architecture which was reusable and explained the systems software components and sequences. We ended up with what we think is a good explanation of the Argos 2.0 system and software from an architecture point of view. We estimated that this task would take around 41 days because of reasons listed in the beginning of this text, which was enough to get architecture finalized.

4.1.6 Visual design review

This task was approximately estimated correctly.

4.1.7 Testing review

We spent a lot of time debugging software and hardware related problems, as well as running unit and integration tests. The group concludes that the time estimated for testing was correct.

4.1.8 Handover review

The group worked continuously preparing for the project handover. This included building standalone application, cleaning the project solution, and documentation. The group concludes that the time estimated for handover of the application was correct.

4. Conclusion

We managed to provide the functionality within the timeframe we estimated to the project Argos 2.0 stakeholders. We followed the scum process model and divided the project timeframe into sprints. More detailed information on each sprint iteration be found in the "*Technical Sprint report*" and "*Administrative Sprint report*" section of the appendix.





Appendix – C: Administrative Sprint Report

1 Administrative Sprint 1	238
1.1 Sprint report.....	238
1.2 Burn up chart.....	239
1.3 Velocity chart.....	239
1.4 User workload.....	240
1.5 Uncompleted tasks.....	240
1.6 Conclusion	241
2 Administrative Sprint 2	242
2.1 Sprint report.....	242
2.2 Burn up chart.....	243
2.3 Velocity chart.....	243
2.4 User workload.....	244
2.5 Uncompleted tasks.....	244
2.6 Conclusion	244
3 Administrative Sprint 3	245
3.1 Sprint report.....	245
3.2 Burn up chart.....	245
3.3 Velocity chart.....	246
3.4 User workload.....	247
3.5 Uncompleted tasks.....	247
3.6 Conclusion	247
4 Administrative Sprint 4	248
4.1 Sprint report.....	248
4.2 Burn up chart.....	249
4.3 Velocity chart.....	249
4.4 User workload.....	250
4.5 Uncompleted tasks.....	250
4.6 Conclusion	250



1 Administrative Sprint 1

This sprint lasted from 18.01 – 25.01 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:

Create an alpha version of the documentation for the first presentation, and learn how to use the Jira tool

1.1 Sprint report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (-)
AA-1	Requirements	Story	↑ Highest	DONE	-
AA-2	Test plan	Story	↑ High	DONE	-
AA-4	Gant diagram	Story	↑ Medium	DONE	-
AA-12	Templates	Story	↑ Medium	DONE	-
AA-14	Schedule presentations	Story	↑ Medium	DONE	-

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (-)
AA-3	Powerpoint	Story	↑ High	IN PROGRESS	-
AA-6	Quality Control	Story	↑ Medium	TO DO	-
AA-8	Questions	Story	↓ Low	TO DO	-
AA-27	Risk analysis	Story	↑ Medium	IN PROGRESS	-
AA-31 *	1st presentation documentation	Story	↑ Medium	IN PROGRESS	-

Issues completed outside of this sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (-)
AA-5	First presentation list	Story	↑ High	DONE	-
AA-7	Scrum document	Story	↑ Medium	DONE	-
AA-9	Vision document	Story	↑ High	DONE	-
AA-10	Use case diagram	Story	↑ High	DONE	-
AA-11	Bachelor Contract	Story	↑ High	DONE	-
AA-13	Google Drive	Story	↑ High	DONE	-

Figure 103: Sprint Report



1.2 Burn up chart

Startup sprint 1 ▾ Story Points ▾

Work scope Scope projection Completed Work Guideline

Get all the designated documents ready for our first presentation.

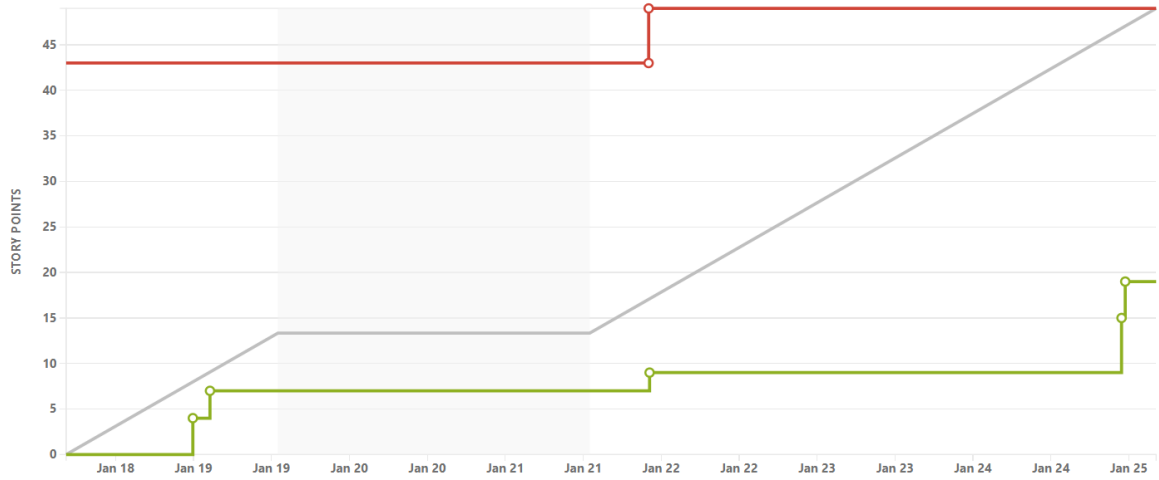


Figure 104: Burn Up Chart

1.3 Velocity chart

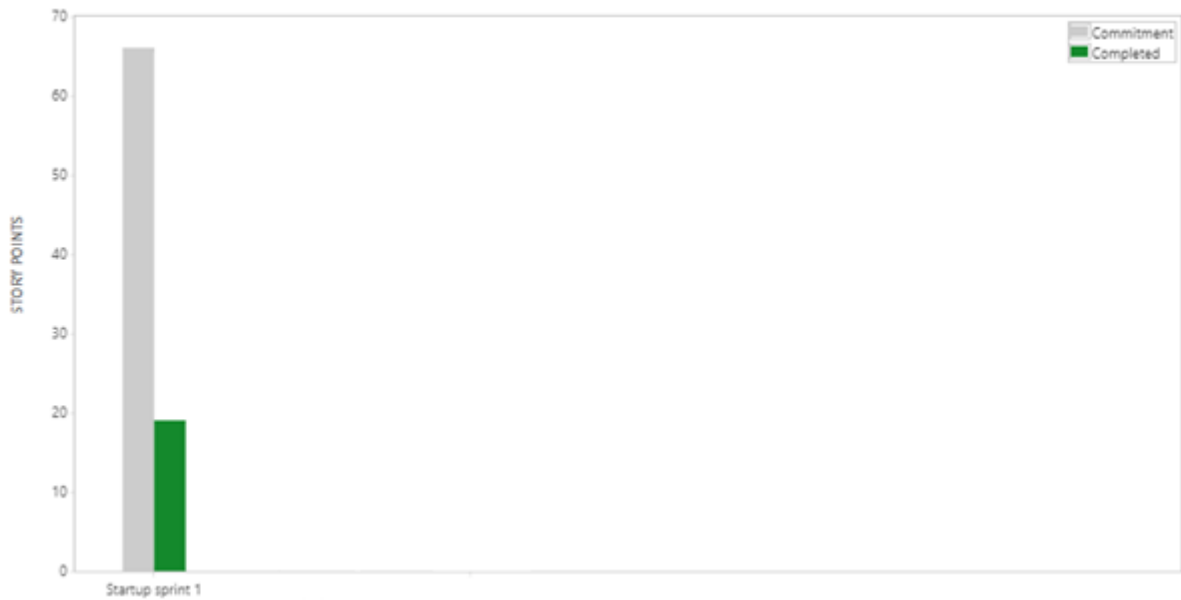


Figure 105: Velocity Chart



1.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	25
Steffen A. Nielsen	Scrum master	28
Magnus E. Muri	Developer	20
Fredrik Kåsin	Developer	24
Vebjørn Tunold	Developer	25
Andreas Holm	Developer	36

Table 85: User Workload

1.5 Uncompleted tasks

Issues Not Completed					View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (-)
AA-3	Powerpoint	Story	↑ High	IN PROGRESS	-
AA-6	Quality Control	Story	↑ Medium	TO DO	-
AA-8	Questions	Story	↓ Low	TO DO	-
AA-27	Risk analysis	Story	↑ Medium	IN PROGRESS	-
AA-31 *	1st presentation documentation	Story	↑ Medium	IN PROGRESS	-

PowerPoint (for first presentation)

- This task was almost completed during the sprint, but we had to lay some final touches on it. It will be completed in the next sprint

Quality Control (of documentation to first presentation)

- This task was not started due to documentation not being finalized. It was included in the next sprint.

Questions (we might get from supervisors during the presentation)

- This task was not started due to lack of time.

Risk analysis (chapter in documentation)

- This task was almost completed during the sprint and will be finalized in the next sprint.

1st presentation document (implementation)

- This task was not completed due to subtasks not being finished in time, such as Risk analysis.

1.6 Conclusion

This was our very first sprint, and it is noticeable. We had some issues adapting to Jira and using it correctly, hence why we got completed tasks outside of the sprint scope. We will learn from this and improve our approach to Scrum during the next sprints. Other than technical challenges, we had good progress and managed to finalize many important chapters for the first presentation. The group had a similar workload, and each member contributed to the outcome of the sprint.



2 Administrative Sprint 2

This sprint lasted from 25.01 – 01.02 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:

Finalize Argos documentation for the first presentation and prepare for the presentation

2.1 Sprint report

Completed Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Story Points (30 – 35)	
AA-3	Powerpoint	Story	High	DONE	7	
AA-6	Quality Control	Story	Medium	DONE	10	
AA-8	Questions	Story	Low	DONE	2	
AA-27	Risk analysis	Story	Medium	DONE	5	
AA-31	1st presentation documentation	Story	Medium	DONE	6	
AA-32	Update project plan	Task	Medium	DONE	-	
AA-33	Adjust documents from feedback	Story	Medium	DONE	- → 5	

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Story Points (-)	
AA-34	Setup Argos laptop	Task	Medium	TO DO	-	

Figure 106: Sprint Report



2.2 Burn up chart

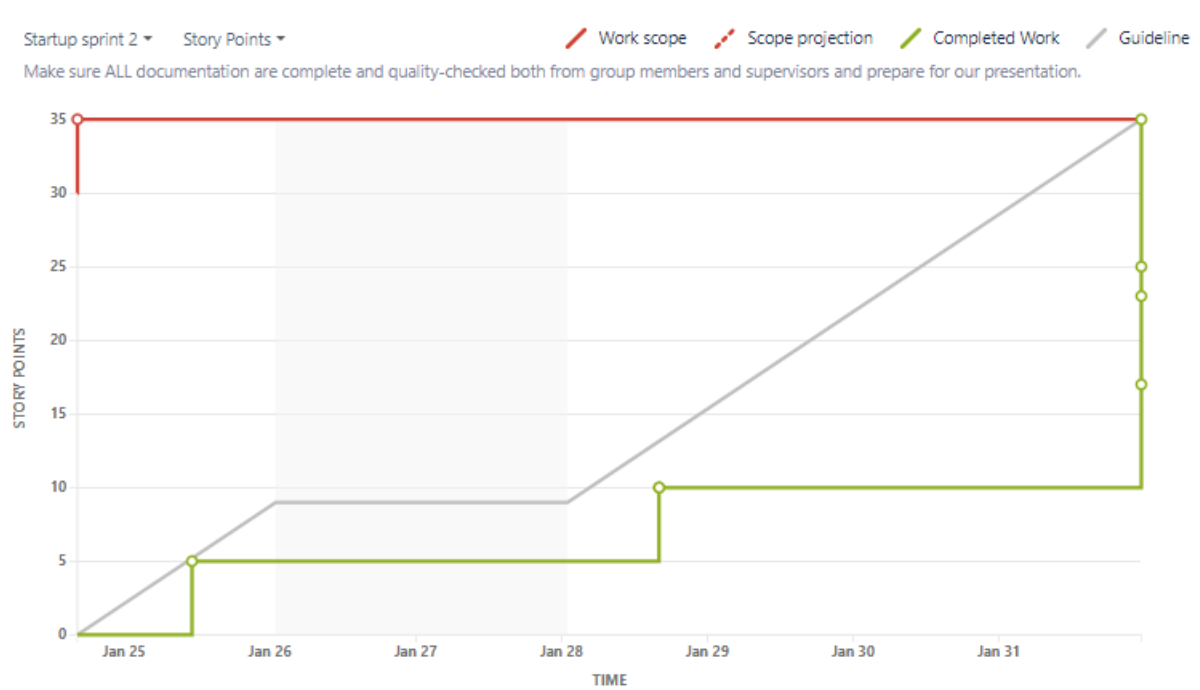


Figure 107: Burn Up Chart

2.3 Velocity chart

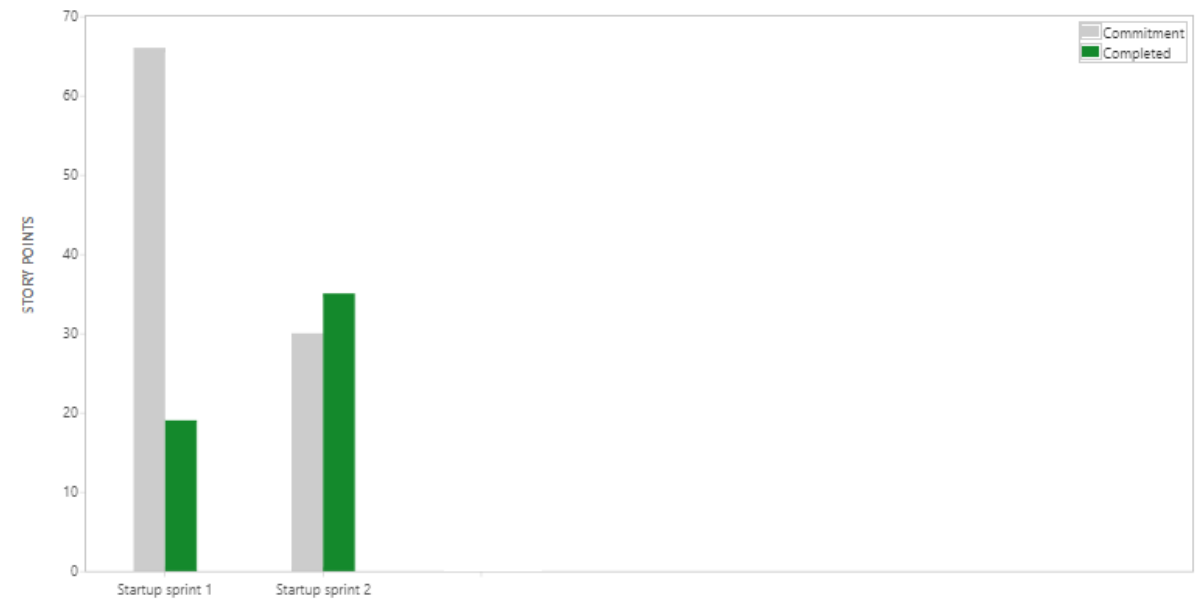


Figure 108: Velocity Chart



2.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	20
Steffen A. Nielsen	Scrum master	22
Magnus E. Muri	Developer	20
Fredrik Kåsin	Developer	20
Vebjørn Tunold	Developer	25
Andreas Holm	Developer	27

Table 86: User Workload

This sprint ran parallel to a technical sprint, and user workload will therefore include this workload as well as workload from technical sprint 2

2.5 Uncompleted tasks

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (-)
AA-34 *	Setup Argos laptop	<input checked="" type="checkbox"/> Task	↑ Medium	IN PROGRESS	-

Setup Argos laptop

- This task was not completed because of lack of time. We will carry out this task over to the next sprint.

2.6 Conclusion

We executed this sprint properly and managed to complete almost every user story. As shown in the burnup chart, we had good progress throughout the entire sprint. We ended up completed the big user stories at the end of the sprint because of their many subtasks. This sprint was running parallel to a technical sprint, which took up some time as well. The sprint goal was met.



3 Administrative Sprint 3

This sprint lasted from 05.02 – 16.02 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:

Fix general administrative tasks to prepare for the second presentation.

3.1 Sprint report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (-)
AA-37	Feedback on the 1. Presentation	Story	Medium	DONE	-
AA-38	Maka a PIT	Story	Medium	DONE	-
AA-39 *	Argos merch	Story	Medium	DONE	-
AA-49 *	Sprint review document	Story	High	DONE	-

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (-)
AA-34 *	Setup Argos laptop	Task	Medium	IN PROGRESS	-

Figure 109: Sprint Report

3.2 Burn up chart

Reach document goals.

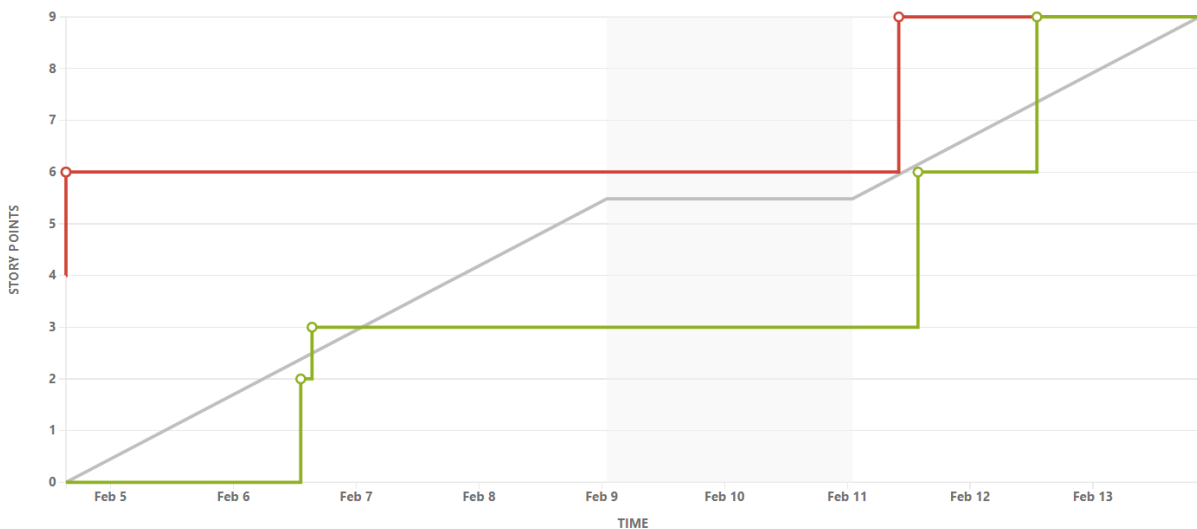


Figure 110: Burn Up Chart



3.3 Velocity chart

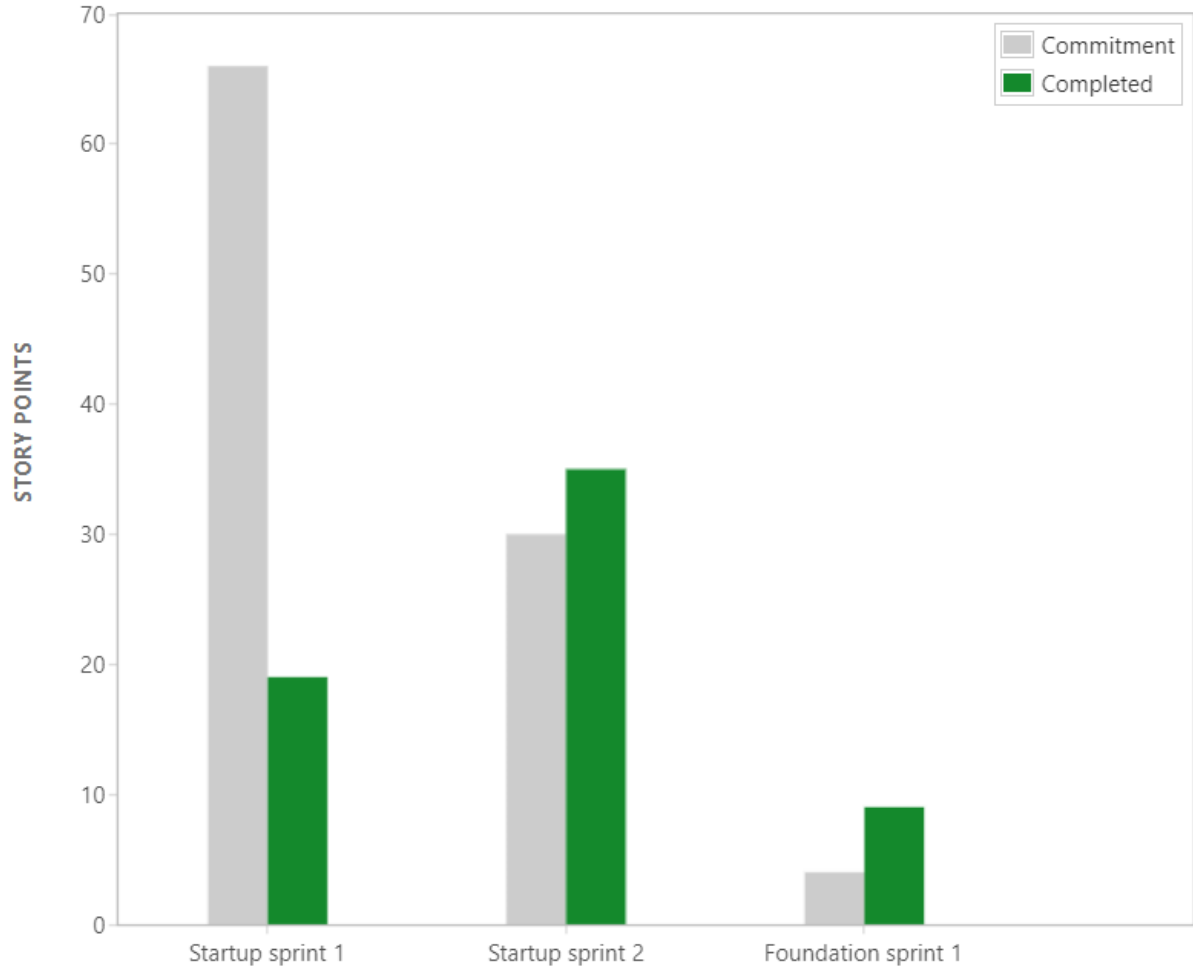


Figure 111: Velocity Chart

3.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	23
Steffen A. Nielsen	Scrum master	14
Magnus E. Muri	Developer	22
Fredrik Kåsin	Developer	23
Vebjørn Tunold	Developer	4,5
Andreas Holm	Developer	8

Table 87: User Workload

This sprint ran parallel to a technical sprint, and user workload will therefore include this workload as well as workload from technical sprint 2

3.5 Uncompleted tasks

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (-)	
AA-3	Powerpoint	Story	High	IN PROGRESS	-	
AA-6	Quality Control	Story	Medium	TO DO	-	
AA-8	Questions	Story	Low	TO DO	-	
AA-27	Risk analysis	Story	Medium	IN PROGRESS	-	
AA-31 *	1st presentation documentation	Story	Medium	IN PROGRESS	-	

Setup Argos laptop

- This task was almost finished and will be moved to finished tasks.

3.6 Conclusion

We executed this sprint properly and managed to complete almost every user story. As shown in the burnup chart, we had good progress throughout the entire sprint. This sprint ran parallel to the “technical sprint two”, and therefore shared workload from the user workload table. We’ve finally beginning to reach some milestones in the project, which feels great! The sprint goal was met.



4 Administrative Sprint 4

This sprint lasted from 08.03 – 23.03 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:

Finalize the documentation for the second presentation.

4.1 Sprint report

Completed Issues [View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1w 1d 4h → 1w 3d 4h)
AA-50	Make architecture document	Story	High	DONE	1d
AA-51	Make our contribution to Argos	Story	High	DONE	1d
AA-52	Challenges	Story	Medium	DONE	- → 1d
AA-54	Smaller tasks for the entire document	Story	Medium	DONE	1d
AA-55	Update front pages for chapters	Story	Low	DONE	4h
AA-56	update vision document	Story	Medium	DONE	4h
AA-57	Update requirements document	Story	High	DONE	1d
AA-58	Quality Control of all documents	Story	Medium	DONE	1d
AA-80 *	Make a budget the project	Story	Medium	DONE	-
AA-81 *	Make the PowerPoint	Story	Medium	DONE	- → 1d
AA-82 *	Make GUI document	Story	Medium	DONE	-
AA-83 *	Write "Argos Design Choices" Document	Story	Highest	DONE	4h

Figure 112: Sprint Report



4.2 Burn up chart

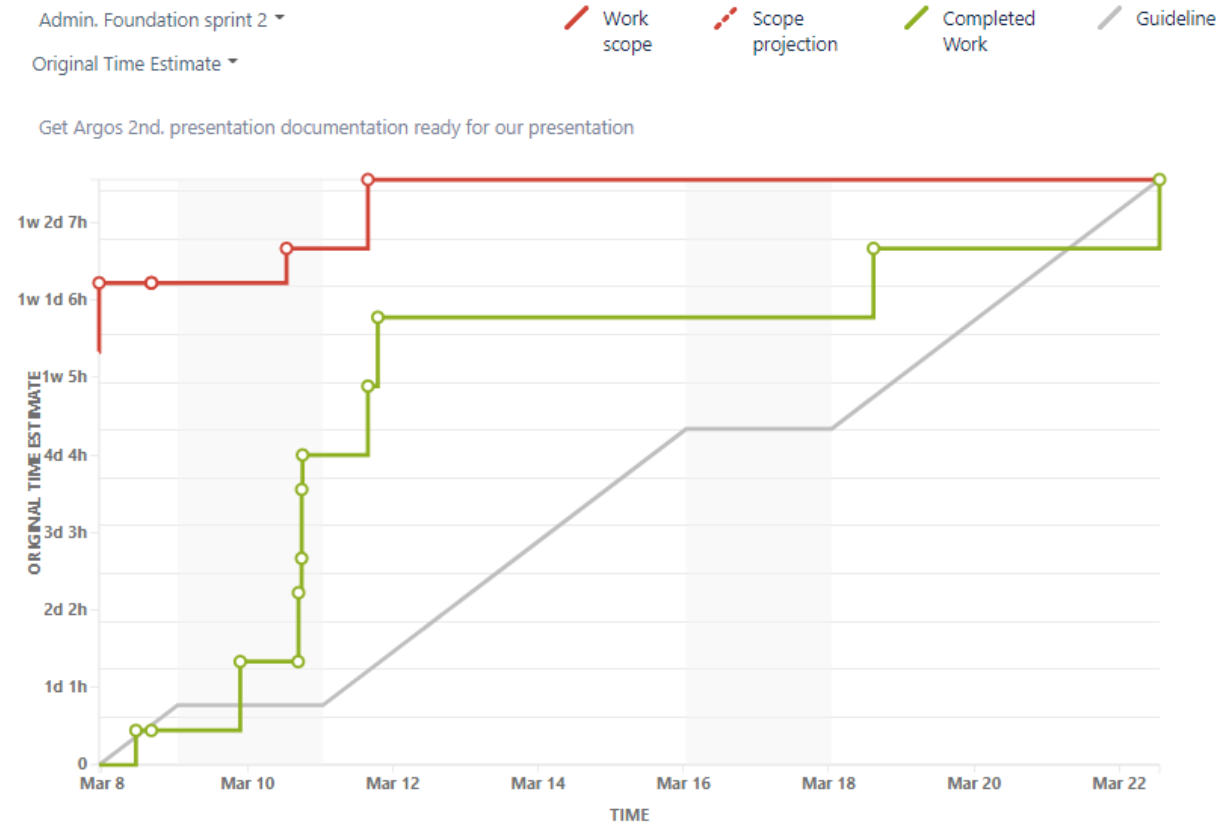


Figure 113: Burn Up Chart

4.3 Velocity chart

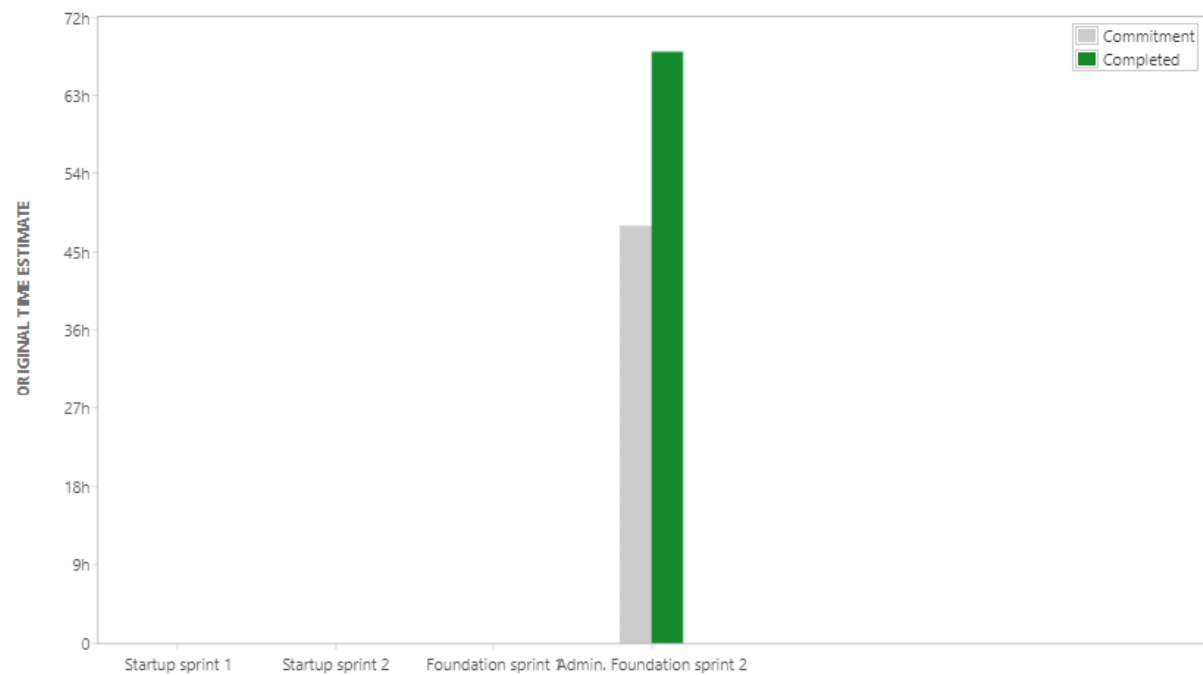


Figure 114: Velocity Chart



4.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	53
Steffen A. Nielsen	Scrum master	44
Magnus E. Muri	Developer	49
Fredrik Kåsin	Developer	43
Vebjørn Tunold	Developer	36,5
Andreas Holm	Developer	23

Table 88: User Workload

This sprint ran parallel to a technical sprint, and user workload will therefore include this workload as well as workload from technical sprint 4

4.5 Uncompleted tasks

All tasks were finished during this sprint

4.6 Conclusion

We planned and executed this sprint to perfection. We managed to have a steady work flow and complete all tasks within the scope of the sprint. This sprint was an administrative sprint working towards the second presentation of our bachelor thesis. It ran semi-parallel to a technical sprint, but we managed to keep both under control. The group members working passionate during this sprint, which led to a good result both technically and administrative for our second presentation. The Sprint goal was met.





Appendix – D: Technical Sprint Report

1	Technical Sprint 1	253
1.1	Sprint report.....	253
1.2	Burn up chart.....	253
1.3	Velocity chart.....	254
1.4	User workload.....	254
1.5	Uncompleted tasks.....	254
1.6	Conclusion.....	255
2	Technical Sprint 2	255
2.1	Sprint report.....	255
2.2	Burn up chart.....	256
2.3	Velocity chart.....	256
2.4	User workload.....	257
2.5	Uncompleted tasks.....	257
2.6	Conclusion.....	257
3	Technical Sprint 3	258
3.1	Sprint report.....	258
3.2	Burn up chart.....	258
3.3	Velocity chart.....	259
3.4	User workload.....	259
3.5	Uncompleted tasks.....	259
3.6	Conclusion.....	260
4	Technical Sprint 4	260
4.1	Sprint report.....	260
4.2	Burnup chart.....	261
4.3	Velocity chart.....	262
4.4	User workload.....	262
4.5	Uncompleted tasks.....	263
4.6	Conclusion.....	263
5	Technical Sprint 5	264
5.1	Sprint report.....	264
5.2	Burnup chart.....	264
5.3	Velocity chart.....	265
5.4	User workload.....	265



5.5 Uncompleted tasks.....	266
5.6 Conclusion	267
6 Technical Sprint 6	268
6.1 Sprint report.....	268
6.2 Burnup chart	269
6.3 Velocity chart	269
6.4 User workload.....	270
6.5 Uncompleted tasks.....	270
6.6 Conclusion	271
7 Technical Sprint 7	271
7.1 Sprint report.....	272
7.2 Burn up chart.....	273
7.3 Velocity chart.....	273
7.4 User workload.....	274
7.5 Uncompleted tasks.....	274
7.6 Conclusion	274



1 Technical Sprint 1

This sprint lasted from 25.01 – 01.02 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:
Complete the web page for Argos 2.0

1.1 Sprint report

Completed Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Story Points (12)	
RGS2018-15	Guidelines	Story	Low	DONE	3	
RGS2018-18	Web page update	Story	Medium	DONE	3	
RGS2018-19	Guidelines	Story	Low	DONE	1	
RGS2018-29	360 camera	Story	Highest	DONE	5	

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Story Points (11)	
RGS2018-21	Remove errors in old Argos	Story	High	TO DO	6	
RGS2018-30	Source control	Story	High	IN PROGRESS	5	

Figure 115: Sprint Report

1.2 Burn up chart

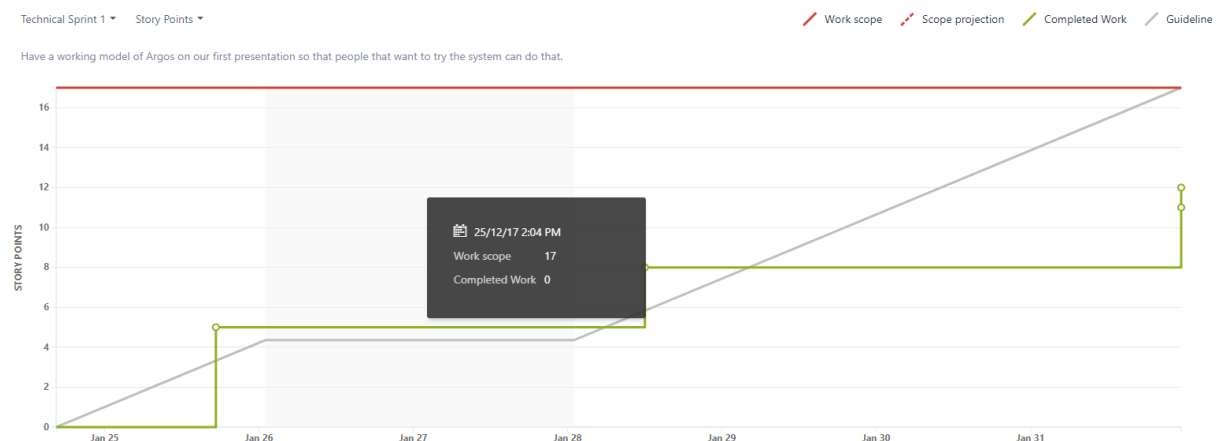


Figure 116: Burn up Chart



1.3 Velocity chart

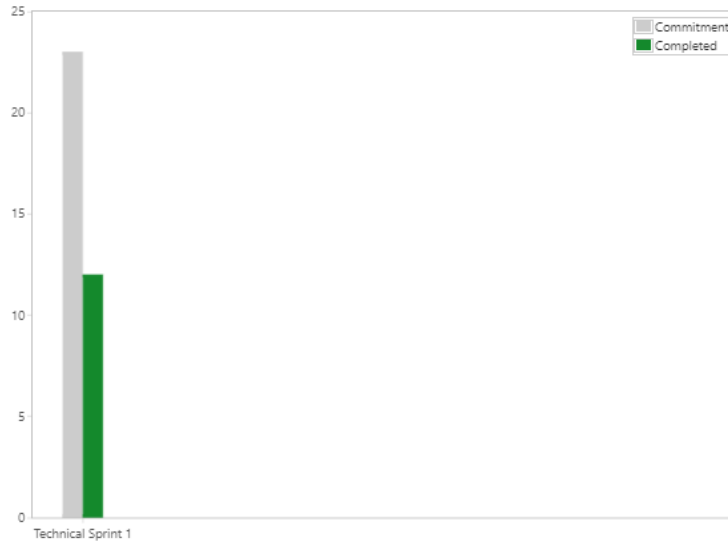


Figure 117: Velocity Chart

1.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	4
Steffen A. Nielsen	Scrum master	5
Magnus E. Muri	Developer	5,5
Fredrik Kåsin	Developer	2
Vebjørn Tunold	Developer	5
Andreas Holm	Developer	8

Table 89: User Workload

This sprint ran parallel to a technical sprint, and user workload will therefore include this workload as well as workload from administrative sprint 2

1.5 Uncompleted tasks

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Story Points (11)	
RGS2018-21	Remove errors in old Argos	Story	High	TO DO	6	
RGS2018-30	Source control	Story	High	IN PROGRESS	5	

Remove errors in old Argos

- This story was not completed due to lack of time and will be worked on during the next sprint.

Source control

- This story was not completed due to challenges with Windows and will be worked on in the next spring

1.6 Conclusion

This Sprint was a stand-aside sprint that went parallel to our Administrative sprint for our first presentation. The prioritized administrative sprint took more time than planned, which resulted in the initial technical sprint being out-prioritized. The sprint goal was met.

















2 Technical Sprint 2

This sprint lasted from 05.02 – 14.02 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:

Complete installation of software and get Argos 1.0 up and running.

2.1 Sprint report

Completed Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Story Points (37)	
RGS2018-30	Source control	 Story	 High	DONE	5	
RGS2018-32	Look into doxygen	 Story	 High	DONE	4	
RGS2018-33	Set up KDA machines	 Story	 High	DONE	3	
RGS2018-34	Get the laptop up and running argos playback	 Story	 Medium	DONE	5	
RGS2018-35	Make clean install document	 Story	 Medium	DONE	3	
RGS2018-36	Look into GPS component	 Story	 High	DONE	5	
RGS2018-38	Housing for Insta 360	 Story	 Medium	DONE	7	
RGS2018-49 *	Get Unreal engine up and running	 Story	 Medium	DONE	5	



Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Story Points (6)	
RGS2018-21	Remove errors in old Argos	 Story	 High	IN PROGRESS	6	

Figure 118: Sprint Report



2.2 Burn up chart

Technical Sprint 2 ▾ Story Points ▾

Start building Argos foundation and finish administrative tasks.

Work scope Scope projection Completed Work Guideline

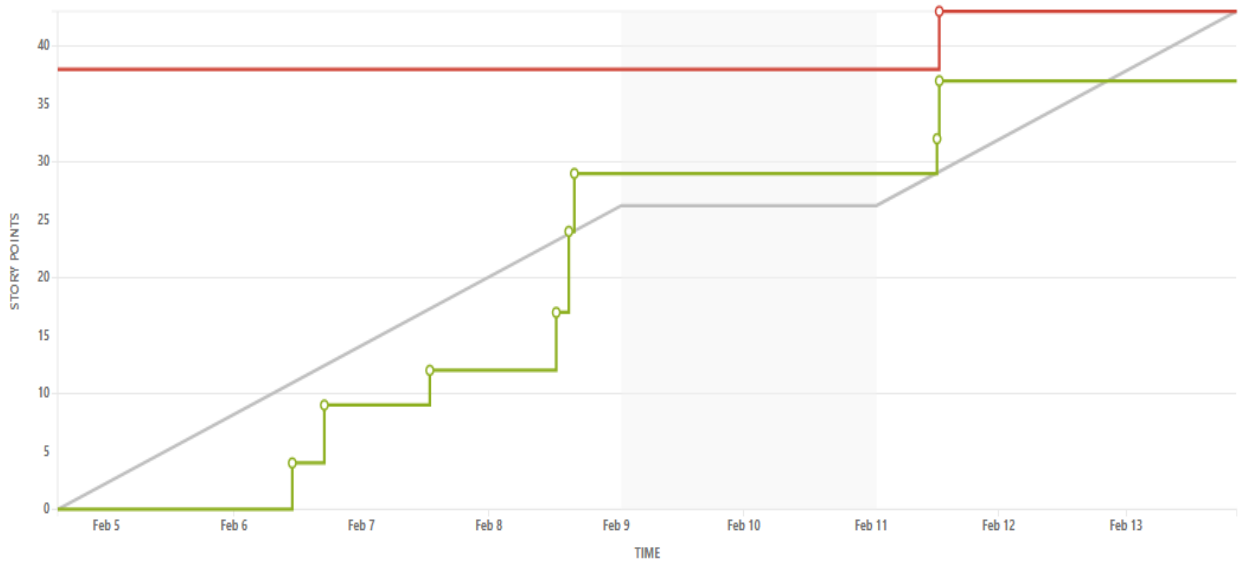
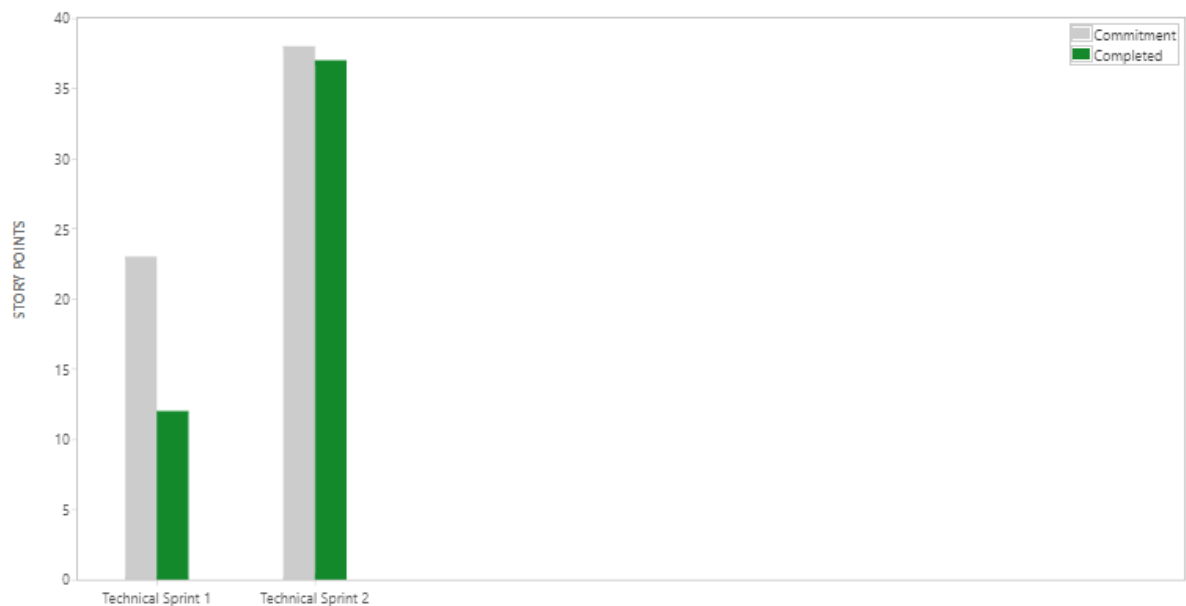


Figure 119: Burn Up Chart

2.3 Velocity chart



Sprint	Commitment	Completed
Technical Sprint 1	23	12
Technical Sprint 2	38	37

Figure 120: Velocity Chart

2.4 User workload



This sprint had a duration of 9 days because we wanted to reset our sprint days to Thursday when he had meetings with Alex.

Members:	Role:	Hours
Henrik Gjestvang	Product owner	9
Steffen A. Nielsen	Scrum master	29
Magnus E. Muri	Developer	33
Fredrik Kåsin	Developer	0
Vebjørn Tunold	Developer	35
Andreas Holm	Developer	37

Table 90: User Workload

This sprint ran parallel to a technical sprint, and user workload will therefore include this workload as well as workload from administrative sprint 3

2.5 Uncompleted tasks

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Story Points (6)	
RG2018-21	Remove errors in old Argos	 Story	 High	IN PROGRESS	6	

Remove Errors in old Argos

- This task was not completed due to complex errors which could not be fixed at the given time. We had errors with libraries, macros and features not working properly.

2.6 Conclusion

We executed this sprint properly and managed to complete almost every user story. As shown in the burn up chart, we had good progress throughout the entire sprint. Every group members contributed, and our external supervisor was pleased with the results. The sprint goal was met.



3 Technical Sprint 3

This sprint lasted from 15.02 – 28.02 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:
Create Argos 2.0 architecture draft and implement OSVR

3.1 Sprint report

Completed Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1w 1d 7h)	
RGS2018-20	Unreal engine as framework	Story	High	DONE	2h	
RGS2018-22	OSVR SDK	Story	Medium	DONE	2h	
RGS2018-26	Remove Oculus-dependent code	Story	Medium	DONE	5h	
RGS2018-28	Update 3rd party libraries	Story	Medium	DONE	4h	
RGS2018-50	SW Architecture	Story	High	DONE	1d	
RGS2018-51	Install Unreal Engine	Story	Highest	DONE	1d	
RGS2018-53	Source control for Unreal Engine	Story	High	DONE	3h	
RGS2018-54	Testing 360 camera	Story	Medium	DONE	5h	
RGS2018-56	Web page	Story	High	DONE	6h	
RGS2018-57	System arketecture	Story	High	DONE	1d	
RGS2018-58	Enviroment picture	Story	High	DONE	4h	

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (6h -- 2d 2h)	
RGS2018-48	Ebus licence	Story	Low	TO DO	2h	
RGS2018-52	Research Unreal Engine	Story	Medium	TO DO	4h -- 2d	

Figure 121: Sprint Report

3.2 Burn up chart

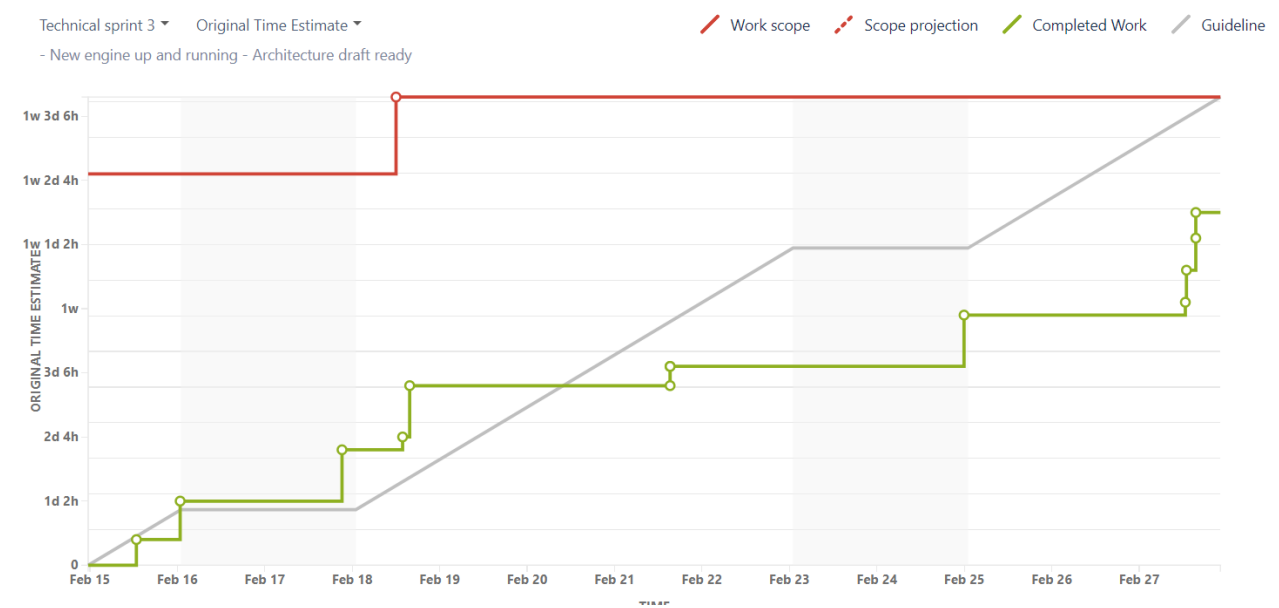


Figure 122: BurnUp Chart

3.3 Velocity chart

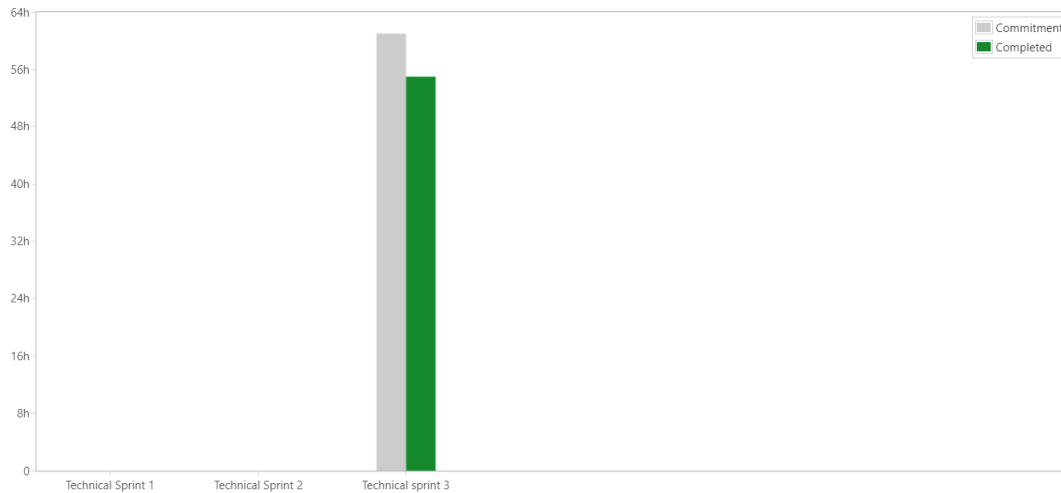


Figure 123: Velocity Chart

3.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	34
Steffen A. Nielsen	Scrum master	40
Magnus E. Muri	Developer	33
Fredrik Kåsin	Developer	31
Vebjørn Tunold	Developer	38,5
Andreas Holm	Developer	37

Table 91: User Workload

3.5 Uncompleted tasks

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (6h → 2d 2h)
RGS2018-48	Ebus licence	Story	Low	TO DO	2h
RGS2018-52	Research Unreal Engine	Story	Medium	TO DO	4h → 2d

Figure 124: Uncompleted Tasks

Ebus License

- This task was not completed due to lack of available license from KDA.

Research Unreal Engine

- This task was not completed due to lack of time and will be carried over to the next sprint.



3.6 Conclusion

We executed this sprint properly and managed to complete almost every user story. As shown in the burnup chart, we had good progress throughout the entire sprint. Every group members contributed, and our external supervisor was pleased with the results. We had some issues with estimating story points, which basically just because an estimate of time. We will therefore investigate estimated time for user stories instead of story point for the next sprint. This will reset our Velocity chart, but we think it will help us and our supervisor to understand the extent of these user stories. Sprint goal was met.

4 Technical Sprint 4

This sprint lasted from 01.03 – 15.03 (ended 21.03 2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:

Produce an alpha demo of the Argos 2.0 project

4.1 Sprint report

Completed Issues [View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1w 4d 2h → 1w 4d 6h)
RGS2018-62	Install graphic cards	Story	↑ Medium	DONE	2h
RGS2018-64	OSVR low quality and gitters	Story	↑ Highest	DONE	1d
RGS2018-77	Software architecture	Story	↑ Highest	DONE	3d
RGS2018-78	One to one ratio for sphere	Story	↑ Medium	DONE	1h
RGS2018-79	Optimize lighting	Story	↑ Medium	DONE	2h
RGS2018-80	Ignore HG	Story	↑ Medium	DONE	4h
RGS2018-84	Media playing features in C++	Story	↑ Medium	DONE	1d
RGS2018-85	Identify problems with viewport	Story	↑ High	DONE	6h
RGS2018-86	Fix problems with color in videosteam	Story	↑ High	DONE	4h
RGS2018-87	Make the sphere an actor	Story	↑ Medium	DONE	1h
RGS2018-88	Make DefaultPawn Blueprint a Pawn Class	Story	↑ Medium	DONE	4h
RGS2018-89	Make GameMode Blueprint as an GameMode Class	Story	↑ Medium	DONE	→ 4h
RGS2018-90	Record and livestream video at the same time	Story	↑ High	DONE	1d
RGS2018-92	Install Visio	Story	↑ Highest	DONE	2h



Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1w 4d 7h)
RGS2018-48	Ebus licence	Story	Low	TO DO	2h
RGS2018-52	Research Unreal Engine	Story	Medium	TO DO	2d
RGS2018-71	Make compass	Story	Medium	TO DO	1d
RGS2018-72	Argos menu	Story	Medium	IN PROGRESS	1d
RGS2018-75	Distance calculation unit	Story	Medium	IN PROGRESS	2d
RGS2018-82	Redundancy for headset	Story	High	TO DO	4h
RGS2018-83	Restrict movement for camera in unreal	Story	Medium	TO DO	1h
RGS2018-91	Inegrate old cameras with new argos	Epic	High	IN PROGRESS	3d

Figure 125: Sprint Report

4.2 Burnup chart

Optimize engine foundation framework.

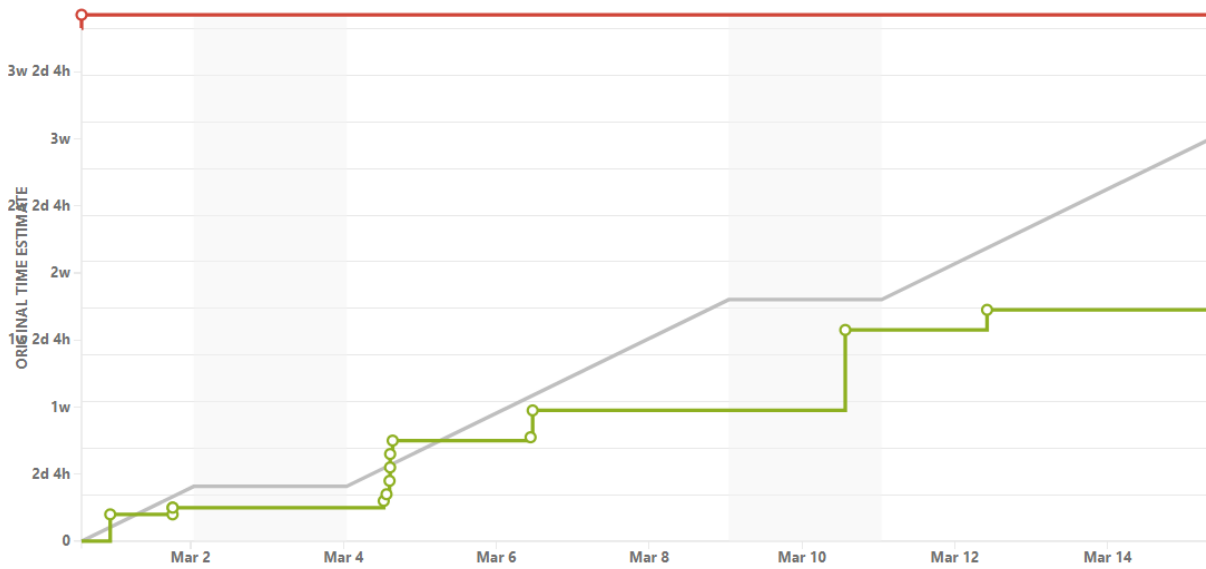


Figure 126: Burn Up Chart



4.3 Velocity chart

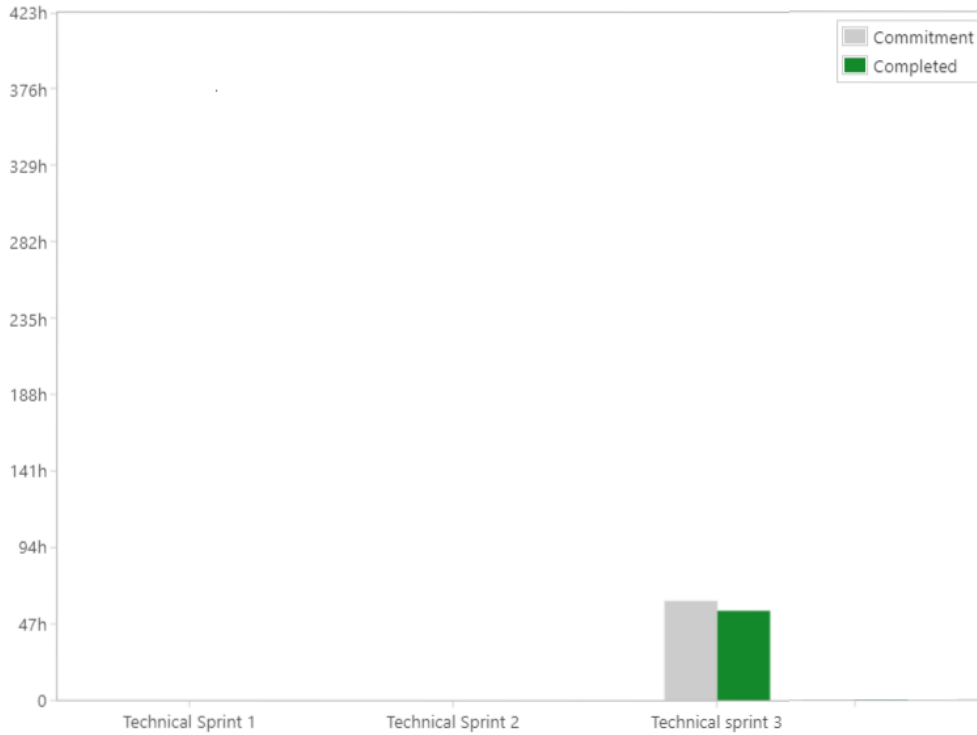


Figure 127: Velocity Chart

4.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	26
Steffen A. Nielsen	Scrum master	33
Magnus E. Muri	Developer	10
Fredrik Kåsin	Developer	13
Vebjørn Tunold	Developer	38
Andreas Holm	Developer	28

Table 92: User Workload

This sprint ran parallel to a technical sprint, and user workload will therefore include this workload as well as workload from administrative sprint 4



4.5 Uncompleted tasks

Issues Not Completed					View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1w 4d 7h)
RGS2018-48	Ebus licence	Story	Low	TO DO	2h
RGS2018-52	Research Unreal Engine	Story	Medium	TO DO	2d
RGS2018-71	Make compass	Story	Medium	TO DO	1d
RGS2018-72	Argos menu	Story	Medium	IN PROGRESS	1d
RGS2018-75	Distance calculation unit	Story	Medium	IN PROGRESS	2d
RGS2018-82	Redundancy for headset	Story	High	TO DO	4h
RGS2018-83	Restrict movement for camera in unreal	Story	Medium	TO DO	1h
RGS2018-91	Inegrate old cameras with new argos	Epic	High	IN PROGRESS	3d

Convert GiGe buffer to MP4

Ebus License

- This user story was not completed due to lack of available license from KDA.

Research Unreal Engine

- This task was not completed due to lack of time and will be carried over to the next sprint.

Make compass

- This user story was completed due to lack of time

Argos menu

- This user story was completed due to lack of time

Distance Calculation

- This user story was completed due to lack of time

Redundancy for headset

- This task was not completed due to lack of time and will be carried over to the next sprint.

Restrict for cameras in unreal

- This task was not completed due to lack of time and will be carried over to the next sprint.

Integrate old cameras with new Argos

- This task is close to finished but took a lot more time than estimated.

4.6 Conclusion

The sprint goal was met. We had good progression, but wrongly estimated some of the user stories. This created a domino effect which resulted in some tasks not being started, while others not being completed. Overall, we had decent progress, and managed to complete high priority tasks.



5 Technical Sprint 5

This sprint lasted from 21.03 – 05.04 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:
Finalized the Foundation sprint

5.1 Sprint report

Completed Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (2d 1h)	
RGS2018-71	Make compass	Story	Medium	DONE	1d	
RGS2018-72	Argos menu	Story	Medium	DONE	1d	
RGS2018-83	Restrict movement for camera in unreal	Story	Medium	DONE	1h	

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3w 1d 2h)	
RGS2018-48	Ebus licence	Story	Low	TO DO	2h	
RGS2018-52	Research Unreal Engine	Story	Medium	TO DO	2d	
RGS2018-63	connect GPS to Argos	Story	Medium	TO DO	2d	
RGS2018-65	Add teralens to Argos 2.0	Story	Medium	IN PROGRESS	4d	
RGS2018-66	GUI overlay for Argos	Story	Medium	TO DO	4h	
RGS2018-67	Object raycasting	Story	Medium	TO DO	2d	
RGS2018-75	Find distance	Story	Medium	IN PROGRESS	2d	
RGS2018-82	Redundancy for headset	Story	High	TO DO	4h	
RGS2018-91	Inegrate old cameras with new argos	Story	High	IN PROGRESS	3d	

Figure 128: Sprint Report

5.2 Burnup chart

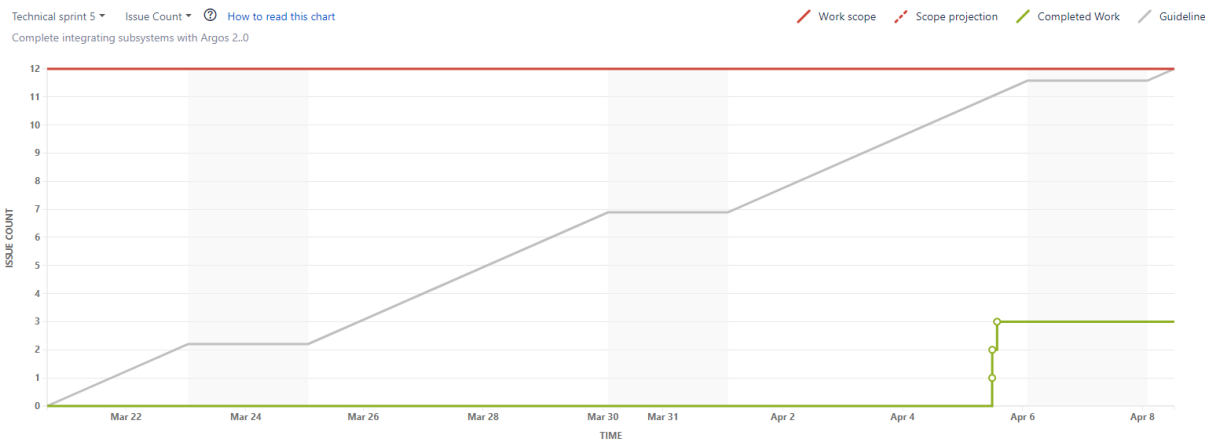


Figure 129: BurnUp Chart



5.3 Velocity chart

Velocity Chart

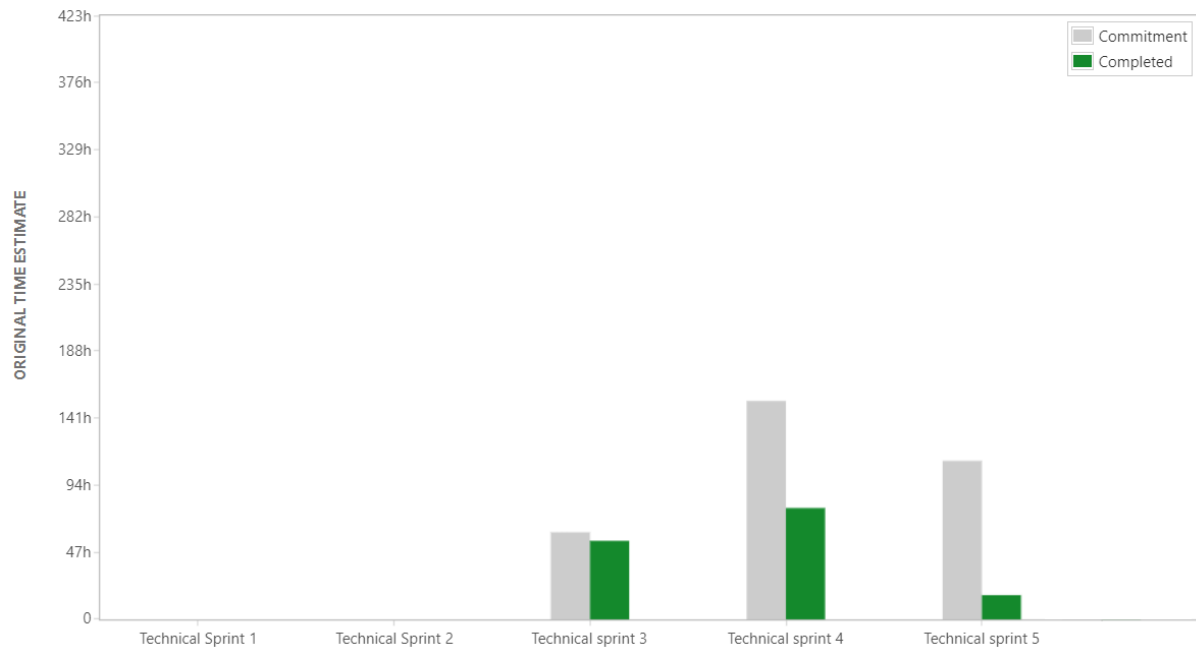


Figure 130: Velocity Chart

5.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	20
Steffen A. Nielsen	Scrum master	42
Magnus E. Muri	Developer	15
Fredrik Kåsin	Developer	23
Vebjørn Tunold	Developer	30,5
Andreas Holm	Developer	36

Table 93: User Workload



5.5 Uncompleted tasks

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3w 1d 2h)	
RGS2018-48	Ebus licence	Story	Low	TO DO	2h	
RGS2018-52	Research Unreal Engine	Story	Medium	TO DO	2d	
RGS2018-63	connect GPS to Argos	Story	Medium	TO DO	2d	
RGS2018-65	Add teralens to Argos 2.0	Story	Medium	IN PROGRESS	4d	
RGS2018-66	GUI overlay for Argos	Story	Medium	TO DO	4h	
RGS2018-67	Object raycasting	Story	Medium	TO DO	2d	
RGS2018-75	Find distance	Story	Medium	IN PROGRESS	2d	
RGS2018-82	Redundancy for headset	Story	High	TO DO	4h	
RGS2018-91	Inegrate old cameras with new argos	Story	High	IN PROGRESS	3d	

Ebus License

- This task was not completed due to lack of available license from KDA.

Research Unreal Engine

- This task was not completed due to lack of time and will be carried over to the next sprint.

Connect GPS to Argos

- This task was not completed due to lack of time and will be carried over to the next sprint.

Add TerraLens to Argos 2.0

- This task was not completed due to higher complexity than estimation, and deprecated code / libraries, and will be carried out over the next sprint

GUI overlays

- This task was not completed due to lack of time and will be carried over to the next sprint.

Object Raycasting

- This task was not completed due to lack of time and will be carried over to the next sprint.

Find distance

- This task is completed and will be switched to completed issues.

Redundancy for headset

- This task was not completed due to lack of time and will be carried over to the next sprint.

Integrate old cameras with new Argos

- This task is close to finished but took a lot more time than estimated.



5.6 Conclusion

This sprint was squeezed in between Easter and other exams which took priority away from the sprint. We also have huge user stories which took much more time than first estimated, because of lack of knowledge of Unreal engine and the way to implement third party software. Although the sprint did not meet expectations, we learned and debugged A LOT. In our next sprint we will create more and smaller user stories to make sure our progress is visible to the reader, and make sure we're back on track. Sprint goal was met.



6 Technical Sprint 6

This sprint lasted from 09.04 – 26.04 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:

Finalize “Tracking and Marking”, “GUI overlays”, “GigE vision”, and “TerraLens” Epics

6.1 Sprint report

Completed Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (6w 1d)	
RGS2018-75	Distance calculation unit	Story	Medium	DONE	2d	
RGS2018-115	Build TerraLens with UE4	Story	Medium	DONE	2d	
RGS2018-116	TerraLens static libraries in UE4	Story	Medium	DONE	4h	
RGS2018-117	TerraLens (DirectX vs OpenGL)	Story	Medium	DONE	2h	
RGS2018-118	Make GiGe build with application	Story	Medium	DONE	3d	
RGS2018-119	Display feed in Argos 2.0 application	Story	Medium	DONE	3d	
RGS2018-120	Establish connection with MAKO cameras in Argos 2.0	Story	Medium	DONE	4d	
RGS2018-121	Make GiGe buffer available in Argos 2.0 application	Story	Medium	DONE	4d	
RGS2018-123	Transform real world coordinates to cartesian coordinates	Story	Medium	DONE	4d	
RGS2018-124	Bearing formula for target tracking	Story	Medium	DONE	1d	
RGS2018-125	Rotation of vectors between two objects	Story	Medium	DONE	4d	
RGS2018-126	Test distance in virtual world vs real world	Story	Medium	DONE	2h	
RGS2018-127	Implement formulas into UE4	Story	Medium	DONE	2d	
RGS2018-128	Create a coordinate class (object buffer)	Story	Medium	DONE	4h	
RGS2018-130 *	Find target in focus	Story	Medium	DONE	4h	

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3w 1d)	
RGS2018-63	connect GPS to Argos 2.0	Story	Medium	TO DO	2d	
RGS2018-66	GUI overlay for Argos	Story	Medium	IN PROGRESS	4h	
RGS2018-67	Object raycasting	Story	Medium	IN PROGRESS	2d	
RGS2018-70	Place objects in TerraLens	Story	Medium	TO DO	2d	
RGS2018-76	GUI for position and distance information	Story	Medium	TO DO	3d	
RGS2018-107	Objectbuffer & BMS	Story	Medium	TO DO	4h	
RGS2018-108	Object test	Story	Medium	TO DO	1d	
RGS2018-113	TerraLens as UE4 object	Story	Medium	IN PROGRESS	2d	
RGS2018-114	TerraLens connected to GPS	Story	Medium	TO DO	2d	
RGS2018-129	Implement Marking & Tracking as a UE4 plugin	Story	Medium	TO DO	1d	

Figure 131: Sprint Report



6.2 Burnup chart

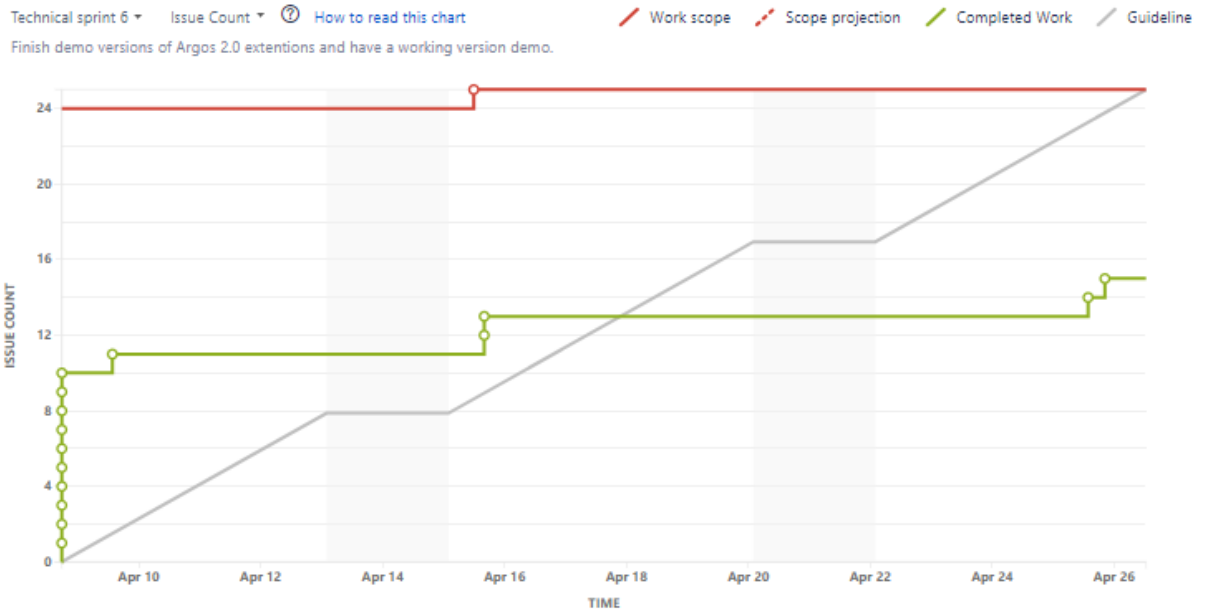


Figure 132: BurnUp Chart

6.3 Velocity chart

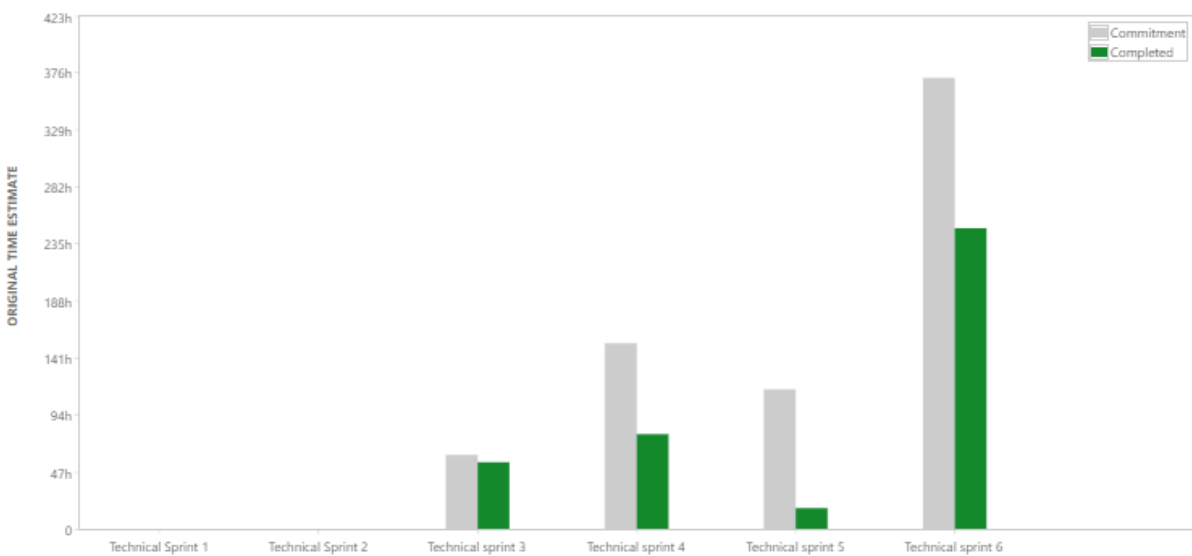


Figure 133: Velocity Chart



6.4 User workload

Members:	Role:	Hours
Henrik Gjestvang	Product owner	82
Steffen A. Nielsen	Scrum master	84
Magnus E. Muri	Developer	79
Fredrik Kåsin	Developer	87
Vebjørn Tunold	Developer	94,5
Andreas Holm	Developer	91

Table 94: User Workload

6.5 Uncompleted tasks

Issues Not Completed					View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3w 1d)
RGS2018-63	connect GPS to Argos 2.0	Story	Medium	TO DO	2d
RGS2018-66	GUI overlay for Argos	Story	Medium	IN PROGRESS	4h
RGS2018-67	Object raycasting	Story	Medium	IN PROGRESS	2d
RGS2018-70	Place objects in TerraLens	Story	Medium	TO DO	2d
RGS2018-76	GUI for position and distance information	Story	Medium	TO DO	3d
RGS2018-107	Objectbuffer & BMS	Story	Medium	TO DO	4h
RGS2018-108	Object test	Story	Medium	TO DO	1d
RGS2018-113	TerraLens as UE4 object	Story	Medium	IN PROGRESS	2d
RGS2018-114	TerraLens connected to GPS	Story	Medium	TO DO	2d
RGS2018-129	Implement Marking & Tracking as a UE4 plugin	Story	Medium	TO DO	1d

Connect GPS to Argos 2.0

- This task was not completed because we had to write a parser for the GPS

GUI overlay for Argos (tracking)

- This task was not completed because more important user stories took more time to complete than estimated

Place objects in TerraLens

- This task was not completed due to “TerraLens as UE4 object” not being completed.

Gui for position and distance information

- This task was not completed due to lack of time and will be carried over to the next sprint.

Objectbuffer & BMS

- This task was not completed due to lack of time and will be carried over to the next sprint. BMS will also be removed from the task scope.

Object Test

- This task was not completed due to “Gui for position and distance information” not being completed.

TerraLens as UE4 object



- This task was not completed due to higher complexity than estimated, and problems between TerraLens integrated OpenGL functionality and UE4 rendering engine.

TerraLens connected to GPS

- This task was not completed due to “TerraLens as UE4 object” not being completed.

Implement Marking & Tracking as a UE4 plugin

- This task was not completed due to lack of time and will be carried over to the next sprint.

6.6 Conclusion

This sprint contained 4 epics and some user stories. The estimated goal with this sprint was complete each subsystem of the Argos 2.0 application and integrate them with UE4. We we're able to implement all the subsystems except TerraLens, which is currently set to inactive. The group worked good throughout the sprint, but the stories and epics we're simply more complicated than first estimated. Even though we were not able to complete all user stories, we're still on track when having all functionality integrated before the deadline. Sprint goal was not met.

7 Technical Sprint 7

This sprint lasted from 27.04 - 14.05 (2018). This document describes how the sprint was executed and finished. It also explains why certain user stories was not finished, as well as how the group members experienced the sprint.

Sprint goal:

Finalize functionality for the Argos 2.0 application.



7.1 Sprint report

Figure 134: Sprint Report

Sprint Report



Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (9w 6h)
RGS2018-67	Object tracing using headset	Story	Medium	DONE	2d
RGS2018-76	GUI for position and distance information	Story	Medium	DONE	3d
RGS2018-82	Redundancy for headset	Story	High	DONE	4h
RGS2018-108	Object GUI information test	Story	Medium	DONE	1d
RGS2018-129	Implement Marking & Tracking as a UE4 plugin	Story	Medium	DONE	1d
RGS2018-134	GUI displayed in headset	Story	Medium	DONE	2d
RGS2018-135	Sync GUI with headset rotation	Story	Medium	DONE	2d
RGS2018-136	Argos Menu is displayed in VR	Story	Medium	DONE	2d
RGS2018-139	Made a VideoStreamHandler	Story	Medium	DONE	3d
RGS2018-141	File directory for video files	Story	Medium	DONE	4h
RGS2018-142	Reduce GiGe delay	Story	High	DONE	2d
RGS2018-143	Line up GiGe Camera	Story	Medium	DONE	4h
RGS2018-144	Static mesh for rear camera	Story	Medium	DONE	3h
RGS2018-145	OpenStreetMap Surface map	Story	Medium	DONE	4h
RGS2018-146	OpenStreetMap MiniMap	Story	Medium	DONE	2d
RGS2018-147	Research OSM license for commercial use	Story	Medium	DONE	4h
RGS2018-148	Create a TopView for minimap	Story	Medium	DONE	1d
RGS2018-149	Create a Birdview for minimap	Story	Medium	DONE	1d
RGS2018-150	Create a free view for the operator	Story	Medium	DONE	2d
RGS2018-151	Merge Gps/OSM/Pawn	Story	Medium	DONE	2d
RGS2018-152	Check if old GPS code/parser fits new GPS	Story	Medium	DONE	2d
RGS2018-153	Write new GPS parser	Story	Medium	DONE	2w
RGS2018-154	Create GPS log for playback	Story	Medium	DONE	2d
RGS2018-155	Remake models for GUI	Story	Medium	DONE	2h
RGS2018-156	Create GUI conditions	Story	Medium	DONE	5h
RGS2018-157 *	Update Architecture for Argos	Story	High	DONE	2d

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3d)
RGS2018-140	Convert GiGe buffer to Mp4	Story	Medium	TO DO	3d



7.2 Burn up chart

Burnup Chart

Technical sprint 7 ▾ Issue Count ⓘ How to read this chart

Work scope Scope projection Completed Work Guideline

Finalize Argos 2.0 for project ending.

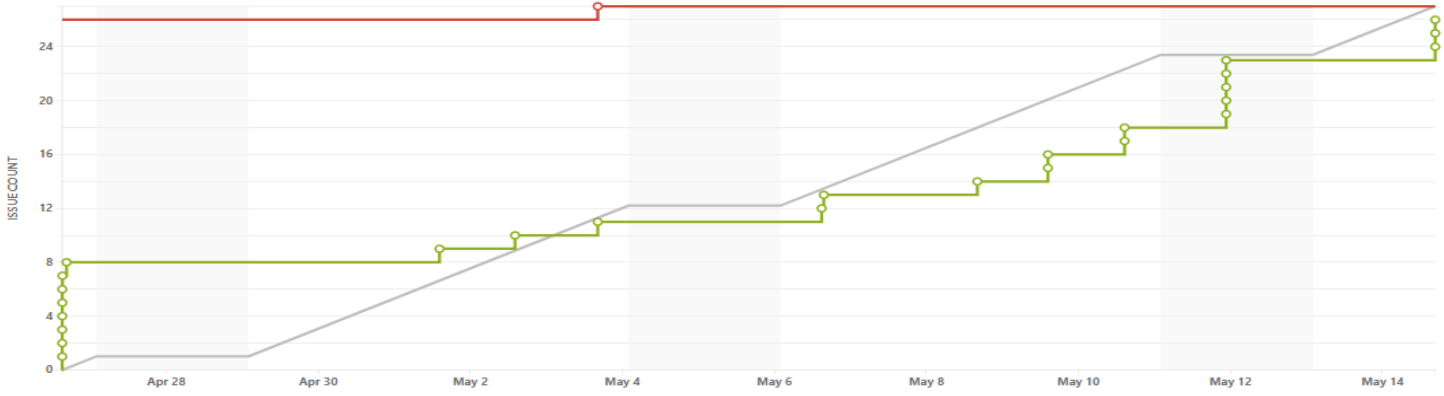


Figure 135: Burn up Chart

7.3 Velocity chart

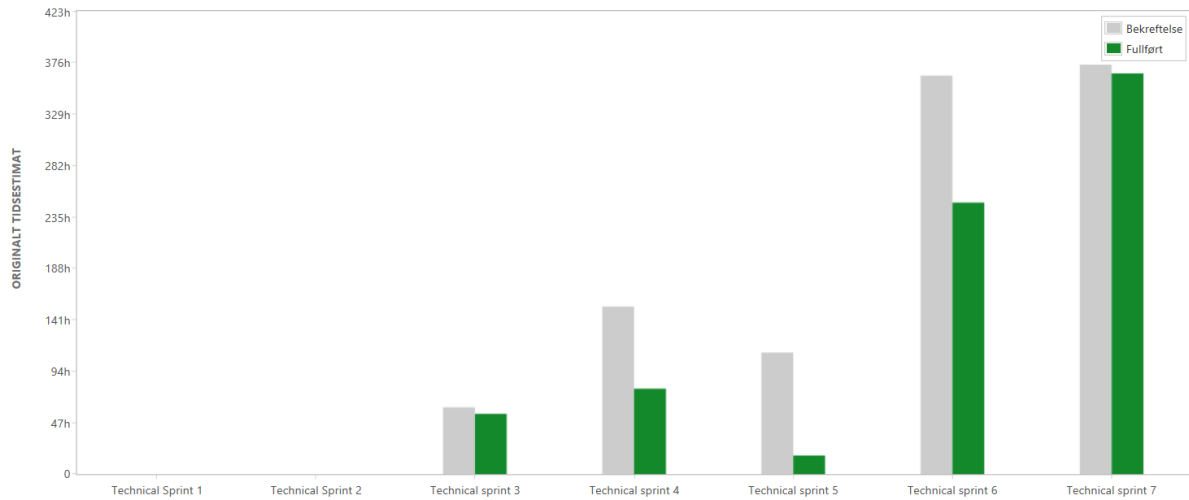


Figure 136: Velocity Chart



7.4 User workload

This sprint had a duration of 9 days because we wanted to reset our sprint days to Thursday when he had meetings with Alex.

Members:	Role:	Hours
Henrik Gjestvang	Product owner	121
Steffen A. Nielsen	Scrum master	111
Magnus E. Muri	Developer	84
Fredrik Kåsin	Developer	93
Vejbjørn Tunold	Developer	115
Andreas Holm	Developer	109

Table 95: User Workload

7.5 Uncompleted tasks

Issues Not Completed						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3d)	
RG52018-140	Convert GiGe buffer to Mp4	Story	Medium	TO DO	3d	

Convert GiGe buffer to MP4

- This story was not completed due to lack of time. The group will do its best to incorporate this last story into the Argos 2.0 application before delivery.

7.6 Conclusion

We had massive progression during this sprint. mostly because most of our long-time issues finally got resolved. We managed to finish our Epics and run our unit and integration tests. The group worked hard and efficiently and managed to finish the application. We had good dialogue with our supervisors and ensured that our project owner was satisfied with the finished product. Sprint goal was met.





Appendix – E: Epic User Stories

This appendix lists epics defined in the Argos 2.0 project. This includes each user story from each epic, and the time it each epic took to complete.

Bachelor project 2018 at
Faculty of Technology, Natural Sciences and Maritime Sciences
University of Southeast Norway
Project owner:
Kongsberg Defence Systems



KONGSBERG



1 Marking and tracking

RGS2018-122 Marking and tracking objects in UE4 worldspace

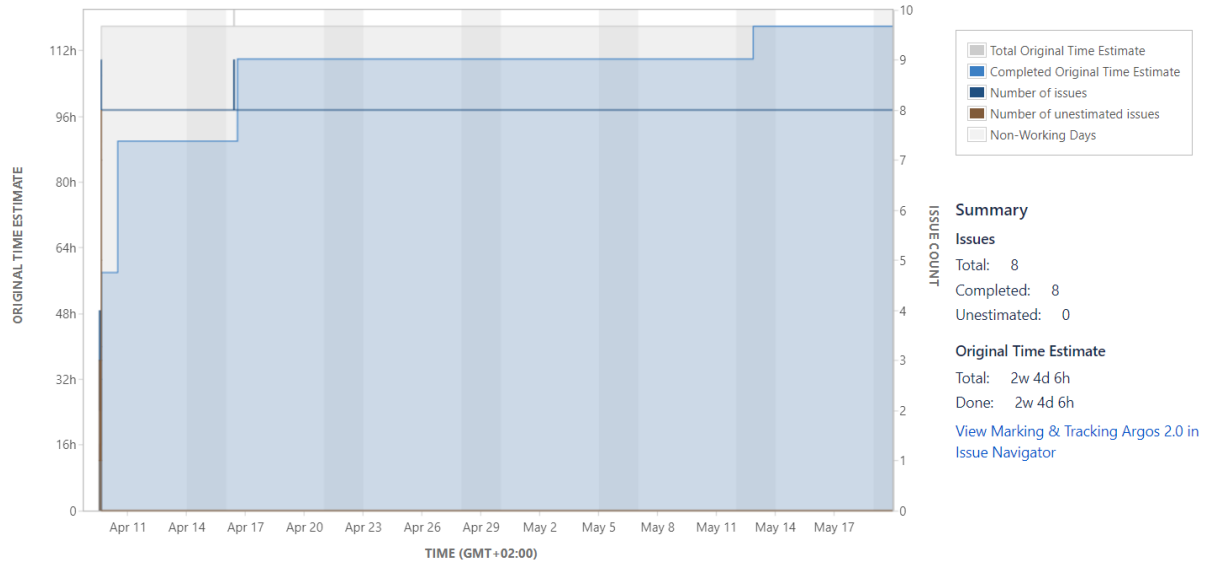


Figure 137: Epic report for Marking and Tracking

1.1 Marking and tracking user stories

Completed Issues [View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (2w 4d 6h)
RGS2018-129	Implement Marking & Tracking as a UE4 plugin	Story	Medium	DONE	1d
RGS2018-75	Distance calculation unit	Story	Medium	DONE	2d
RGS2018-123	Transform real world coordinates to cartesian coordinates	Story	Medium	DONE	4d
RGS2018-124	Bearing formula for target tracking	Story	Medium	DONE	1d
RGS2018-125	Rotation of vectors between two objects	Story	Medium	DONE	4d
RGS2018-126	Test distance in virtual world vs real world	Story	Medium	DONE	2h
RGS2018-127	Implement formulas into UE4	Story	Medium	DONE	2d
RGS2018-128	Create a coordinate class (object buffer)	Story	Medium	DONE	4h

Figure 138: User stories which makes up Marking and Tracking



2 Gige vision camera interaction

RGS2018-91 Inegrate old cameras with new argos

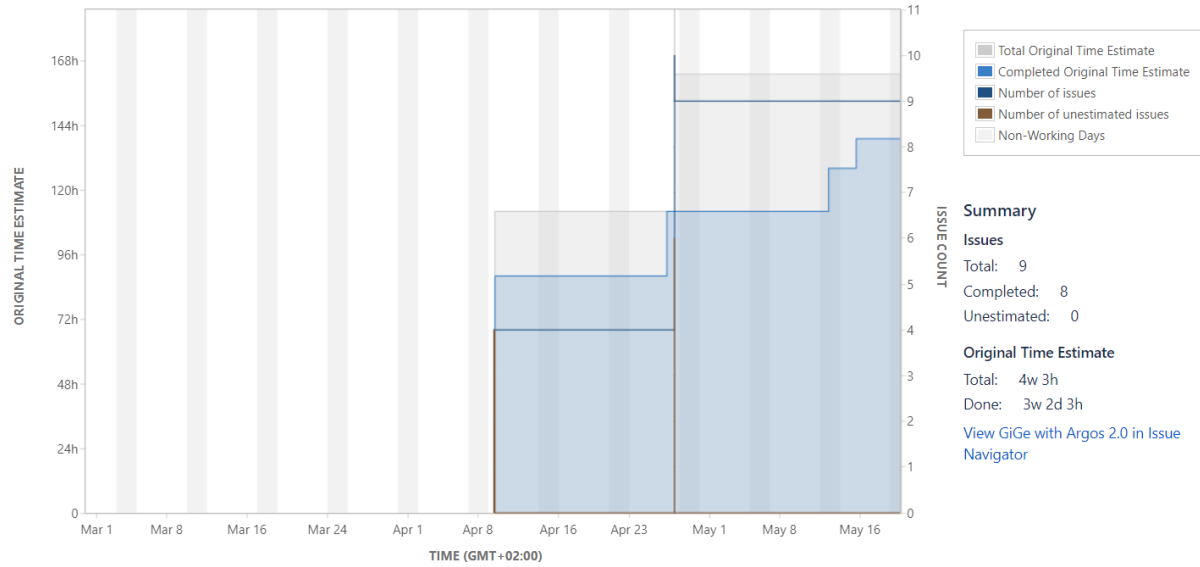


Figure 139: Epic report for GiGe vision integration

2.1 GigE Vision user stories

Completed Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3w 2d 3h)	
RGS2018-141	File directory for video files	Story	Medium	DONE	4h	
RGS2018-142	Reduce GiGe delay	Story	High	DONE	2d	
RGS2018-143	Line up GiGe Camera	Story	Medium	DONE	4h	
RGS2018-144	Static mesh for rear camera	Story	Medium	DONE	3h	
RGS2018-118	Make GiGe build with application	Story	Medium	DONE	3d	
RGS2018-119	Display feed in Argos 2.0 application	Story	Medium	DONE	3d	
RGS2018-120	Establish connection with MAKO cameras in Argos 2.0	Story	Medium	DONE	4d	
RGS2018-121	Make GiGe buffer available in Argos 2.0 application	Story	Medium	DONE	4d	

Incomplete Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3d)	
RGS2018-140	Convert GiGe buffer to Mp4	Story	Medium	TO DO	3d	

Figure 140: User stories which makes up GiGe vision integration



3 Graphical overlays

RGS2018-17 Feature

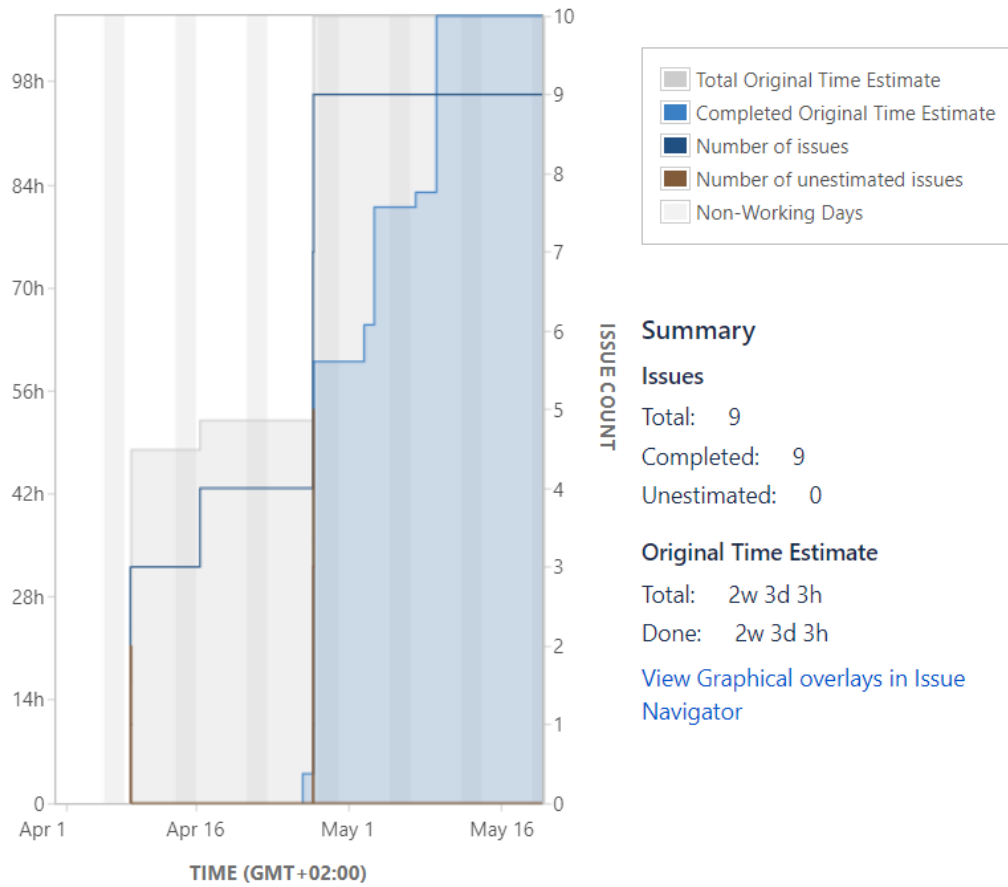


Figure 141: Epic report for Graphical Overlays

3.1 Graphical overlays user stories

Completed Issues [View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (2w 3d 3h)
RGS2018-130	Fint target in focus	Story	Medium	DONE	4h
RGS2018-155	Remake models for GUI	Story	Medium	DONE	2h
RGS2018-156	Create GUI conditions	Story	Medium	DONE	5h
RGS2018-135	Sync GUI with headset rotation	Story	Medium	DONE	2d
RGS2018-136	Argos Menu is displayed in VR	Story	Medium	DONE	2d
RGS2018-134	GUI displayed in headset	Story	Medium	DONE	2d
RGS2018-67	Object tracing using headset	Story	Medium	DONE	2d
RGS2018-108	Object GUI information test	Story	Medium	DONE	1d
RGS2018-76	GUI for position and distance information	Story	Medium	DONE	3d

Figure 142: User stories which makes up graphical overlays



4 TerraLens Implementation

RGS2018-102 Terralens implemented in UE4

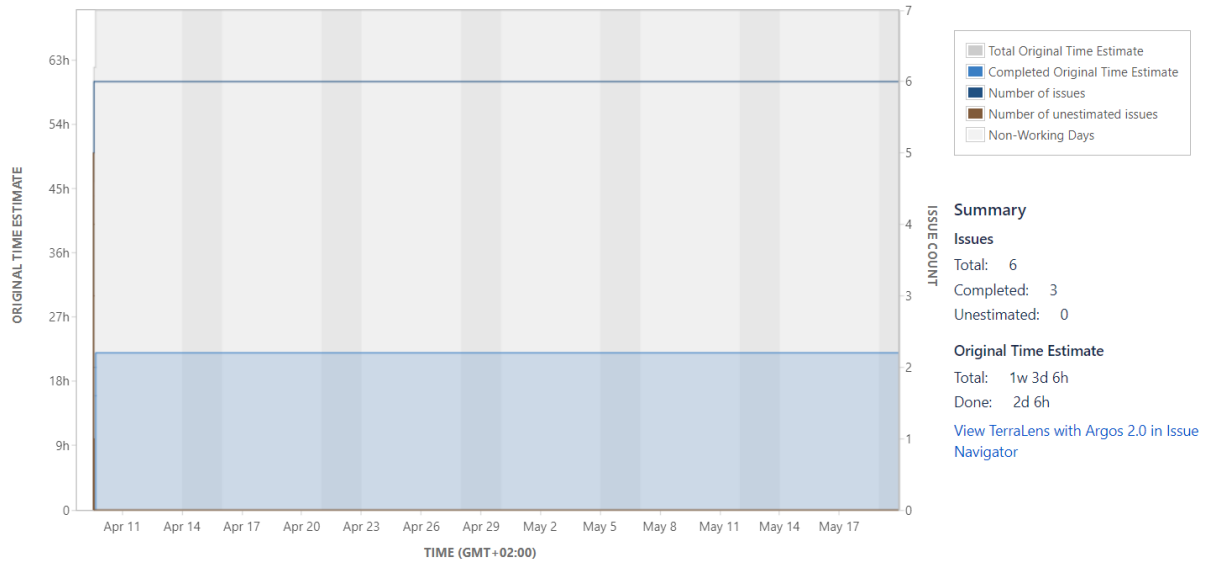


Figure 143: Epic report for TerraLens Implementation

4.1 TerraLens implementation user stories

Completed Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (2d 6h)	
RGS2018-115	Build TerraLens with UE4	Story	Medium	DONE	2d	
RGS2018-117	TerraLens (DirectX vs OpenGL)	Story	Medium	DONE	2h	
RGS2018-116	TerraLens static libraries in UE4	Story	Medium	DONE	4h	

Incomplete Issues						View in Issue Navigator
Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1w 1d)	
RGS2018-113	TerraLens as UE4 object	Story	Medium	IN PROGRESS	2d	
RGS2018-114	TerraLens connected to GPS	Story	Medium	TO DO	2d	
RGS2018-70	Place objects in TerraLens	Story	Medium	TO DO	2d	

Figure 144: User stories for TerraLens implementation



5 Conclusion

We managed to finish all our epics except for TerraLens integration, which we're stalemated because of challenges documented in the implementation chapter Tracking (TerraLens). We managed to deliver a shippable iteration from each epic, which added value to our project owner / customer.





Appendix – F: User Stories

This appendix lists the user stories as stated in Jira through the course of this bachelor project.

Bachelor project 2018 at
Faculty of Technology, Natural Sciences and Maritime Sciences
University of Southeast Norway
Project owner:
Kongsberg Defence Systems



KONGSBERG



[RGS2018-157] Update Architecture for Argos Created: 04/May/18 Updated: 04/May/18 Resolved: 04/May/18	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Sprint:	Technical sprint 7
----------------	--------------------

Description
As a supervisor, I want a new set of architecture, so that I can help the Argos group get their architecture right before the presentation.
Acceptance test: send updated architecture to supervisor

[RGS2018-156] Create GUI conditions Created: 27/Apr/18 Updated: 02/May/18 Resolved: 02/May/18	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	5 hours		
Time Spent:	Not Specified		
Original Estimate:	5 hours		

Epic Link:	Graphical overlays
Sprint:	Technical sprint 7

Description
As a developer I want to create conditions for when GUI should be displayed, so that we don't have GUI interfering with menu's etc.
Acceptance test: GUI conditions are working.

[RGS2018-155] Remake models for GUI Created: 27/Apr/18 Updated: 07/May/18 Resolved: 07/May/18	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	4 hours		
Original Estimate:	2 hours		

Epic Link:	Graphical overlays
Sprint:	Technical sprint 7

Description
As a developer I want to remake/polish some models for the Argos GUI so that they stand out in the world space.
Acceptance test: GUI is still functional in world space

[RGS2018-154] Create GPS log for playback Created: 27/Apr/18 Updated: 11/May/18 Resolved: 11/May/18	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Henrik Gjestvang
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		
Sprint:	Technical sprint 7		

Description
As a developer I want to create a GPS log in XML ,JSON or other formats, so that our system can track GPS positions when recording video.
Acceptance test: GPS creates a log when recording is set.



[RGS2018-153] [Write new GPS parser](#) Created: 27/Apr/18 Updated: 11/May/18 Resolved: 11/May/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Henrik Gjestvang
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 week		
Time Spent:	Not Specified		
Original Estimate:	2 weeks		

Sprint:	Technical sprint 7
----------------	--------------------

Description

As a developer I want to write a parser for the new GPS, so that we can use it with the old Argos GPS API.

Acceptance test: New parser parses the GPS messages

[RGS2018-151] [Merge GPS/OSM/Pawn](#) Created: 27/Apr/18 Updated: 12/May/18 Resolved: 12/May/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Sprint:	Technical sprint 7
----------------	--------------------

Description

As a developer, I want to place the Argos pawn with its actors in OpenStreetMap, bases on GPS coordinates, so that Argos virtual position = real world position.

Acceptance test: Position on virtual map = position in real life

[RGS2018-152] [Check if old GPS code/parser fits new GPS](#) Created: 27/Apr/18 Updated: 27/Apr/18 Resolved: 27/Apr/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Henrik Gjestvang
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Sprint:	Technical sprint 7
----------------	--------------------

Description

As a developer I want to see if Argos 1.0 GPS code can be used with the new GPS component, so that we can integrate the GPS into the Argos system.

Acceptance test: Know what code is usable or not from the old project

[RGS2018-150] [Create a free view for the operator](#) Created: 27/Apr/18 Updated: 12/May/18 Resolved: 12/May/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Sprint:	Technical sprint 7
----------------	--------------------

Description

As a developer I want to create a free view for the operator, which allows him to enter the 3D world space of OpenStreetMap, so that he can get virtual information about the terrain.

Acceptance test: You can move the camera in 3D space over the map.



[RGS2018-149] Create a Bird view for minimap <small>Created: 27/Apr/18 Updated: 10/May/18 Resolved: 10/May/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 day		
Time Spent:	Not Specified		
Original Estimate:	1 day		

Sprint:	Technical sprint 7
----------------	--------------------

Description
As a developer I want to create a bird-view layout for the minimap, so that the operator can choose which view he prefers.
Acceptance test: bird-view layout is created

[RGS2018-148] Create a TopView for minimap <small>Created: 27/Apr/18 Updated: 10/May/18 Resolved: 10/May/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 day		
Time Spent:	Not Specified		
Original Estimate:	1 day		

Sprint:	Technical sprint 7
----------------	--------------------

Description
As a developer I want to create a topview layout for the minimap, which gives the operator the choice of seeing the map from top-view, so that the operator can choose the view-setting he prefers.
Acceptance test: Top-view is created

[RGS2018-147] Research OSM license for commercial use <small>Created: 27/Apr/18 Updated: 27/Apr/18 Resolved: 27/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	4 hours		
Time Spent:	Not Specified		
Original Estimate:	4 hours		

Sprint:	Technical sprint 7
----------------	--------------------

Description
As a developer, I want to research OSM license to see if this is something that can use or if it has to be replaced. in the future.
Acceptance test: Write a document about your findings.

[RGS2018-146] OpenStreetMap MiniMap <small>Created: 27/Apr/18 Updated: 07/May/18 Resolved: 07/May/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Sprint:	Technical sprint 7
----------------	--------------------

Comments
Comment by Steffen Nielsen [27/Apr/18]
As a developer I want to create a minimap in the VR world, so that the operator can get information about roads, buildings and targets in the minimap.
Acceptance test: Minimap is Created



[RGS2018-145] OpenStreetMap Surface map <small>Created: 27/Apr/18 Updated: 27/Apr/18 Resolved: 27/Apr/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	4 hours		
Time Spent:	Not Specified		
Original Estimate:	4 hours		
Sprint:	Technical sprint 7		

Description

As a developer, I want to create a map surface for Argos 2.0, so that the operator can get information about streets and building in the area.

Acceptance test: Map surface is created.

[RGS2018-144] Static mesh for rear camera <small>Created: 27/Apr/18 Updated: 15/May/18 Resolved: 15/May/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	3 hours		
Time Spent:	Not Specified		
Original Estimate:	3 hours		
Epic Link:	GiGe with Argos 2.0		
Sprint:	Technical sprint 7		

Description

As a developer, I want to make a static mesh for the rear camera which follows the head rotation, so that the operator can view the rear camera from within the application with ease.

Acceptance test: Static mesh for rear camera is created

[RGS2018-143] Line up GiGe Camera <small>Created: 27/Apr/18 Updated: 15/May/18 Resolved: 15/May/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	4 hours		
Time Spent:	Not Specified		
Original Estimate:	4 hours		
Epic Link:	GiGe with Argos 2.0		
Sprint:	Technical sprint 7		

Description

As a developer I want to line up GiGe vision cameras to the static mesh, to remove as much stitching loss as possible, and give a seamless view of all four cameras on the mesh.

Acceptance test: GiGe cameras are aligned in Unreal

[RGS2018-142] Reduce GiGe delay <small>Created: 27/Apr/18 Updated: 12/May/18 Resolved: 12/May/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		

Type:	Story	Priority:	High
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Epic Link:	GiGe with Argos 2.0		
Sprint:	Technical sprint 7		

Description

As a developer, I want to reduce the GiGe camera delay we experience using UE4, so that the system can meet the requirement given from KDA.

Acceptance test: Delay reduced to the requirement from KDA



[RGS2018-141] File directory for video files <small>Created:</small>			
<small>27/Apr/18 Updated: 15/May/18 Resolved: 15/May/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	4 hours		
Time Spent:	Not Specified		
Original Estimate:	4 hours		
Epic Link:	GiGe with Argos 2.0		
Sprint:	Technical sprint 7		

Description

As a developer, I want to make a file directory for the saved video files, in order to make it easy for the video stream handler to fetch the videos.

Acceptance test: Video files are stored in the new directory

[RGS2018-140] Convert GiGe buffer to Mp4 <small>Created:</small>			
<small>27/Apr/18 Updated: 27/Apr/18</small>			
Status:	To Do		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	3 days		
Time Spent:	Not Specified		
Original Estimate:	3 days		

Epic Link:	GiGe with Argos 2.0
Sprint:	Technical sprint 7

Description

As a developer I want to convert the GiGe buffer for each camera to MP4 files, so that I can fulfill the "Record video" use case for Argos 2.0.

Acceptance test: GiGe videos are stored in MP4 format

[RGS2018-139] Made a VideoStreamHandler <small>Created:</small>			
<small>27/Apr/18 Updated: 12/May/18 Resolved: 12/May/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		


Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days, 6 hours		
Time Spent:	2 hours		
Original Estimate:	3 days		

Sprint:	Technical sprint 7
----------------	--------------------

Description

As a developer, I want to create a video stream handler that fetches movie files from a folder and places it onto the static mesh, so that the operator can navigate through video playback with ease. This handler must also be able to separate between 360 videos and Gige videos and place it the videos to the right mesh.

Acceptance test: file handler is able to find the video files

Argos Menu is displayed in VR <small>Created:</small>  [RGS2018-138] Interact with the menu <small>Created:</small>			
<small>27/Apr/18 Updated: 03/May/18 Resolved: 03/May/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 7
----------------	--------------------

Description

Be able to use the menu functionalities in VR world space

Acceptance test: Use the menu in world space



Argos Menu is displayed in VR Created: 27/Apr/18 Updated: 02/May/18 Resolved: 27/Apr/18

[RGS2018-137] Display Menu Created: 27/Apr/18 Updated: 02/May/18 Resolved: 27/Apr/18

Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Sprint:	Technical sprint 7		

Description

Be able to see the menu in VR world space

Acceptance test: look at the menu in VR

[RGS2018-136] Argos Menu is displayed in VR Created: 27/Apr/18 Updated: 03/May/18 Resolved: 03/May/18

Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Σ Remaining Estimate:	1 day	Remaining Estimate:	1 day
Σ Time Spent:	Not Specified	Time Spent:	Not Specified
Σ Original Estimate:	2 days	Original Estimate:	2 days

Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-137	Display Menu	Sub-task	Done	Fredrik Kåsin
	RGS2018-138	Interact with the menu	Sub-task	Done	Fredrik Kåsin

Epic Link: [Graphical overlays](#)

Sprint: Technical sprint 7

Description

As an operator I want to be able to interact with the main menu from the VR world, so that the operator can use the VR headset to navigate through the different options and settings.

Acceptance test: Navigate the menu in VR.

[RGS2018-135] Sync GUI with headset rotation Created: 27/Apr/18 Updated: 27/Apr/18 Resolved: 27/Apr/18

Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	Not Specified		
Original Estimate:	2 days		
Epic Link:	Graphical overlays		
Sprint:	Technical sprint 7		

Description

As a developer, I want to synchronize the rotation of GUI with the VR headset rotation, so that overlay information that should be static always traces the head rotation of the operator

Acceptance test: shake your head in VR an se that GUI is synchronized with head movement.

[RGS2018-134] GUI displayed in headset Created: 27/Apr/18 Updated: 27/Apr/18 Resolved: 27/Apr/18

Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	Not Specified		
Original Estimate:	2 days		
Epic Link:	Graphical overlays		
Sprint:	Technical sprint 7		

Description

As a developer, I want to confirm that I get the desired graphical user interface elements displayed in the VR headset we're currently using (HTC ViVE)

Acceptance test: User interface looks as desired in VR.



Find target in focus <small>26/Apr/18</small>			
[RGS2018-133] Synchronize <small>Created: 16/Apr/18 Updated: 26/Apr/18 Resolved: 26/Apr/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Sprint:	Technical sprint 6		

Find target in focus <small>16/Apr/18 Resolved: 16/Apr/18</small>			
[RGS2018-131] Implement algorithm <small>Created: 16/Apr/18 Updated: 16/Apr/18 Resolved: 16/Apr/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	3 hours		
Original Estimate:	4 hours		
Sprint:	Technical sprint 6		

Description
Write the algorithm
Acceptance test: algorithm is written

Find target in focus <small>16/Apr/18 Updated: 16/Apr/18 Resolved: 16/Apr/18</small>			
[RGS2018-132] Implement algorithm into UE4 project <small>Created: 16/Apr/18 Updated: 16/Apr/18 Resolved: 16/Apr/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 hour, 30 minutes		
Time Spent:	30 minutes		
Original Estimate:	2 hours		
Sprint:	Technical sprint 6		

[RGS2018-130] Find target in focus <small>26/Apr/18 Resolved: 26/Apr/18</small>					
Status:	Done				
Project:	Argos2018 Technical				
Component/s:	None				
Affects Version/s:	None				
Fix Version/s:	None				
Type:	Story	Priority:	Medium		
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen		
Resolution:	Done	Votes:	0		
Labels:	None				
Σ Remaining Estimate:	5 hours, 30 minutes	Remaining Estimate:	4 hours		
Σ Time Spent:	3 hours, 30 minutes	Time Spent:	Not Specified		
Σ Original Estimate:	1 day, 2 hours	Original Estimate:	4 hours		
Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-131	Implement algorithm	Sub-task	Done	Vebjørn Tunold
	RGS2018-132	Implement algorithm into UE4 project	Sub-task	Done	Vebjørn Tunold
	RGS2018-133	Synchronize	Sub-task	Done	Vebjørn Tunold
Epic Link:	Graphical overlays				
Sprint:	Technical sprint 6				

Description
As a developer I want to make a vector which measures the angle between the operators' vision and target vectors, to calculate which object the operator is looking at.
Acceptance test: The vectors can measure the angle described



[RGS2018-129] Implement Marking & Tracking as a UE4 plugin <small>Created: 09/Apr/18 Updated: 12/May/18 Resolved: 12/May/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 day		
Time Spent:	Not Specified		
Original Estimate:	1 day		
Epic Link:	Marking & Tracking Argos 2.0		
Sprint:	Technical sprint 6, Technical sprint 7		

Description

As a developer I want to implement the tracking and marking algorithms into a UE4 plugin, so that it can be used within the Argos 2.0 application

Acceptance test: making and tracking is transferred from Visual Studio to Unreal Engine

[RGS2018-127] Implement formulas into UE4 <small>Created: 09/Apr/18 Updated: 16/Apr/18 Resolved: 16/Apr/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	2 days		
Original Estimate:	2 days		
Epic Link:	Marking & Tracking Argos 2.0		
Sprint:	Technical sprint 6		

Description

As a developer I want to implement the different formulas into classes to be used in Unreal Engine, so that I can apply it to objects within the virtual world.

Acceptance test: formulas implemented into Unreal

[RGS2018-128] Create a coordinate class (object buffer) <small>Created: 09/Apr/18 Updated: 16/Apr/18 Resolved: 16/Apr/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	4 hours		
Original Estimate:	4 hours		

Epic Link: [Marking & Tracking Argos 2.0](#)

Sprint: Technical sprint 6

Description

As a developer I want to write a class that contains coordinates and other information about a given object, so that I can generate a buffer to store all objects of this type in, which will be in charge of inserting objects into the virtual world.

Acceptance test: This class is made

[RGS2018-126] Test distance in virtual world vs real world <small>Created: 09/Apr/18 Updated: 10/Apr/18 Resolved: 09/Apr/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	2 hours		
Original Estimate:	2 hours		

Epic Link: [Marking & Tracking Argos 2.0](#)

Sprint: Technical sprint 6

Description

As a developer I want to test calculation formulas, so that I can be sure that the distance between objects in virtual world equals the distance in the real world.

Acceptance test: test the formulas with known real-world positions.



[RGS2018-125] [Rotation of vectors between two objects](#)

<small>Created: 09/Apr/18 Updated: 10/Apr/18 Resolved: 10/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 day, 2 hours		
Time Spent:	6 hours		
Original Estimate:	4 days		

Epic Link:	Marking & Tracking Argos 2.0
Sprint:	Technical sprint 6

Description

As a developer I want to identify and write a formula for rotation of vectors, in order to implement a vector between two objects.

Acceptance test: formula can make two vectors rotate

[RGS2018-124] [Bearing formula for target tracking](#)

<small>Created: 09/Apr/18 Updated: 10/Apr/18 Resolved: 09/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	1 day		
Original Estimate:	1 day		

Epic Link:	Marking & Tracking Argos 2.0
Sprint:	Technical sprint 6

Description

As a developer I want to identify and create a formula for calculating bearing (navigation), so that I can have compass direction for targets.

Acceptance test: Formula can calculate bearing

[RGS2018-123] [Transform real world coordinates to cartesian coordinates](#)

<small>Created: 09/Apr/18 Updated: 10/Apr/18 Resolved: 09/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	4 days		
Original Estimate:	4 days		

Epic Link:	Marking & Tracking Argos 2.0
Sprint:	Technical sprint 6

Description

As a developer I want to make and implement a formula that lets me convert real world coordinates to cartesian coordinates, so that I can place objects at the correct position on the virtual space.

Acceptance test: Formula can convert real world coordinates to cartesian coordinates.

[RGS2018-122] [Marking and tracking objects in UE4 world space](#)

<small>Created: 09/Apr/18 Updated: 16/Apr/18</small>	
Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Epic	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Name: [Marking & Tracking Argos 2.0](#)



[RGS2018-121] Make GiGe buffer available in Argos 2.0 application <small>Created: 09/Apr/18 Updated: 09/Apr/18 Resolved: 09/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	Not Specified		
Original Estimate:	4 days		

Epic Link:	GiGe with Argos 2.0
Sprint:	Technical sprint 6

Description

Make a GiGe camera buffer interface which allow us to stream it in Argos 2.0 application in UE4.

Acceptance test: GiGe buffer is make

[RGS2018-119] Display feed in Argos 2.0 application <small>Created: 09/Apr/18 Updated: 26/Apr/18 Resolved: 26/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	2 weeks, 4 days		
Original Estimate:	3 days		

Epic Link:	GiGe with Argos 2.0
Sprint:	Technical sprint 6

Description

Stream data from the camera buffer to a surface in the Argos 2.0 application in real time.

Acceptance test: Stream data from the camera buffer to a surface in the Argos 2.0 application in real time.

[RGS2018-120] Establish connection with MAKO cameras in Argos 2.0 <small>Created: 09/Apr/18 Updated: 09/Apr/18 Resolved: 09/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	Not Specified		
Original Estimate:	4 days		

Epic Link:	GiGe with Argos 2.0
Sprint:	Technical sprint 6

Description

Establish connection with MAKO cameras in the Argos 2.0 application

Acceptance test: Connection to MAKO cameras is made

[RGS2018-118] Make GiGe build with application <small>Created: 09/Apr/18 Updated: 09/Apr/18 Resolved: 09/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	Not Specified		
Original Estimate:	3 days		
Epic Link:	GiGe with Argos 2.0		
Sprint:	Technical sprint 6		

Description

Make GiGe files, dll's & libs build with the Argos 2.0 application

Acceptance test: GiGe builds in Unreal



[RGS2018-117] TerraLens (DirectX vs OpenGL) <small>Created: 09/Apr/18 Updated: 09/Apr/18 Resolved: 09/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 hours		
Time Spent:	Not Specified		
Original Estimate:	2 hours		

Epic Link:	TerraLens with Argos 2.0
Sprint:	Technical sprint 6

Description

As a developer I want to research if TerraLens uses openGL or DirectX to make the engine implementation easier.

Acceptance test: Find out TerraLens uses openGL or DirectX

[RGS2018-116] TerraLens static libraries in UE4 <small>Created: 09/Apr/18 Updated: 09/Apr/18 Resolved: 09/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	4 hours		
Time Spent:	Not Specified		
Original Estimate:	4 hours		

Epic Link:	TerraLens with Argos 2.0
Sprint:	Technical sprint 6

Description

As a developer I want to link TerraLens libraries statically so that I don't have to include them into the solution properties to use TerraLens each time I generate a project.

Acceptance test: Link TerraLens libraries statically

[RGS2018-115] Build TerraLens with UE4 <small>Created: 09/Apr/18 Updated: 09/Apr/18 Resolved: 09/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Epic Link:	TerraLens with Argos 2.0
Sprint:	Technical sprint 6

Description

As a developer I want to be able to build TerraLens code, libraries and dll's with unreal engine, so that I use it within the engine

Acceptance test: TerraLens builds in Unreal

[RGS2018-114] TerraLens connected to GPS <small>Created: 09/Apr/18 Updated: 09/Apr/18</small>	
Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	1 day		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Epic Link:	TerraLens with Argos 2.0
Sprint:	Technical sprint 6

Description

As a developer I want to connect TerraLens with the GPS component to be able to display my position on the map.

Acceptance test: TerraLens can read the location from the GPS

[RGS2018-113] TerraLens as UE4 object <small>Created:</small>	
<small>09/Apr/18 Updated: 09/Apr/18</small>	
Status:	In Progress
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Epic Link:	TerraLens with Argos 2.0
Sprint:	Technical sprint 6

Description

As a developer I want to map TerraLens to a surface in Unreal Engine, so that I can view the map in real time within the application.

Acceptance test: TerraLens is mapped to a dynamic texture

[RGS2018-108] Object GUI information test <small>Created:</small>	
<small>09/Apr/18 Updated: 27/Apr/18 Resolved: 27/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kásin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	Not Specified		
Original Estimate:	1 day		

Epic Link:	Graphical overlays
Sprint:	Technical sprint 6, Technical sprint 7

Description

As a developer I want to place an object in the world space to check if the distance tracking & GUI overlays work properly.

Acceptance test: Place an object from C++ code

[RGS2018-102] TerraLens implemented in UE4 <small>Created:</small>	
<small>09/Apr/18 Updated: 09/Apr/18</small>	
Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Epic	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Name:	TerraLens with Argos 2.0
-------------------	--------------------------

Description

As an operator I want to have TerraLens 2D map as a graphical overlay in the Argos application, so that I can see the environment and positions ahead.

Acceptance test: You can see information from TerraLens in the VR goggles while using the application

connect GPS to Argos 2.0 <small>resolved</small>	
<small>21/Mar/18 Updated: 21/Mar/18</small>	
[RGS2018-101] TerraLens compatible <small>Created:</small>	
<small>21/Mar/18 Updated: 21/Mar/18</small>	
Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 5, Technical sprint 6
----------------	--



connect GPS to Argos 2.0 <small>connect</small> [RGS2018-100] Unreal plugin <small>Created: 21/Mar/18 Updated: 09/Apr/18</small>	
Status:	In Progress
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Henrik Gjestvang
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 5, Technical sprint 6
----------------	--

Description	Create an unreal plugin for the GPS
--------------------	-------------------------------------

Software architecture <small>connect</small> [RGS2018-98] Class diagram (View Recording) <small>Created: 01/Mar/18 Updated: 11/Mar/18 Resolved: 11/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 4
----------------	--------------------

Software architecture <small>connect</small> [RGS2018-99] Class diagram (GUI overlays) <small>Created: 01/Mar/18 Updated: 11/Mar/18 Resolved: 11/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 4
----------------	--------------------

Software architecture <small>connect</small> [RGS2018-97] Class diagram (Record Video) <small>Created: 01/Mar/18 Updated: 11/Mar/18 Resolved: 11/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 4
----------------	--------------------

Software architecture <small>argos2018</small>	
[RGS2018-96] Class diagram (Live Video) <small>Created:</small>	
<small>01/Mar/18 Updated: 11/Mar/18 Resolved: 11/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 4
----------------	--------------------

Software architecture <small>argos2018</small>	
[RGS2018-94] Class diagram (Tracking) <small>Created:</small>	
<small>01/Mar/18 Updated: 11/Mar/18 Resolved: 11/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 4
----------------	--------------------

Software architecture <small>argos2018</small>	
[RGS2018-95] Class diagram (Marking) <small>Created:</small>	
<small>01/Mar/18 Updated: 11/Mar/18 Resolved: 11/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 4
----------------	--------------------

Software architecture <small>argos2018</small>	
[RGS2018-93] Overall software architecture <small>Created:</small>	
<small>01/Mar/18 Updated: 11/Mar/18 Resolved: 11/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical sprint 4
----------------	--------------------

Description

Create a model for the software architecture containing all objects and their communication

Acceptance test: software architecture is created



[RGS2018-92] Install Visio <small>Created: 01/Mar/18 Updated: 05/Mar/18 Resolved: 05/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Highest
Reporter:	Henrik Gjestvang	Assignee:	Henrik Gjestvang
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	2 hours		
Original Estimate:	2 hours		

Sprint:	Technical sprint 4
----------------	--------------------

Description
As a developer I want to install Visio on the Argos work-laptop, so that I can fulfil the policy from KDA.
Acceptance test: Visio is working on Argos work-laptop

[RGS2018-89] Make Game Mode Blueprint as a Game Mode Class <small>Created: 29/Feb/18 Updated: 05/Mar/18 Resolved: 05/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 hour		
Time Spent:	3 hours		
Original Estimate:	4 hours		

Sprint:	Technical sprint 4
----------------	--------------------

Description
As a developer I want the Game Mode Blueprint to be cast as a Game Mode Class so that it can be manipulated from other C++ classes.
Acceptance test: Blueprint can be manipulated from C++

[RGS2018-91] Integrate old cameras with new Argos <small>Created: 01/Mar/18 Updated: 27/Apr/18</small>	
Status:	In Progress
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Epic	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Andreas Holm
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	1 day		
Original Estimate:	Not Specified		

Epic Name:	GiGe with Argos 2.0
Sprint:	Technical sprint 4, Technical sprint 5

Description
As a customer I want the Mako cameras to be operational with the new unreal engine Argos application, so that I can get the desired latency.
Acceptance: Cameras loads into Argos with desired latency

Comments
Comment by Andreas Holm <small>[07/Mar/18]</small>
Collaboration with Vebjørn!
Comment by Andreas Holm <small>[07/Mar/18]</small>
Andreas and Vebjørn collaborate.
Comment by Steffen Nielsen <small>[11/Mar/18]</small>
As a developer, I want to integrate the Mako cameras with new Argos, so that I can fulfill the latency requirement from KDA.
Acceptance test: Mako cameras integrated into Unreal



[RGS2018-90] Record and livestream video at the same time <small>Created: 01/Mar/18 Updated: 01/Mar/18 Resolved: 01/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Henrik Giestvang	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	1 day		
Original Estimate:	1 day		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As an operator I want to be able to record video while having a live preview with of the Insta 360 pro camera so that we are able to view the live footage later

Acceptance test:
Application can record and stream at the same time.

Comments

Comment by [Vebjørn Tunold](#) [01/Mar/18]

Jeg har ikke funnet en plugin, eller en annen enkel måte å lagre video fra capture card i unreal. Å lage en egen løsning for dette er potensielt svært tidkrevende.
Den enkleste løsningen er å lagre originalvideoene fra kameraet mens vi livestreamer, dette har kameraet støtte for. Ulempen er at videoen må stiches i ettertid før den kan sees som 360 video.

[RGS2018-88] Make Default Pawn Blueprint a Pawn Class <small>Created: 28/Feb/18 Updated: 05/Mar/18 Resolved: 05/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Giestvang	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 hour		
Time Spent:	3 hours		
Original Estimate:	4 hours		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a developer I want the Default Pawn Blueprint to be cast as a Pawn Class so that it can be manipulated from other C++ classes.

Acceptance test: Pawn Blueprint is made to C++

[RGS2018-87] [Make the sphere an actor](#) Created: 28/Feb/18 Updated: 07/Mar/18 Resolved: 07/Mar/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Giestvang	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	1 hour		
Original Estimate:	1 hour		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a developer I want the rendering sphere to be cast as an Actor Class so that it can be manipulated from other C++ classes.

Acceptance test: Rendering sphere is made to C++



[RGS2018-86] Fix problems with color in video stream <small>Created: 28/Feb/18 Updated: 05/Mar/18 Resolved: 05/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	3 hours, 55 minutes		
Time Spent:	5 minutes		
Original Estimate:	4 hours		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a developer I want to identify why video stream has different color settings than video samples so that they both can be optimized.

Acceptance test:
Preview quality is improved.

Comments

Comment by [Henrik Gjestvang](#) [05/Mar/18]

The fluorescent lighting in the room made the color of the video red on the capture card

[RGS2018-85] Identify problems with viewport <small>Created: 28/Feb/18 Updated: 13/Mar/18 Resolved: 13/Mar/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	6 hours		
Original Estimate:	6 hours		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a developer I want to identify why the application build show a different viewport than the application preview and fix the issue.

Acceptance test:
Application build show the same viewport for application and preview.

[RGS2018-84] Media playing features in C++ Created: 28/Feb/18 Updated: 07/Mar/18 Resolved: 07/Mar/18	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	1 day		
Original Estimate:	1 day		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a developer I want media playing features to be pushed to C++ so that we have a solid C++ foundation.

Acceptance:
Component builds and works as before, also document improvements from blueprint to C++.

Comments

Comment by [Andreas Holm](#) [07Mar18]

Features are now accessible from C++, but it is by no means finished or perfect. This task will glide into making the Mako cameras accessible in Unreal engine and will for now be seen as completed.

[RGS2018-83] [Restrict movement for camera in unreal](#) Created: 28/Feb/18 Updated: 06/Apr/18 Resolved: 06/Apr/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	1 hour		
Original Estimate:	1 hour		

Sprint:	Technical sprint 4, Technical sprint 5
----------------	--

Description

As a developer I want the Camera Object to be landlocked so that it cannot move within the world space.

Acceptance:
Camera object can't be moved from inside the application.

[RGS2018-82] [Redundancy for headset](#) Created: 28/Feb/18 Updated: 27/Apr/18 Resolved: 27/Apr/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	4 hours		
Time Spent:	Not Specified		
Original Estimate:	4 hours		

Sprint:	Technical sprint 4, Technical sprint 5, Technical sprint 7
----------------	--

Description

As a tester I want reimplement Oculus Rift so that it can be used to help debug the OSVR headset.

Also check out the options for using VIVE headset.

Acceptance:
ViVe works with Argos 2.0

[RGS2018-80] [Ignore HG](#) Created: 28/Feb/18 Updated: 05/Mar/18 Resolved: 05/Mar/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 hours		
Time Spent:	2 hours		
Original Estimate:	4 hours		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a developer I want the ignore function in TortoiseHG to be revisited so that source control only keeps track of necessary changes to the project.

Acceptance test: Mercurial ignores unnecessary files

Comments

Comment by [Andreas Holm](#) [05Mar18]

Mercurial should now ignore all intermediate files.



[RGS2018-79] Optimize lighting Created: 28/Feb/18 Updated: 02/Mar/18 Resolved: 02/Mar/18	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 hour		
Time Spent:	1 hour		
Original Estimate:	2 hours		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a developer I want all unnecessary lighting in the prototype to be removed so that the stress on the GPU is minimal.

Acceptance test: Lighting is optimized

[RGS2018-78] One to one ratio for sphere Created: 28/Feb/18 Updated: 19/Mar/18 Resolved: 19/Mar/18	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 hour		
Time Spent:	Not Specified		
Original Estimate:	1 hour		

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a developer I want the render sphere in UE to be resized to a 1 to 1 size so that it can be optimized within Unreal Engine.

Acceptance test:
Resized and optimization confirmed

[RGS2018-77] Software architecture Created: 28/Feb/18 Updated: 11/Mar/18 Resolved: 11/Mar/18	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Highest
Reporter:	Henrik Gjestvang	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Σ Remaining Estimate:	3 days	Remaining Estimate:	3 days
Σ Time Spent:	Not Specified	Time Spent:	Not Specified
Σ Original Estimate:	3 days	Original Estimate:	3 days

Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-93	Overall software architecture	Sub-task	Done	Steffen Nielsen
	RGS2018-94	Class diagram (Tracking)	Sub-task	Done	Steffen Nielsen
	RGS2018-95	Class diagram (Marking)	Sub-task	Done	Steffen Nielsen
	RGS2018-96	Class diagram (Live Video)	Sub-task	Done	Steffen Nielsen
	RGS2018-97	Class diagram (Record Video)	Sub-task	Done	Steffen Nielsen
	RGS2018-98	Class diagram (View Recording)	Sub-task	Done	Steffen Nielsen
	RGS2018-99	Class diagram (GUI overlays)	Sub-task	Done	Steffen Nielsen

Sprint:	Technical sprint 4
----------------	--------------------

Description

As a product owner I want a software architecture model of the system itself, so that the scrum team can work appropriately.

Acceptance test: Software architecture is made



[RGS2018-76] [GUI for position and distance information](#) Created: 28/Feb/18 Updated: 09/May/18 Resolved: 09/May/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Fredrik Kásin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	1 day		
Original Estimate:	3 days		

Epic Link:	Graphical overlays
Sprint:	Technical sprint 6, Technical sprint 7

Description

As a developer I want to write an interface between the object buffer and the GUI, which allow the operator to have position and distance information about objects in the GUI.

Acceptance test: Interface between object buffer and GUI is made

[RGS2018-75] [Distance calculation unit](#) Created: 28/Feb/18 Updated: 10/Apr/18 Resolved: 09/Apr/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 day		
Time Spent:	1 day		
Original Estimate:	2 days		
Epic Link:	Marking & Tracking Argos 2.0		
Sprint:	Technical sprint 4, Technical sprint 5, Technical sprint 6		

Description

As a developer I want to make a calculation unit that can calculate positions and distance between objects, so that the operator have the most accurate information available

Acceptance test: calculate the distance between two objects

[RGS2018-74] [Argos camera tracking algorithm](#) Created: 28/Feb/18 Updated: 09/Apr/18

Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Magnus Muri
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Description

As developer I want to write a recognition algorithm and apply it on the camera feed, so that the operator can track objects in real time.

Acceptance test: can track objects in camera feed in real time

[RGS2018-73] [Upload video for playback](#) Created: 28/Feb/18 Updated: 09/Apr/18

Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Description

As an operator I would like to upload a video from my computer to Argos playback so that I can view the new videos in the application.

Acceptance test: View a new video in Argos playback



[RGS2018-72] Argos menu <small>Created: 28/Feb/18 Updated: 06/Apr/18 Resolved: 06/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	4 days		
Original Estimate:	1 day		

Sprint:	Technical sprint 4, Technical sprint 5
----------------	--

Description

As a developer I want to make a menu for the Argos application, so that I can toggle between live and playback mode.

Acceptance:
Buildable Argos2.0 version with set features.

[RGS2018-71] Make compass <small>Created: 28/Feb/18 Updated: 06/Apr/18 Resolved: 06/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Fredrik Kâsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	1 day		
Time Spent:	Not Specified		
Original Estimate:	1 day		

Sprint:	Technical sprint 4, Technical sprint 5
----------------	--

Description

As a developer I want to make a compass-feature atop of the screen, so that the operator can know which direction he is driving at any given time.

Acceptance test: Compass is visible in GUI

[RGS2018-70] Place objects in TerraLens <small>Created: 28/Feb/18 Updated: 09/Apr/18</small>	
Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Steffen Nielsen
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	2 days		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Epic Link:	TerraLens with Argos 2.0
-------------------	--

Sprint:	Technical sprint 6
----------------	--------------------

Description

As a developer I want to be able to map out objects from the object buffer into TerraLens as pin's, so that the operator can get detailed information about objects in the terrain.

Acceptance test: Objects from the object buffer is marked in at the map.

[RGS2018-67] Object tracing using headset <small>Created: 28/Feb/18 Updated: 27/Apr/18 Resolved: 27/Apr/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Fredrik Kâsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	4 hours		
Time Spent:	Not Specified		
Original Estimate:	2 days		

Epic Link:	Graphical overlays
-------------------	------------------------------------

Sprint:	Technical sprint 5, Technical sprint 6, Technical sprint 7
----------------	--

Description

As an operator I want to be able to get detailed information about an object if I look at it, so that I can get a better overview of the situation I find myself in.

Acceptance test: Info about objects pop up when you look at them



[RGS2018-64] OSVR low quality and gitters Created: 28/Feb/18 Updated: 19/Mar/18 Resolved: 19/Mar/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Highest
Reporter:	Henrik Gjestvang	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	2 days		
Original Estimate:	1 day		

Sprint: Technical sprint 4

Description

As a user I want the OSVR headset to run smoothly in high quality so that I can avoid motion sickness.

Acceptance: headset doesn't jitter and the resolution is in HD quality

Comments

Comment by [Andreas Holm](#) (13/Mar/18)
Fixed the issue with the gittering.

Comment by [Andreas Holm](#) (19/Mar/18)
Problem is video feed.

[RGS2018-63] connect GPS to Argos 2.0 Created: 28/Feb/18 Updated: 13/May/18 Resolved: 13/May/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Henrik Gjestvang
Resolution:	Done	Votes:	0
Labels:	None		
Σ Remaining Estimate:	2 days	Remaining Estimate:	2 days
Σ Time Spent:	Not Specified	Time Spent:	Not Specified
Σ Original Estimate:	2 days	Original Estimate:	2 days

Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-100	Unreal plugin	Sub-task	In Progress	Henrik Gjestvang
	RGS2018-101	TerraLens compatible	Sub-task	To Do	Steffen Nielsen

Sprint: Technical sprint 5, Technical sprint 6

Description

As a developer I want to connect the GPS and the GPS.h class so that we can use TerraLens properly.

Acceptance test: GPS and TerraLens is connected

[RGS2018-62] Install graphic cards Created: 28/Feb/18 Updated: 05/Mar/18 Resolved: 05/Mar/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Henrik Gjestvang
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	3 hours		
Original Estimate:	2 hours		

Sprint: Technical sprint 4

Description

As a developer I want testing in unreal to be faster so that we can work more efficiently.

Acceptance: install the new graphic cards and check that they are working

Comments

Comment by [Henrik Gjestvang](#) (05/Mar/18)
PSU is insufficient for the graphics cards.

[Research Unreal Engine](#) (research)

[RGS2018-61] Unreal engine tracking Created: 19/Feb/18 Updated: 22/Feb/18

Status:	In Progress
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	1 day		
Time Spent:	Not Specified		
Original Estimate:	1 day		

Sprint: Technical sprint 3, Technical sprint 4, Technical sprint 5

Description

Research how unreal engine can be used in combination with map structures to be able to track virtual targets

Acceptance test: When you have an application where you can see targets on a map.



Research Unreal Engine <small>[RGS2018-60]</small>			
[RGS2018-60] Unreal engine media player <small>Created: 19/Feb/18 Updated: 24/Feb/18 Resolved: 24/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	1 day		
Original Estimate:	1 day		
Sprint:	Technical sprint 3, Technical sprint 4, Technical sprint 5		

Description

Look into how we can incorporate our 360 camera into unreal engine.

Acceptance test: having a project running with a working media object + camera

Research Unreal Engine <small>[RGS2018-59]</small>			
[RGS2018-59] Unreal engine canvas/overlay <small>Created: 19/Feb/18 Updated: 22/Feb/18 Resolved: 22/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kásin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	7 hours		
Original Estimate:	1 day		

Attachments:	compass ps.png marking.png widgetBP.png		
Sprint:	Technical sprint 3, Technical sprint 4, Technical sprint 5		

Description

Research and create some overlays in unreal engine using canvas or other elements

Comments

Comment by [Fredrik Kásin](#) [21/Feb/18]

Created compass and compass bar

Comment by Fredrik Kásin <small>[22/Feb/18]</small>
created text indicating what you are looking at

[RGS2018-58] Environment picture <small>Created: 15/Feb/18 Updated: 18/Feb/18 Resolved: 18/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	3 hours		
Original Estimate:	4 hours		
Sprint:	Technical sprint 3		

Description

As a developer I want to know what environment the product is going to be used on so that we have an easy to understand description of the system and where it could be applied

Acceptance: Make a picture of the product in environment

Comments

Comment by [Steffen Nielsen](#) [16/Feb/18]

As a project owner, I want a picture that simply explains the Argos system, to display proof of concept to business interests.

Acceptance test:

Picture is created, saved and uploaded to google drive.

[RGS2018-57] System architecture <small>Created: 15/Feb/18 Updated: 18/Feb/18 Resolved: 18/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	Not Specified		
Original Estimate:	1 day		

Sprint: Technical sprint 3

Description

As a developer I want a system architecture, so that I have a good overview over how the hardware and software in the system are going to work together

Acceptance test: System architecture is made



[RGS2018-56] [Web page](#) Created: 15/Feb/18 Updated: 22/Feb/18 Resolved: 16/Feb/18

Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	2 days		
Original Estimate:	6 hours		
Sprint:	Technical sprint 3		

Description

As a user I want an updated web page so that the school can follow the activity of the group

Acceptance: the web page it published

Comments

Comment by [Magnus Muri](#) [22/Feb/18]

webpage is published.

article Publishing Works now, and email Notification Works as well

[RGS2018-55] [Simulation for artificial combat simulation](#) Created: 15/Feb/18 Updated: 09/Apr/18

Status:	To Do		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Epic	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Epic Name:	Virtual battle space		

Description

As a project owner I want a simulation to test overlays like tracking, so that we can test the functionality without getting out in the field.

Acceptance: the simulation puts out something that we can track using the tracking function of the overlays

[RGS2018-54] [Testing 360 camera](#) Created: 15/Feb/18 Updated: 28/Feb/18 Resolved: 28/Feb/18

Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	5 hours		
Time Spent:	Not Specified		
Original Estimate:	5 hours		
Sprint:	Technical sprint 3		

Description

As a developer I want to research 360 cameras SDK so that it can be implemented in Argos.

Acceptance test: Check if the 360 camera has a SDK

[RGS2018-53] [Source control for Unreal Engine](#) Created: 15/Feb/18 Updated: 22/Feb/18 Resolved: 22/Feb/18

Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Henrik Gjestvang
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	5 hours		
Original Estimate:	3 hours		
Sprint:	Technical sprint 3		

Description

As a lead developer I want to share a base project in Unreal Engine so that all team members can start from the same template.

Acceptance: push a change to source control, and have someone else pull the change on another computer



[RGS2018-52] Research Unreal Engine <small>Created: 15/Feb/18 Updated: 21/Mar/18</small>	
Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Σ Remaining Estimate:	1 day, 4 hours	Remaining Estimate:	4 hours
Σ Time Spent:	1 day, 7 hours	Time Spent:	Not Specified
Σ Original Estimate:	1 week	Original Estimate:	2 days

Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-59	Unreal engine canvas/overlay	Sub-task	Done	Fredrik Kásin
	RGS2018-60	Unreal engine media player	Sub-task	Done	Andreas Holm
	RGS2018-61	Unreal engine tracking	Sub-task	In Progress	Magnus Muri
Sprint:	Technical sprint 3, Technical sprint 4, Technical sprint 5				

Description
As a lead developer I want all developers to research Unreal Engine so that a clear understanding of the engines functionality can be achieved.
Acceptance: test the latency of unreal engine

Comments
Comment by Steffen Nielsen <small>[15/Feb/18]</small> course for the group by the lead developer

[RGS2018-51] Install Unreal Engine <small>Created: 15/Feb/18 Updated: 19/Feb/18 Resolved: 19/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Highest
Reporter:	Henrik Gjestvang	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	2 days		
Original Estimate:	1 day		

Sprint:	Technical sprint 3
----------------	--------------------

Description
As a lead developer I want all developer machines up and running with Unreal Engine 4.18.3 so that development of a prototype can begin.
Acceptance: all computers have a working version of Unreal Engine

Comments
Comment by Steffen Nielsen <small>[15/Feb/18]</small> Install on all laptops and desktops.
Comment by Andreas Holm <small>[19/Feb/18]</small> One machine remaining: Magnus. Conflicts with visual studio 2015 persists.

[RGS2018-50] SW Architecture <small>Created: 12/Feb/18 Updated: 25/Feb/18 Resolved: 25/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	Not Specified		
Original Estimate:	1 day		
Sprint:	Technical sprint 3		

Description
As a developer I want a software architecture model so that we have a software structure
Acceptance test: Software architecture is made



[RGS2018-49] [Get Unreal engine up and running](#)
Created: 12/Feb/18 Updated: 12/Feb/18 Resolved: 12/Feb/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Sprint:	Technical Sprint 2		
Story Points:	5		

Description

As a developer, I want Unreal Engine to build on the Argos computer, so that we can have a solid foundation for the sprint when we implement Argos with Unreal.

Acceptance test: Unreal Engine Builds

[RGS2018-48] [Ebus license](#)
Created: 08/Feb/18 Updated: 21/Mar/18

Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Low
Reporter:	Henrik Gjestvang	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	2 hours		
Time Spent:	Not Specified		
Original Estimate:	2 hours		

Sprint: Technical sprint 3, Technical sprint 4, Technical sprint 5

Description

As a developer I want to know why the Ebus watermark shows up on the Argos application.

Acceptance: watermark is removed.

[Look into GPS component](#)
Created: 05/Feb/18 Updated: 09/Feb/18 Resolved: 09/Feb/18

[RGS2018-47] [Send Alex document and purchase GPS](#)
Created: 05/Feb/18 Updated: 09/Feb/18 Resolved: 09/Feb/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint: Technical Sprint 2

[Look into GPS component](#)
Created: 05/Feb/18 Updated: 09/Feb/18 Resolved: 09/Feb/18

[RGS2018-46] [Write document with GPS specs](#)
Created: 05/Feb/18 Updated: 09/Feb/18 Resolved: 09/Feb/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint: Technical Sprint 2



Get the laptop up and running Argos playback [RGS2018-45] Get Argos playback installed and running.			
<small>Created: 05/Feb/18 Updated: 12/Feb/18 Resolved: 07/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Sprint:	Technical Sprint 2		

Get the laptop up and running Argos playback [RGS2018-44] Install Drivers after clean-install			
<small>Created: 05/Feb/18 Updated: 07/Feb/18 Resolved: 07/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Sprint:	Technical Sprint 2		

Get the laptop up and running Argos playback [RGS2018-43] Clean-install laptop			
<small>Created: 05/Feb/18 Updated: 07/Feb/18 Resolved: 07/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Fredrik Kåsin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Sprint:	Technical Sprint 2		

Look into Doxygen [RGS2018-42] Teach Doxygen			
<small>Created: 05/Feb/18 Updated: 12/Feb/18 Resolved: 12/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Sprint:	Technical Sprint 2		
Description	To fellow group members.		



[Look into Doxygen](#)

[RGS2018-41] [Learn Doxygen](#) Created: 05/Feb/18 Updated: 08/Feb/18 Resolved: 08/Feb/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 2
----------------	--------------------

[Set up KDA machines](#)

[RGS2018-39] [Install VS 2017 on all machines](#) Created: 05/Feb/18 Updated: 12/Feb/18 Resolved: 08/Feb/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	High
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 2
----------------	--------------------

[Set up KDA machines](#)

[RGS2018-40] [Install Argos libraries](#) Created: 05/Feb/18 Updated: 12/Feb/18 Resolved: 08/Feb/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	High
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 2
----------------	--------------------

Description

Make sure Argos compiles

[RGS2018-38] [Housing for Insta 360](#) Created: 05/Feb/18 Updated: 09/Feb/18 Resolved: 09/Feb/18

Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Giestvang	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 2
Story Points:	7

Description

As a hardware responsible, I want a housing for the Insta 360 camera, so that it can be operated in different weather conditions and environment.

Acceptance: order/make housing for the camera and make requirement documentation(temperature, hydrophobic, fog) for the housing



[RGS2018-38] Housing for Insta 360 <small>Created: 05/Feb/18 Updated: 09/Feb/18 Resolved: 09/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 2
Story Points:	7

Description

As a hardware responsible, I want a housing for the Insta 360 camera, so that it can be operated in different weather conditions and environment.

Acceptance: order/make housing for the camera and make requirement documentation(temperature, hydrophobic, fog) for the housing

[RGS2018-37] Troubleshooting GPS drift <small>Created: 05/Feb/18 Updated: 09/Apr/18</small>	
Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Low
Reporter:	Henrik Gjestvang	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Story Points:	5
----------------------	---

Description

As a HW responsible, I want to know why the GPS starts drifting, so that we have a more stable system.

Acceptance: test and document why it is happening

[RGS2018-36] Look into GPS component <small>Created: 05/Feb/18 Updated: 12/Feb/18 Resolved: 09/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Henrik Gjestvang	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	Remaining Estimate:	Not Specified
Σ Time Spent:	Not Specified	Time Spent:	Not Specified
Σ Original Estimate:	Not Specified	Original Estimate:	Not Specified

Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-46	Write document with GPS specs	Sub-task	Done	Magnus Muri
	RGS2018-47	Send Alex document and purchase GPS	Sub-task	Done	Magnus Muri
Sprint:	Technical Sprint 2				
Story Points:	5				

Description

As a developer, I want to have a stable and redundant GPS component, so that I can track position of the vehicle.

Acceptance: send order to Alex



[RGS2018-35] Make clean install document <small>Created: 05/Feb/18 Updated: 12/Feb/18 Resolved: 12/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Fredrik Kásin
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 2
Story Points:	3

Description

As a product owner I want to have a document for how to setup a clean install for Argos on laptop, so that in case of failure I can follow it to get Argos working again.

Acceptance: Argos playback works after following this document step for step

[RGS2018-34] Get the laptop up and running Argos playback <small>Created: 05/Feb/18 Updated: 09/Feb/18 Resolved: 07/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Henrik Gjestvang	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	Remaining Estimate:	Not Specified
Σ Time Spent:	Not Specified	Time Spent:	Not Specified
Σ Original Estimate:	Not Specified	Original Estimate:	Not Specified

Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-43	Clean-install laptop	Sub-task	Done	Fredrik Kásin
	RGS2018-44	Install Drivers after clean-install	Sub-task	Done	
	RGS2018-45	Get Argos playback installed and runn...	Sub-task	Done	Vebjørn Tunold

Sprint:	Technical Sprint 2
Story Points:	5

Description

As a project promoter, I want a laptop running Argos playback, so that I can bring it to fairs and conventions.

Acceptance: Argos playback runs on laptop



[RGS2018-33] Set up KDA machines <small>Created: 05/Feb/18 Updated: 12/Feb/18 Resolved: 08/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Henrik Giestvang	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	Remaining Estimate:	Not Specified
Σ Time Spent:	Not Specified	Time Spent:	Not Specified
Σ Original Estimate:	Not Specified	Original Estimate:	Not Specified

Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-39	Install VS 2017 on all machines	Sub-task	Done	Steffen Nielsen
	RGS2018-40	Install Argos libraries	Sub-task	Done	Andreas Holm

Sprint:	Technical Sprint 2
Story Points:	3

Description

As a scrum master, I want to set up all KDA machines with the proper software, so that all members can use them for work.

Acceptance: computers are set up with, Argos code, the necessary libraries and visual studio 2017

[RGS2018-32] Look into Doxygen <small>Created: 05/Feb/18 Updated: 07/Feb/18 Resolved: 07/Feb/18</small>					
Status:	Done				
Project:	Argos2018 Technical				
Component/s:	None				
Affects Version/s:	None				
Fix Version/s:	None				
Type:	Story	Priority:	High		
Reporter:	Henrik Giestvang	Assignee:	Vebjørn Tunold		
Resolution:	Done	Votes:	0		
Labels:	None				
Σ Remaining Estimate:	Not Specified	Remaining Estimate:	Not Specified		
Σ Time Spent:	Not Specified	Time Spent:	Not Specified		
Σ Original Estimate:	Not Specified	Original Estimate:	Not Specified		
Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-41	Learn Doxygen	Sub-task	Done	Vebjørn Tunold
	RGS2018-42	Teach Doxygen	Sub-task	Done	Vebjørn Tunold
Sprint:	Technical Sprint 2				
Story Points:	4				

Description

As a project owner, I want the team to look into, and learn Doxygen, so that code can be properly documented using this software

Acceptance: use it in code

[RGS2018-31] Group pictures <small>Created: 25/Jan/18 Updated: 29/Jan/18 Resolved: 29/Jan/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Sub-task	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint: Technical Sprint 1

Description

Update pictures and names



[RGS2018-30] Source control <small>Created: 25/Jan/18 Updated: 09/Feb/18 Resolved: 09/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 1, Technical Sprint 2
Story Points:	5

Description

As a scrum master, I want a source control software, so that the scrum team can work on the same projects at the same time.

- Compare Mercurial and Git

ACCEPTANCE:
Every team member must be able to pull, commit and push to a combined dictionary

[RGS2018-29] 360 camera <small>Created: 17/Jan/18 Updated: 26/Jan/18 Resolved: 26/Jan/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Highest
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 1
Story Points:	5

Description

As a developer I want to look into the option of using a 360 camera, in order to make the stitching between cameras better, so that we get better quality

ACCEPTANCE TEST:
A document containing information and specification of an ideal camera needs to be complete.

[RGS2018-28] Update 3'rd party libraries <small>Created: 17/Jan/18 Updated: 28/Feb/18 Resolved: 28/Feb/18</small>	
--	--

Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	4 hours		
Time Spent:	Not Specified		
Original Estimate:	4 hours		
Sprint:	Technical sprint 3		
Story Points:	12		

Description

As a developer I want the 3'rd party libraries to be updated to their latest 64-bit versions in order to ensure that the end product is 64-bit exclusive application.

Acceptance test: 3'rd party library's are updated



[RGS2018-26] Remove Oculus-dependent code <small>Created: 17/Jan/18 Updated: 28/Feb/18 Resolved: 28/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	5 hours		
Time Spent:	Not Specified		
Original Estimate:	5 hours		

Sprint:	Technical sprint 3
Story Points:	4

Description

As a developer I want to remove all Oculus dependent code from the project in order to eliminate conflicts with OSVR

Acceptance test: All Oculus dependent code is removed

[RGS2018-22] OSVR SDK <small>Created: 17/Jan/18 Updated: 19/Feb/18 Resolved: 19/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	2 hours		
Original Estimate:	2 hours		
Sprint:	Technical sprint 3		
Story Points:	17		

Description

As a customer, I want OSVR to be the SDK for Argos, so that Argos can run on multiple Virtual Reality headsets.

Acceptance test: Argos 2.0 can run multiple VR headset

Comments

Comment by [Andreas Holm](#) [19/Feb/18]

OSVR working with Unreal Engine.

[RGS2018-20] Unreal engine as framework <small>Created: 17/Jan/18 Updated: 28/Feb/18 Resolved: 28/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	High
Reporter:	Steffen Nielsen	Assignee:	Andreas Holm
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	4 hours		
Original Estimate:	2 hours		

Sprint:	Technical sprint 3
Story Points:	5

Description

As a developer, I want unreal engine to be the framework, so that Argos compiles with unreal engine.

Acceptance test: Unreal Engine is set as the framework for Argos.

Comments

Comment by [Andreas Holm](#) [28/Feb/18]

Collaboration with Vebjørn!

[RGS2018-19] Guidelines <small>Created: 17/Jan/18 Updated: 12/Feb/18 Resolved: 01/Feb/18</small>			
Status:	Done		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		
Type:	Story	Priority:	Low
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
Sprint:	Technical Sprint 1		
Story Points:	1		

Description

As a customer, I want the programming language to be C++, so that existing subsystems can communicate with Argos.

ACCEPTANCE TEST:
Code must be review and approved.



[RGS2018-18] Web page update <small>Created: 17/Jan/18 Updated: 09/Feb/18 Resolved: 09/Feb/18</small>					
Status:	Done				
Project:	Argos2018 Technical				
Component/s:	None				
Affects Version/s:	None				
Fix Version/s:	None				
Type:	Story	Priority:	Medium		
Reporter:	Steffen Nielsen	Assignee:	Magnus Muri		
Resolution:	Done	Votes:	0		
Labels:	None				
Σ Remaining Estimate:	Not Specified	Remaining Estimate:	Not Specified		
Σ Time Spent:	Not Specified	Time Spent:	Not Specified		
Σ Original Estimate:	Not Specified	Original Estimate:	Not Specified		
Sub-tasks:	Key	Summary	Type	Status	Assignee
	RGS2018-31	Group pictures	Sub-task	Done	Magnus Muri
Sprint:	Technical Sprint 1				
Story Points:	3				

Description

As a customer, I want www.projectargos.org to be up-up-date so that business interests can follow Argos development and news.

ACCEPTANCE TEST:
Pictures and information about the project is included

[RGS2018-17] Feature <small>Created: 17/Jan/18 Updated: 27/Apr/18</small>			
Status:	To Do		
Project:	Argos2018 Technical		
Component/s:	None		
Affects Version/s:	None		
Fix Version/s:	None		

Type:	Epic	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Name: Graphical overlays

Description

As a customer, I want graphical overlays in the Argos application, so that GUI subsystems can be displayed in Argos.

Acceptance test: Graphical overlays is added to the application

[RGS2018-15] Guidelines <small>Created: 17/Jan/18 Updated: 12/Feb/18 Resolved: 01/Feb/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Low
Reporter:	Steffen Nielsen	Assignee:	Steffen Nielsen
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	Technical Sprint 1
Story Points:	3

Description

As a customer, I want straightforward, extendable code, so that development in the future can be done.

ACCEPTANCE TEST:
All new code must be quality controlled.

[RGS2018-14] Hardware <small>Created: 17/Jan/18 Updated: 19/Jan/18 Resolved: 19/Jan/18</small>	
Status:	Done
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Vebjørn Tunold
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Story Points:	3
----------------------	---

Description

As a salesman I want a laptop that can run OSVR HDK2 and Argos playback, so that I can show the full system on career days.

Acceptance test: Argos 2.0 runs on the Argos laptop



[RGS2018-12] [As a customer, I want the Argos application to run on Windows 10, so that it runs on a supported OS](#) Created: 17/Jan/18 Updated: 17/Jan/18

Status:	To Do
Project:	Argos2018 Technical
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Epic	Priority:	Medium
Reporter:	Steffen Nielsen	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Name:	Update Argos to Windows 10
-------------------	----------------------------

Description
Windows 7 will soon be discontinued by Windows, our customer therefore wants Argos to be updated so that it can run on windows 10.
Acceptance test: Argos is updated to Windows 10



Appendix – G: Accounting

This appendix contains the accounting for the equipment bought this far during the bachelor project. One table for what we have bought and one for equipment bought by KDA.

Describes

- Prices for the equipment
- When it was bought
- The total price for everything



Argos needs some major improvements from the previous project, not only the software but the hardware as well. We made an accounting for the equipment we have bought this far in the project, including the how many items was bought, the price per unit excluding VAT, the total VAT for the order and the total price. The products marked with grey are returned

Name	Date	Amount	Price per unit without VAT (NOK)	Total VAT (NOK)	Total price (NOK)
Navilock NL-8002U GPS-mottaker	26.02.2018	3	749,25	599,40	2 997,00
Pique	13.02.2018	7	296,00	518,00	2 590,00
Acer Predator GX-792 17,3"	24.01.2018	1	23 190,00	5 882,75	29 218,75
Insta360 Pro Spherical VR 360 8K VR-camera	12.02.2018	1	30 250,00	7 562,50	37 812,50
ASUS GeForce GTX 1060 6GB	28.02.2018	6	2 924,25	5 848,50	(23 394,00) RETURNED
Protective glass tube for camera	01.03.2018	1	611,50	152,88	764,37
Elgato Game Capture 4K 60 Pro	28.02.2018	1	3 149,25	1 049,75	4 199,00
HTC Vive pro	02.05.2018	1	7 121,25	2 373,75	9 495,00
HTC Vive Base station	02.05.2018	1	1 124,25	374,75	1 499,00
Total price					88 575,62

Table 96: Accounting for the products bought by KDA during our project



We have had a few minor expenses as well that would be impractical to have to go through KDA to buy. Underneath is a table of these things.

Item	Amount	Price total (NOK)	Paid for by	Bought at	Reason
Jira licence	1	~503,00 (62,50\$)	Everyone - Vebjørn	https://www.atlassian.com	For keeping track of scrum
HDMI adapter	1	219,00	Vebjørn	Elkjøp Kongsberg	For the 360° camera
Office supplies		317,80	Steffen, Magnus, Henrik	Brage Bokhandel Kongsberg, Europris Mjøndalen, Class Olson Kongsberg, Norli Kongsberg	Binders, staplers and other office supplies
Memory stick 8GB	2	198,00	Henrik	Brage Bokhandel Kongsberg	For handing in documentation
Mounting plate for 360° camera	1	269,00	Henrik	Asker Trelast AS Drammen	Plate for mounting the 360° camera to the roof of a car
Bolts and nuts		590,60	Henrik and Steffen	Biltema and Class Olson	Bought too short screws and had to buy a set of screws to get the right size for the attaching the camera
Cable ties	1	29,90	Henrik	Biltema Kongsberg	Fore cable management
Wireless presenter	1	499,00	Magnus	Elkjøp Gulskogen	For presentations
SD card for 360° Camera	1	799,00	Henrik	Elkjøp Kongsberg	Memory card for the 360° camera
Memory stick 128GB	1	799,00	Henrik	Power Kongsberg	Memory stick for final submission
Total:		4 224,3			

Table 97: Accounting for the products bought by the bachelor group during our project





Appendix – H: Time Sheet

This appendix contains the overview of time spent by each group member while working on this project through the sprints.

Bachelor project 2018 at
Faculty of Technology, Natural Sciences and Maritime Sciences
University of Southeast Norway
Project owner:
Kongsberg Defence Systems



KONGSBERG



Some sprints ran simultaneously and is therefore contain low values

Technical sprints

	25.01 - 01.02	05.02 - 14.02	17.02 - 28.02	01.03 - 15.03	21.03 - 05.04	09.04 - 26.04	27.04 - 14.05	
Name	Sprint 1*	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Spring 6	Sprint 7	Total
Fredrik Kåsin	2	0	31	13	23	87	93	249
Magnus Muri	5,5	34	33	10	15	79	84	260,5
Steffen Nielsen	5	29	40	33	42	84	111	344
Vebjørn Tunold	5	35	38,5	38	30,5	94,5	115	356,5
Andreas Holm	8	37	37	28	36	91	109	346
Henrik Gjestvang	4	9	34	26	20	82	121	296
Sprint sum	29,5	135	213,5	148	166,5	517,5	633	1843

Administrative Sprints

	18.01 - 25.01	26.01 - 01.02	05.02 - 16.02	08.03 - 23.03	12.05 - 21.05		
Name	Sprint 1	Sprint 2*	Sprint 3	Sprint 4	Sprint 5	Other	Total
Fredrik Kåsin	24	20	23	43	63	27	200
Magnus Muri	20	20	22	49	77	36	224
Steffen Nielsen	28	22	14	44	82	38	228
Vebjørn Tunold	22,5	25	4,5	36,5	66	25	179,5
Andreas Holm	36	27	8	23	79	31	204
Henrik Gjestvang	25	20	23	53	70	32,5	223,5
Sprint sum	155,5	134	94,5	248,5	437	189,5	822

Name	Total
Fredrik Kåsin	449
Magnus Muri	484,5
Steffen Nielsen	572
Vebjørn Tunold	536
Andreas Holm	550
Henrik Gjestvang	519,5
Sum	3111

Figure 145: Sprint time sheet diagram

Pie chart for Argos 2.0 workload

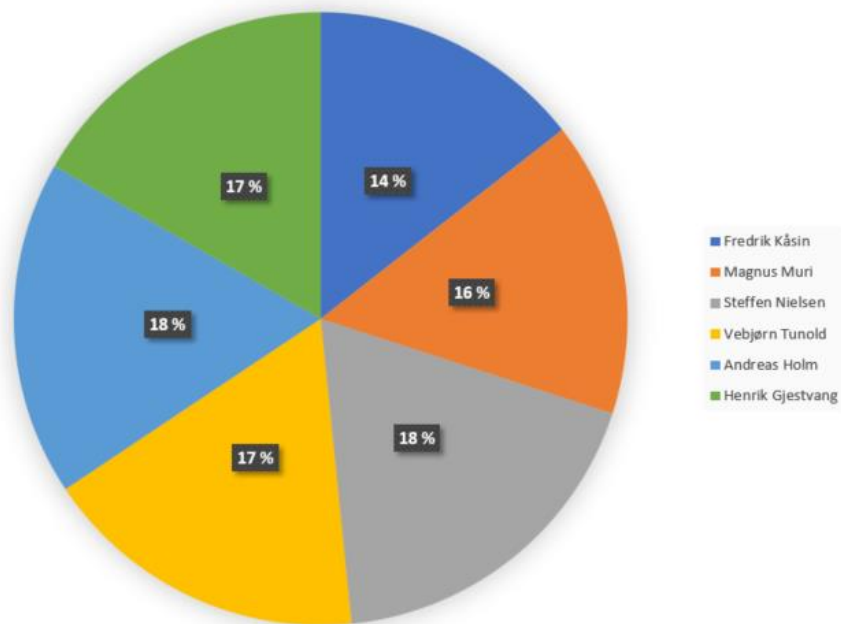


Figure 146: Pie chart for Argos 2.0 workload





Appendix – I: Glossary

This appendix contains the glossary for the whole documentation. Here all the abbreviations and terms used in the documentation is listed and explained.

Bachelor project 2018 at
Faculty of Technology, Natural Sciences and Maritime Sciences
University of Southeast Norway
Project owner:
Kongsberg Defence Systems



KONGSBERG



Phrase	Description
\$GLGSV	An NMEA standard for GLONASS messages
\$GNGGA	An NMEA standard for GNSS messages
\$GNLL	An NMEA standard for GNSS messages
\$GNRMC	An NMEA standard for GNSS messages
\$GNVTG	An NMEA standard for GNSS messages
Acceptance test	A test for the acceptance criteria of a user story
Actor	An Actor is any object that can be placed into a level.
Administrative sprint	A sprint containing only User Stories containing administrative and document work.
AFV	Armoured Fighting Vehicle
AIA	Automated Imaging Association
API	Application Programming Interface
AR	Augmented Reality
Argos	Refers both to the project and the application called Argos
Argos 2.0	Refers specifically to the version of Argos made during spring 2018
ArgosOperator	The actor representing the user of the Argos system
Async Multithreading	C++ Function, ensures safe and effective threading
BayerBG8	Color encoding for RGB
Bearing	The initial compass direction to move towards to follow the shortest path to the target. (Changes as system moves closer to the target over longer distances)
Blueprint	Blueprints is a scripting system in Unreal Engine, which utilises the idea of “Visual Scripting” or programming without writing code.
BlueprintReadWrite	Function for making a property or function available in blueprint from a C++ class
Cisco	Company that makes internet routers and network switches
Class diagram	A diagram describing
Component – architecture	A part of the total system
Component – hardware	A hardware component used in the system or that communicates with the system
Component – software	A part of the software solution of the system i.e. a class, subprogram, plugin etc.
Constraint	A requirement describing a limitation of the complete system or the project
DeckLink	High performance capture cards for computers
Doxygen	Documentation generator, a tool for writing software reference documentation
Dynamic Material Instance	Material instancing used to change the appearance of a Material without incurring an expensive recompilation of the Material
Dynamic_cast	C++ function that converts pointers and references to classes along the inheritance hierarchy.
eBus	A development kit for receiving video over GigE communication
Elgato	Capture card
Engine	A central part of a computer program
Epic - scrum	Large user story containing many smaller user stories.
Epic Games	The production company behind Unreal Engine
FPS	Frames Per Second
Functional requirement	The system requirements that describes the direct functionality of the system



GenICam	A generic programming interface for decoupling industrial cameras from their API
Geocentric Cartesian coordinates	A coordinate to a place on earth represented as X, Y, Z in meters from the center of the earth
Geodetic coordinates	A coordinate to a place on earth represented by latitude, longitude as degrees from NULL Island (west of Africa) and altitude from the mean sea level.
GigE Vision	An interface standard for high performance industrial cameras
GLONASS	Global Navigation Satellite System. Navigational system made by the Soviet Union completed in 1995
GNSS	<i>Global Navigation Satellite System</i> . An umbrella term for navigation systems with global coverage
GPS	<i>Global Positioning System</i> . Navigational system made by the United States
GUI / UI	Graphical User Interface / User Interface
GVCP	GigE Vision Control Protocol
GVSP	GigE Vision Stream Protocol
HDMI	<i>High-Definition Multimedia Interface</i>
Heading	Current compass direction of the system
HTC Vive	The VR headset used in the final prototype of Argos 2.0
HUD	Heads up display. In this document used to describe information overlay in the VR goggles
IDE	<i>Integrated Development Environment</i>
IEEE	Institute of Electrical and Electronics Engineers. Where relevant sources is found
Insta 360	The 360-degree camera used in Argos 2.0
Integration test	A test to see if the individual component functions as expected with the rest of the system
Jira	Project management tool that supports the scrum methodology.
KDA	Kongsberg Defence and Aerospace
LACP	Link Aggregation Control Protocol
LAN	<i>Local Area Network</i>
Latency	Time between an event happens in the real world until it's displayed in the VR-goggles
Level	The current part of the system that is running. A map usually contains all other software components in the system
Macro	A rule or pattern that specifies how a certain input sequence should be mapped
Main Menu	The menu that prompts at application launch. Where you decide which map that will be launched.
Mako	A low latency camera series made by allied vision that uses the GigE Vision standard
Map / Worldspace	The current part of the system that is running. A map usually contains all other software components in the system
Marking	A marker placed on a map to display a targets position from a top-down perspective
Material	An asset that can be applied to a mesh to control the visual look of the scene
Menu	A menu in world space, used for controlling and navigating within the application
Minimap	A small window in the GUI, showing a small version of a map
Naviloc	Company that makes Navigational equipment. Naviloc made the GPS receiver in ARGOS



NDI	Network Device Interface. A royalty free standard for enabling video compatible products to communicate, deliver and receive video
NIC	Network Interface Controller
NMEA	National Marine Electronics Association. Controls standards for messages passed by marine electronics components
NMEA parser	A program that splits and translates NMEA messages to useful data
Non-Functional requirement	A requirement that describes a demand for the system that doesn't impact the functionality of the system
Oculus Rift	The VR headset that the original Argos application used
Open Streetmap	An open source map application
Operator	The person using the system
OS	Operating System
OSM	Open StreetMap.
OSVR/ OSVR HDK2	An Open Source VR headset made by razer that the system was initially intended to run on
Overlay	A graphic illustration or text placed over the videofeed in the VR goggles
Pawn	A controllable actor in Unreal Engine
Pleora	Company that made GigE standard
Plugin	Software component that adds a specific feature to another software component
PoE	Power over Ethernet. Usually refers to a hardware component that gets power through the ethernet cable instead of a separate power cable
Product backlog	The list of User stories
Product owner	In Scrum, the person responsible for the product backlog
Raycast	A line that traces collisions from a start to an end position in Unreal Engine
Refresh Rate	The amount of times per second the screen is updated with a new image given in Hz
Relative Cartesian coordinates	A 3-dimensional vector from the system to the target
Requirement	A demand from the customer of the product or the developers that defines the functionality or constraints of the system
RGB	Red, Green, Blue. Used as a color standard for digital images and video
RTSP	Real Time Streaming Protocol
Scrum	Agile Project Model
Scrum Master	The person that manages the process for how information is exchanged regarding the Scrum project model
SDK	Software Development Kit
Sequence diagram	<i>Diagram</i> that shows object interactions arranged in time sequence
Sprint	A set period of time for doing a specific set of User Stories, regarding Scrum
Sprint review	Summary of what has been done and what needs to be done after a Sprint
SSD	Solid State Drive
Stakeholder	Any involved person or party of a system.
Steam VR	Library / plugin that makes the HTC Vive headset compatible with the system



STL	Standard Template Library. An international standard library of templates written for c++
Story points	Number representing the importance, size and time required to complete a User Story in Scrum.
Switch	A network switch used to communicate between the computer and cameras as well as powering the cameras using PoE
System architecture	Model of how the system is constructed
Target	A static object in the real world that is tracked by the system
Task	Technical work a development team performs to complete a part of a User Story in Scrum
Technical sprint	Sprint containing only use cases related to technical development
TerraLens	Geospatial platform proprietary software made by Kongsberg Defence and Aerospace
Test	A way to check if the functionality meets the requirements
Texture	Images that are used in Unreal Engine
Thread	The smallest sequence of instructions that can be managed independently by the OS
Tracking	Calculate an object in the real world's relative position, distance and bearing.
UDP	User Datagram Protocol. Network protocol
UFUNCTION	A macro used in Unreal to make functions blueprint accessible
UPROPERTY	A macro used in Unreal to make variables blueprint accessible and instruct the engine on what can be done using blueprints such as read/write access
UMG UI Designer	Unreal Motion Graphics UI Designer. Tool for creating GUI elements
UML	Unified Modeling Language. Standard for software related modeling
Unit test	A test for a single unit in the system to see if that unit is shippable
Unreal Engine	Game engine developed by Epic Games
Unrealifying	Translating C++ STL code to use the corresponding functions from Unreal Engine
Use case	Case of system functionality from the users perspective
User story Scrum	A set of tasks done to achieve one specific function
USN	University of Southeast Norway
VR	Virtual Reality
WGS84	World Geodetic System from 1984. The global standard for the shape and size of the earth in navigational systems
Widget	Canvas containing GUI-elements in Unreal Engine
Windows 10	Operating system made by Mircrosoft that the Argos application runs on
X64	64 bit architecture

