NTNU
Kunnskap for en bedre verden

Open source JavaScript voxelization library, with complementary applications

André Storhuag

May 2020

PROJECT / BACHELOR'S THESIS

Department of ICT and Natural Sciences

Norwegian University of Science and Technology

Supervisor 1: Ricardo Da Silva Torres

Supervisor 2: Saleh Abdel-Afou Alaliyat

Preface

This bachelor thesis is written a student from Computer Engineering at NTNU Ålesund ....

This type of technology....

What intrigued us ...

Acknowledgement

We would like to thank......

- Our mentors....
- Family and friends...
- ....

Summary and Conclusions

This report concerns the development of ....

The purpose of this project ....

Terminology

PID Proportional integral derivative controller

GUI Graphical User Interface, makes it possible to interact with a computer

API Application Programming Interface, activates functions from a remote software

TCP Transmission Control Protocol, connection oriented transmission protocol of information.

UDP User Datagram Protocol, non connection based transmission protocol of information.

IP Internet Protocol is a "best effort" delivery protocol

Notation

K_p Proportional term of a PID controller

K_i Integral term of a PID controller

K_d Derivative term of a PID controller

Kg System International unit for Kilogram

ACK acknowledge message

Abbreviations

IEEE Institute of Electronical and Electronic Engineers

I2C Inter Integrated Circuit

Gnd Ground in electronical circuits

DOF Degrees of Freedom, number of configurations for a object

List of Figures

List of Tables

Contents

Chapter 1

Introduction

This report will review different … …

1.1 Background

Norway is among …….

1.2 Problem Formulation

Would a ….

Problems to be addressed

- Design ..
- Develop …

1.3 Literature Survey

This report is based on …. published in 2016.

1.4 Objectives

The Objectives for this report thesis are:

1. Make ...
2. Implement ...
3. Create ...

1.5 Limitations

In this particular project .....

1.6 Approach

In this project the group will .....

1.7 Structure of the Report

The rest of the report is structured as follows.

Chapter 2 - Theory: Chapter two gives an introduction to the theoretical background ....

Chapter 3 - Method: Contains a description of the methodology and materials that were considered throughout the project ....

Chapter 4 - Result: Contains a description of the finished software systems

Chapter 5 - Discussion: Discusses the achieved results, the execution of methodologies and tools, in addition to encountered difficulties.

Chapter 6 - Conclusions: This chapter present an overall conclusion of the project, reviewing the objectives and teh progress made.

Chapter 2

Theoretical basis

2.1 JavaScript

Some relevant text

2.1.1 Interpreter

More text.

And a Formula:

$$V = W \times L \times D \tag{2.1}$$

and another

$$F_{buoyancy} = V \times \rho \times g \tag{2.2}$$

And a reference to a clever researchers paper [1].

2.2 Object picking

2.2.1 Ray casting

2.2.2 Color picking

2.3 Graphics

Faces, vertexes.... What is 3D models built up by? (polygons) triangles. . . Rendered by shaders (pipeline) Abstracted avay by three.js.

Chapter 3

Materials and methods

3.1 Project Organisation

The group consists of ... Meetings ....

3.2 Working methodology

3.3 Scrum

Even though i am alone, i have tried to adapt the scrum methodology.

3.4 GitFlow

3.5 Requirements sepcification

Acceptance criteria - custom field in Jira. Specially for the voxelisation algorithm.

3.6 Tools and libraries

3.7 Semantic versioning

3.7.1 JavaScript version

ES6 - needs to be transpiled with Babel.

3.7.2 Third party libraries

Why did i use them??? list up like: - xxx because ... - xxx provides ... Elaborate some on three.js

3.7.3 GitHub Actions

Why? How did i use it?

3.8 Debugging and metrics

how did i provide this?

3.9 Graphix

three-voxel-loader

3.10 Voxelization

The voxelization algorithm is mainly based on picking. More specifically, raycasting. ....

Alternatively to raycasting, one could make use of color picking. This is in principle significantly quicker than normal raycasting since it is computed on the GPU. One could render a "heightmap" from each side of the 3D model. Each pixel in the heightmaps could then be looped over, mapping the pixel color back to the face's location in space, representing a filled voxel. Even though effective, it has a severe downside. No hidden or internal structures would be detected with this sort of system. Raycasting is therefore the obvious choise, even thoug it is CPU bound.

3.10.1 Raycasting

The raycasting is supplied by three.js library. The library provides a thoughourly tested and accurate raycasting solution.

The raycasting is demonstrated in figure 3.1. A ray is directed towards an object. If it intersects a face

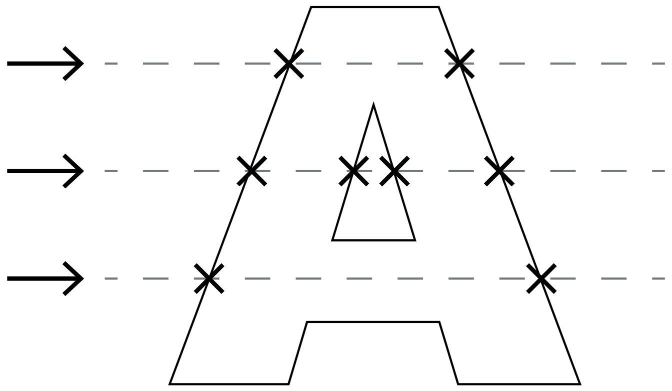


Figure 3.1: Raycasting intersections example.

Asd

3.10.2 Algorithm

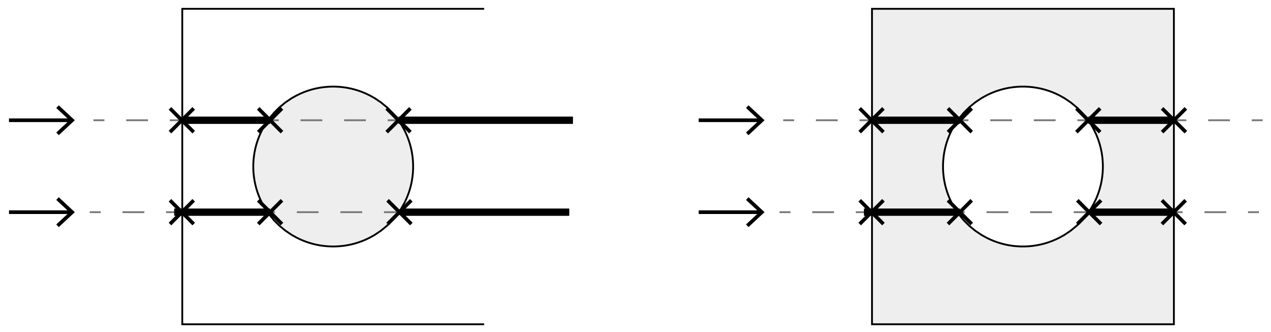
Asd

3.10.3 Shell voxelization

Asd

3.10.4 Solid voxelization

Maybe more discussion or result related? The solid voxelisation, or filled voxelisation, is achieved by interpreting the first raycast intersect as the surface of the object. From this point will everything be considered "inside" the object. When a second intersect is detected, the state is changed to be "outside" the object. A new hit would indicate "inside", and so on. This works verry well with a watertight (edges are manifolded?) 3D model, as can be seen from figure 3.2a. However, when trying to fill an object which is not watertight, this can result in severe inaccuracies. This can be seen in figure 3.2b



(a) Non-watertight 3D model cross section. (b) Watertight 3D model cross section.

Figure 3.2: Solid (voxelization) filling of 3D model cross section.

3.11 Color system

Asd

3.12 GUI

3.13 Design and modelling

This section contains a description of the alternatives considered for designing...

3.14 Plattform cross compatibility

Mainly voxelizer-desktop program....??

3.15 Building

Id fugiat et ut veniam do anim anim labore dolor eiusmod eiusmod ex sit pariatur. Sit culpa nulla do aute. Eiusmod est enim do in in velit excepteur nulla id nostrud esse ipsum ipsum exercitation. Elit excepteur sint ad exercitation eiusmod exercitation fugiat. Magna laboris proident nisi aliqua sit cillum non velit est. Id commodo in culpa est ad irure officia velit duis.

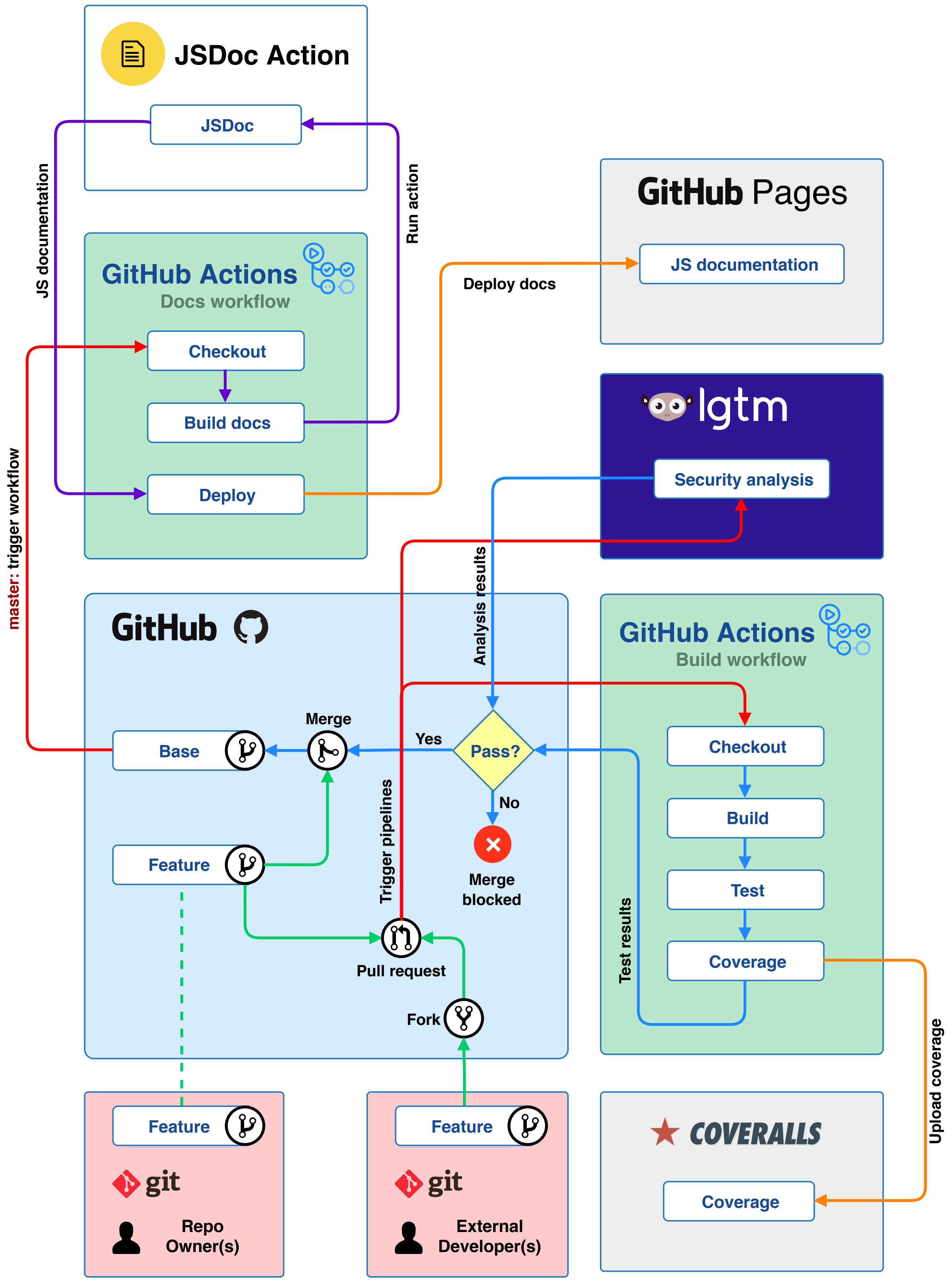


Figure 3.3: CI/CD pipelines

Tempor esse exercitation est sunt eiusmod ea occaecat laboris exercitation et in. Officia amet sit nulla sit. Cillum sint consequat labore adipisicing aliqua laborum pariatur magna tempor adipisicing mollit qui voluptate.

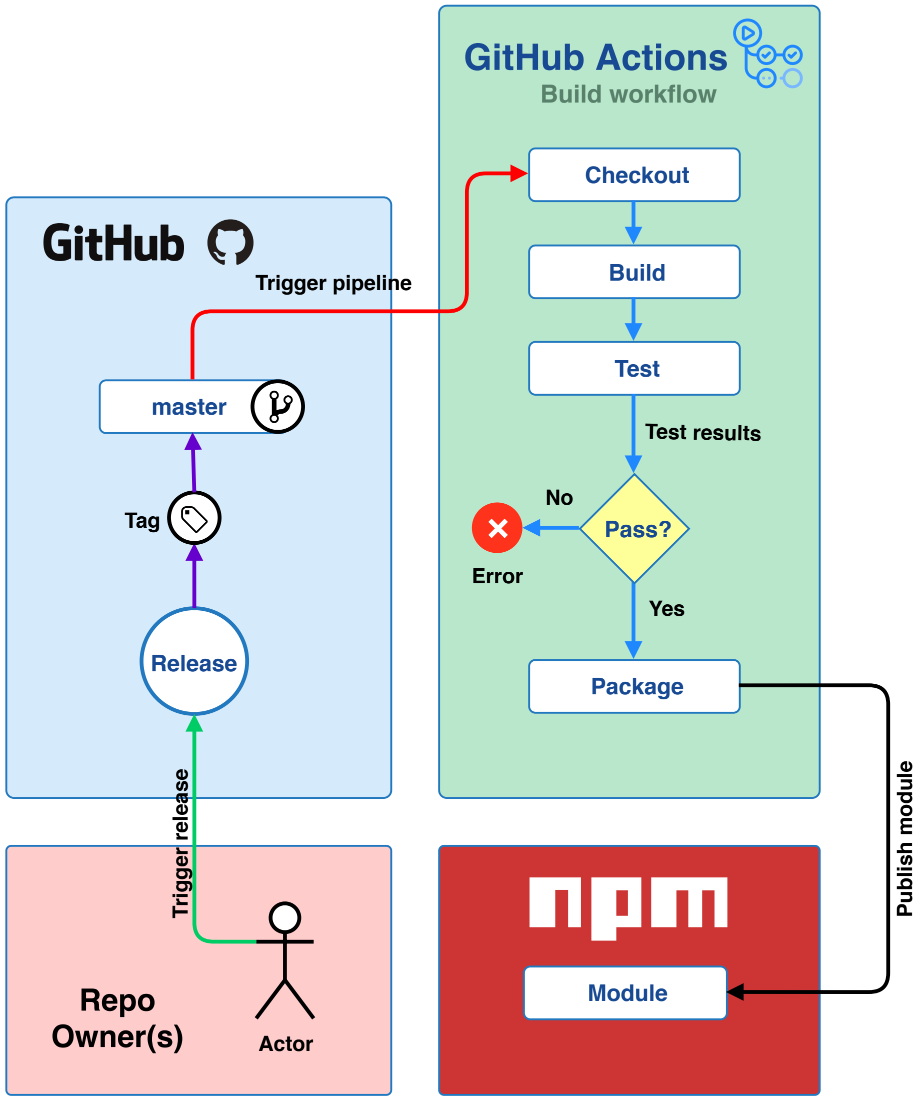


Figure 3.4: Automation of release publishing process.

Aliquip proident sint laborum proident magna ut proident. Lorem irure laboris laboris in id adipisicing eiusmod ex. Mollit culpa commodo dolore enim. Ipsum sit aliqua nostrud cupidatat. Sit magna adipisicing sit adipisicing quis in magna nulla reprehenderit est quis incididunt. Eu reprehenderit ex dolore aliquip esse mollit nulla aute officia ea sit non culpa.

Chapter 4

Result

4.1 Designing .....

This section presents a walk through of designing and building ...

4.1.1 Simulation-sketches

.....

4.2 Data collection and calculation

During the design stage......

4.3 Choosing design

...

Chapter 5

Discussion

5.1 Completness compared to requirements specification

The system is overall considered a sucess. All primary objectives are completed and

5.2 Test results

5.2.1 Result 1

The ...

5.3 Future work

5.3.1 three-voxel-loader

Performance

The three-voxel-loader plugin generates a cube buffergeometry for every voxel. Even for voxels that are not visible from outside the model. When loading a large and filled voxel model, this results in an enourmous number of faces being rendered, putting a heavy load on the hardware. A future improvement could be to only render a shell geometry based on the voxel "cubes". This would dramatically reduce the number of triangles needed to render the voxel model.

Chapter 6

Conclusions

The ....

6.1 Further work

In the .....

- Implement ....
- Expand .....
- Upgrade ......

Bibliography

[1] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* JOHN WILEY and SONS, Ltd., John Wiley and Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom, 2011.

Appendices

Appendix A

Preliminary report

Preliminary report
for bachelor's thesis

Kunnskap for en bedre verden

TITLE
Open source JavaScript voxelization library, with complementary applications

CANDIDATE			
André Storhaug			
DATE	SUBJECT CODE	SUBJECT	DOCUMENT ACCESS
31.01.2020	IE303612	Bachelor's thesis	- Open
STUDY		PAGES/ATTACHMENTS	BIBL. NR.
Computer engineering		21/0	Not used

SUPERVISOR(S)
Primary: Ricardo Da Silva Torres
Secondary: Saleh Abdel-Afou Alaliyat

Summary

This preliminary report concerns the plans and preparations for a bachelor's thesis at the Norwegian University of Science and Technology (NTNU).

The purpose of the thesis is to improve and further develop the already existing open source project named Voxelizer. Voxelizer is a JavaScript library for converting 3D models into a volumetric representation, a process known as voxelization. The library needs to be refined, professionalized and extended.

To ease the use of the library, a cross platform desktop application and a command line interface will be developed. In order to make the Voxelizer library and the complementary software easy to maintain, the projects will have a focus on automation. Especially, a GitHub action will be developed for automating the generation of JavaScript documentation.

Contents

Chapter 1

Terminology

Concepts

Voxel	Three-dimensional analogue of a pixel, representing a value on a regular grid in three-dimensional space.
Voxelization	The process of converting a 3D model into voxels.
Library	A collection of data and programming code that are used to develop software.
Cross-platform	Computer software that can be run on multiple computing platforms.

Abbreviations

MDN	Mozilla Developer Network
API	Application Programming Interface
UML	Unified Modeling Language
GUI	Graphical User Interface
CLI	Command Line Interface

Chapter 2

Introduction

This project aims to improving an already existing open source [27] library named Voxelizer [20]. It is a cross-platform library for conducting voxelization of 3D models, and is written in JavaScript.

The background for its creation was an assignment in a simulation course. The objective was to simulate diffusion using a cellular automaton. I wanted to do the simulation in the shape of a 3D object. Hence, I needed some way of constructing a volume representation out of a 3D model. Further, I also wanted to make the simulation with web technologies by making use of Three.js [21], an abstraction layer over WebGL [11].

To my surprise, I couldn't find any simple open source JavaScript solution for this. I therefore decided to make a solution myself. The result was the open source library "Voxelizer". However, due to time constraints, the current state of the library can only be considered a crude prototype. It has several issues, lacks important features and needs to be professionalized.

Alongside the library, a desktop program and a CLI interface will be developed. These will be making use of the Voxelizer library, and will greatly simplify the use of it.

Chapter 3

Project organization

3.1 Project group

Table 3.1: Students in the group

Name of student
André Storhaug

3.2 Steering group

The steering group will consist of Ricardo Da Silva Torres at NTNU in Ålesund, along with Saleh Abdel-Afou Alaliya at NTNU in Ålesund. Torres will act as supervisor and Alaliyat will be the co-supervisor.

Chapter 4

Attitudes

As a computer engineer, one is expected to behave responsible and professional. One should be curious of new technology and strive to provide the best solutions possible. Further, one should take proud in ones own work and feel a certain responsibility of the work that is done.

In a world that is becoming increasingly smaller, good collaboration is essential. In the role as a computer engineer, it is expected that one will come into contact and collaborate with persons from many different disciplines. It is therefore vital to have open mindsets and welcome new ideas. Discussions are to be expected. However, disagreements should be kept factual and handled respectfully.

The development of software is often a large part of the job as a computer engineer. Software has potential to affect human lives. Either directly or indirectly. The creation of software should therefore be rooted in strong ethics and respect for the users privacy. With this in mind, open source is a great way of achieving both transparency and openness. Today, everything revolves around profit. Companies are doing everything from charging huge amounts for proprietary software, to profiting on your personal information. However, open source has become a popular platform in which people can collaborate on software projects. By join forces and helping one another, one can achieve truly great things.

Chapter 5

Project description

5.1 Thesis problem - goals - purpose

5.1.1 Thesis problem

There exists an open source JavaScript library for conducting voxelization of 3D models. This library is called "Voxelizer". However, the library faces several issues and is lacking important features. It needs to be professionalized and made easy to both use and maintain.

5.1.2 Goals

This project has two main goals. The first goal is to improve and extend the open source Voxelizer library in such a way that it fulfills the requirements specified in the next section. The second goal is to develop a cross platform desktop application and a CLI for easy voxelization of 3D models, based the Voxelizer library.

In order to ensure maintainability of the various software projects, automation is critical. Therefore, a common subgoal will be to develop a GitHub action in order to automate documentation generation.

5.1.3 Purpose

The purpose of this project is to make it easy to conduct high quality voxelization of 3D models.

5.2 Requirements specification

The scope of this project is defined and limited by the requirements specification defined in following sections. In addition to this specification below, a backlog with user stories shall be created.

5.2.1 Voxelizer

Algorithms

The voxelisation algorithm should provide an accurate render of the original 3D model (polygon mesh [25]). The result should be geometrically representative witout distortions. No holes should be presen, unless dictated so by the given 3D model shape. Internal cavities and structures needs to be accuratly preserved. Lastly, there should be an absolute minimum of artifacts.

It should be possible to do two types of voxelization. One that is a shell voxelization, and another that is a filled volume version. The shell-type algorithm should only capture the surface of the 3D model. The filled-type algorithm needs to capture a complete volume representation of the 3D model.

One should be able to set the wanted resolution of the voxelization.

Input

The library should support a large variety of different input types. Both in terms of various file types and data structures. Support for popular file formats such as OBJ, STL and gLTF should be implemented.

Output

A diverse mixture of output types have to be supported. This includes relevant file formats and data structures. Some file formats for saving voxel data are VOX by MagicaVoxel [6], XML, BINVOX [14] and minecraft SCHEMATIC format. Relevant data structure exports include 3D arrays and octrees.

It should also be possible to export the voxelized result as normal 3D models. This could be file formats such as OBJ, STL and gLTF. Each voxel in the model should be represented as a cube.

Lastly, one could also support image export for each layer of the voxelized result. File format could for example be JPEG or PNG.

Coloring

The texture of a 3D model should carry over to surface voxels. This should be in the form of the most representative color.

Optimization

tree.js raycasting should be optimized. three.js raycasting is CPU based. It itterates each face in a 3D model, checking if the ray intersects a face or not. However, one can speed up the raycasting by employing a spatial index, for example with the help of an octree [24] or aabb tree .

5.2.2 three.js voxel loader

The three.js voxel loader module needs to be able to load voxel data into a three.js mesh [23]. The module should manage to load the voxel file formats and data structures that the voxelizer library supports exporting. This is voxel data stored in the form of a 3D array or an octree, or in a file format like VOX, XML BINVOX or SCHEMATIC.

It should be possible to customize the appearance of the loaded voxels. Both in terms of size, material and/or color.

5.2.3 Voxelizer Desktop

The Voxelizer Desktop shall be a corss-plattform [26] desktop application. It should work on both MacOS, Windows and Linux. The application should be able to voxelize a 3D models with the use of the Voxelizer library [20]. Also, it should automatically update itself when a new release of the application is published.

The application should provide intuitive GUI. It should be possible to view both the original

3D model and the voxelized result. Also, it should be possible to generate a 2D view of the cross-sections of the voxelized model.

5.2.4 Voxelizer CLI

The Voxelizer CLI should be a corss-plattform [26] CLI application. It should function on both MacOS, Windows and Linux. The application should be able to voxelize a 3D models with the use of the Voxelizer library [20].

5.2.5 JSDoc Action

The JSDoc GitHub Action should be an installable GitHub Action [9], available from the GitHub marketplace [2]. It should automate the process of generating JavaScript documentation with the help of JSDoc [12].

5.2.6 Automation

Automation should be used to ease maintenance of the various software projects. Firstly, JavaScript projects needs to have the documentation automatically generated with JSDoc [12]. Secondly, the process of publishing new versions should be automated to the greatest extent.

5.3 Methodology

The method that will be utilize in this project is the agile methodology Scrum [19]. Scrum is a very popular working methodology in the software development business. It uses an iterative and incremental approach, where each sprint gives an opportunity to improve the development process. Scrum organizes the work in sprints. This is a predefined period of time that is devoted to a set of very defined goals. The tasks to be done are often defined in a product backlog [18].

Scrum is mainly intended for teams. However, even though this is a one man project. The Scrum methodology will serve as a project framework for keeping up with progress, in addition to being able to adapt the project pace to the available working capacity. By breaking down the tasks to be done in sprints, this will help with organizing the work and steering the project in the right direction, allowing adjustmens along the way.

For this project, each sprint will be two weeks long. After each spring, a review of the completed sprint will be made. This will be an opportunity to reflect on the process, and see which goals were completed and which wasn't. Further, this review will be highly valuable for determining if adjustments should be made for the next sprint.

Scrum also seems to be a good fit because there will be a meeting with the supervisor every two weeks. By organizing the tasks to be done in two-week sprints, this will make the meetings with the supervisor more effective and relevant. New functionallity can be discussed and reviewed, in addition to planning ahead for the next two weeks.

5.4 Information gathering

The main source of information will come from various web resources. Everything from articles to code documentation will be needed for this project. The MDN Web Docs [15] will be an important source for JavaScript documentation. For Node.js related work, the Node.js API Docs [16] will be put to good use. Also, documentation from the third party library Three.js [22] will be essential. Further, Stack Overflow [17] will be a highly valued source of information due to its wast amount of questions and answers in a lot of topics.

5.5 Risk analysis

A qualitative approach will be used for assessing the risk of this project. A risk can be described as the likelihood of an event times the impact. A **MEDIUM** risk level will be accepted. The table 5.1 will be used in order to define the various risk levels.

Table 5.1: Risk level matrix.

		IMPACT			
		LOW	**MEDIUM**	**HIGH**	**VERY HIGH**
LIKELIHOOD	**VERY HIGH**	MEDIUM	HIGH	VERY HIGH	VERY HIGH
	HIGH	MEDIUM	HIGH	HIGH	VERY HIGH
	MEDIUM	LOW	MEDIUM	HIGH	HIGH
	LOW	LOW	LOW	MEDIUM	MEDIUM

In table 5.2 below, a risk assessment and risk control is conducted. The letter “L" stands for “Likelihood”, “I” for “Impact” and "R" for "Risk".

Table 5.2: Risk assessment table.

						Residual risk		
ID	**Description**	**L**	**I**	**R**	**Risk control**	**L**	**I**	**R**
R1	Services like GitHub, Jira and Confluence may go down, making various resources unavailable.	L	VH	H	Perform regular backups of important data.	L	M	M
R2	Sickness, resulting in inability to work.	M	M	M	Practicing good hygiene.	L	M	M
R3	Damaged equipment used for development.	L	VH	M	Exercise caution when handling important equipment.	L	H	M
R4	Lost or corrupt files due to system crash or failure.	M	VH	H	Perform regular backups of important data.	M	L	L
R5	Incompatibilities between technologies.	M	M	M	Properly assess the technology and plan ahead before starting development.	L	M	L
R6	Security vulnerability in package dependency.	VH	H	VH	Automatic package auditing and fixing provided by GitHub [10].	L	H	M

VH:	VERY HIGH risk
H:	HIGH risk
M:	MEDIUM risk
L:	LOW risk

The risk assessment done in table 5.2 shows that with the appropriate counter measures, all risks are reduced to a MEDIUM level. This is an acceptable level.

5.6 Primary activities in further work

Table 5.3: Main activities.

Nr	Main activity	Time/scope
A1	**Writing**	**18 weeks**
A11	Preliminary report	3 weeks
A12	Bachelor's thesis	15 weeks
A2	**voxelizer**	**7 weeks**
A21	Core improvements	1 week
A22	Algorithm improvements	2 weeks
A23	Texture support	1 week
A24	Extending 3D model file loading	1 week
A25	Extending data exporting	1 week
A26	Write tests	3 days
A27	Optimization	2 days
A3	**three-voxel-loader**	**2 weeks**
A4	**voxelizer-desktop**	**3 weeks**
A41	Core	1 week
A42	GUI	2 weeks
A5	**voxelizer-cli**	**1 week**
A6	**jsdoc-action**	**1 week**
A7	**Automation**	**1 week**

5.7 Progress plan

5.7.1 Master plan

Following is a gantt diagram 5.1 for the planned time sceduling. This includes all activities listed in section 5.6. These activities primarly include writing and software development. Activity A1 is concerned about writing the preliminary report and the thesis. Activity A2, A3, A4, A5 and A6 is concerned with the development of various software systems, where each activity is a confined project. A7 is concerned with automation of various tasks in many of the software projects.

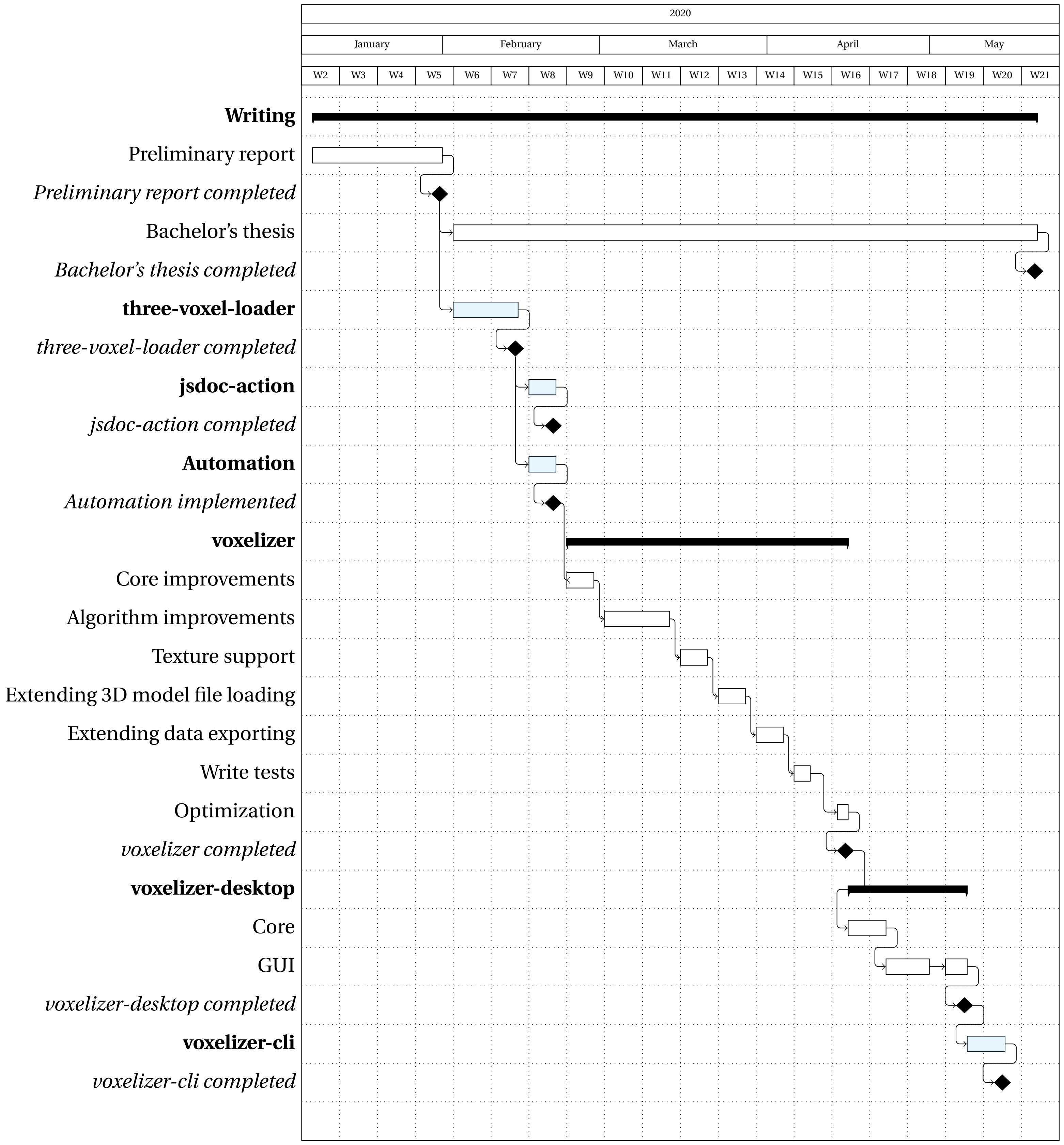


Figure 5.1: Gantt diagram of progress plan.

5.7.2 Project control assets

In order to keep the project on track, Jira [4] will be used. Jira is a project management tool developed by Atlassian, supporting a vast number of features such as issue tracking and project management. The main reason for choosing Jira over for example GitHub's solutions, is that Jira supports the agile methodology Scrum. For managing documents, minutes of meetings, UML diagrams, etc., Confluence [3] will be used.

Since this project revolves around open source projects, Jira and Confluence will only be used for internal related work. For public usage, the GitHub issue tracker and wiki will be used. Any issues, bugs or documentation of public interest shall be be placed on GitHub, instead of Jira and Confluence.

5.7.3 Development assets

For developing the various systems, the development tools listed in table 5.4 will be used.

Table 5.4: Development tools.

Development tool	**Description**
Visual Studio Code [13]	Editor for writing and debugging code.
Blender [7]	3D modeling software.
Git [8]	Version control.
GitHub [1]	Hosting of git repositories.
SourceTree [5]	Git desktop client.

5.7.4 Internal control and evaluation

At the end of each sprint, a review of the completed sprint will be conducted. A burndown chart will be generated for each sprint. This will help identifying if adjustments to the plan is nesseseary.

The requirements specification will serve as the primary criteria in order to decide wether a goal is completed or not. If a system is implemented but contains minor bugs, it will still be considered complete.

Chapter 6

Documentation

6.1 Reports and technical documents

Firstly, a progress report shall be created for every two weeks. Secondly, documentation for the various systems will be produced. In order to make a successful software library, good documentation is imperative. A lot of this documentation will mainly be automatically generated by JSDoc. The automation will be integrated in the workflow of a new version release of a system (GitHub repository). This ensures the validity of the documentation, as well as ensures future maintenance. This integration will be provided by a GitHub action, also to be a part of this project. The generated documentation will be publicly available, hosted at GitHub Pages. Lastly, various UML diagrams will be created. These will serve as illustration for the relationship between the various systems.

As mentioned in section 5.7.2, internal documentation will be kept private in Jira and Confluence. Documents of public interest will be placed publicaly available on GitHub.

Chapter 7

Planned meetings and reports

7.1 Meetings

A meeting with the advisor will be held every two weeks. These meetings will be used for reporting on the current progress. The meetings are an opportunity of gathering constructive feedback from the supervisor. Further, they will serve as documentation for working both professionally and responsible. A minutes of a meeting report will be written for every meeting.

7.2 Progress reports

Progress reports will be developed up-front of each meeting. These will describe what activities were planned, and what activities were actually seen through. If any deviations from the plan occurred during the period, these should also be included in the progress report. Further, the activities planned for the next period should also be outlined. The report will be sent to the supervisor at least a day before the meetings. This will form the basis for the matters to be discussed at the meetings.

Chapter 8

Planned deviation management

In the event of deviations from the current plans, both in terms of progress or content, several measures needs to be taken. If the deviations from the plan are of greater significance, the supervisor should be alerted. If the deviation is of lesser importance, it should be discussed with the supervisor at the regular meeting. One should then consider to change the planned approach.

Many of the planned systems builds upon one another. Therefore, if a task shows to be harder and more time consuming than first anticipated, it should consume time from tasks of lower priority. However, if a task exceeds its planned time scedule because of minor bugs, then these bugs should be properly documented and the task considered finished. These bugs should then be revisited at a later stage if there is time to spare. Since the systems are open source projects, these bugs might also be resolved by volunteers after this project is finished.

Bibliography

[1] About github, . URL https://github.com/about.

[2] Extend github, . URL https://github.com/marketplace.

[3] Atlassian. Confluence, . URL https://www.atlassian.com/software/confluence.

[4] Atlassian. Jira, . URL https://www.atlassian.com/software/jira.

[5] Atlassian. Sourcetree, . URL https://www.sourcetreeapp.com.

[6] ephtracy. Magicavoxel. URL https://ephtracy.github.io/index.html?page=mv_main.

[7] Blender Foundation. About blender. URL https://www.blender.org/about.

[8] Git. About git. URL https://git-scm.com/about.

[9] GitHub. Github actions, . URL https://github.com/features/actions.

[10] GitHub. Managing vulnerabilities in your project's dependencies, . URL https://help.github.com/en/github/managing-security-vulnerabilities/about-security-alerts-for-vulnerable-dependencies#alerts-and-automated-security-updates-for-vulnerable-dependencies.

[11] Khronos Group. Webgl overview. URL https://www.khronos.org/webgl/.

[12] JSDoc. Jsdoc. URL https://github.com/jsdoc/jsdoc.

[13] Microsoft. Visual studio code. URL https://code.visualstudio.com.

[14] Patrick Min. Binvox voxel file format specification. URL https://www.patrickmin.com/binvox/binvox.html.

[15] Mozilla. Mdn web docs. URL https://developer.mozilla.org/en-US/.

[16] Node.js. Node.js documentation. URL https://nodejs.org/api/.

[17] Stack Overflow. Stack overflow. URL https://stackoverflow.com.

[18] Scrum.org. What is a product backlog?, . URL https://www.scrum.org/resources/what-is-a-product-backlog.

[19] Scrum.org. Scrum, . URL http://www.scrum.org/.

[20] André Storhaug. Voxelizer. URL https://github.com/andstor/voxelizer.

[21] three.js. three.js, . URL https://threejs.org.

[22] three.js. three.js docs, . URL http://threejs.org/docs/.

[23] three.js. Mesh, . URL https://threejs.org/docs/#api/en/objects/Mesh.

[24] Wikipedia. Octree, . URL https://en.wikipedia.org/wiki/Octree.

[25] Wikipedia. Polygon mesh, . URL https://en.wikipedia.org/wiki/Polygon_mesh.

[26] Wikipedia. Cross-platform software, 1 2020. URL https://en.wikipedia.org/wiki/Cross-platform_software.

[27] Wikipedia. Open source, 1 2020. URL https://en.wikipedia.org/wiki/Open_source.

Appendix B

Progress reports

asd