Preliminary report
for bachelor's thesis

NTNU
Norwegian University of Science and Technology

Title			
Voxelizer: an Open Source Voxelization Engine			

Candidate				
André Storhaug				
Date	**Subject code**	**Subject**	**Document access**	
31.01.2020	IE303612	Bachelor's thesis	- Open	
Study		**Pages/Attachments**	**Bibl. nr.**	
Computer engineering		24/0	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 engine for converting 3D models into a volumetric representation, a process known as voxelization.

The engine needs to be refined, professionalized, and extended. To ease the use of the engine, a cross platform desktop application and a command line interface will be developed. In order to make the Voxelizer engine 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.

Postal address	Visitation address	Phone	Fax	Bank account
NTNU i Ålesund	Larsgårdsvegen 2	70 16 12 00	70 16 13 00	7694 05 00636
N-6025 Ålesund	**Internet**	**E-mail**		**Foretaksregisteret**
Norway	www.ntnu.no	postmottak@ntnu.no		NO 947 767 880

This assignment is an exam submission done by a student at NTNU in Ålesund.

Contents

Chapter 1

Terminology

Concepts

Cross-platform software Computer software that can be run on multiple computing platforms.

Library A collection of data and programming code that is used to develop software.

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.

Abbreviations

API Application Programming Interface

CLI Command Line Interface

GUI Graphical User Interface

MDN Mozilla Developer Network

UML Unified Modeling Language

Chapter 2

Introduction

This project aims to improving an already existing open source [1] engine named Voxelizer [2]. It is a cross-platform engine 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 [3], an abstraction layer over WebGL [4].

To the best of my knowledge, there was not any simple open source solution for this in JavaScript. I therefore decided to make a solution myself. The result was the open source project "Voxelizer". However, due to time constraints, the current state of the engine can only be considered a crude prototype. It has several issues, lacks important features and needs to be professionalized.

Alongside the engine, a desktop program and a CLI interface will be developed. These will be making use of the Voxelizer software, 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 engine for conducting voxelization of 3D models. This engine is called "Voxelizer". However, the engine 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 engine 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 engine.

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 voxelization algorithm should provide an accurate render of the original 3D model (polygon mesh [5]). The result should be geometrically representative without distortions. No holes should be present, unless dictated so by the given 3D model shape. Internal cavities and structures need to be accurately 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. It should also be possible to precisely control the wanted resolution of the voxelization.

Input

The engine 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 [7] 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 iterates 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 [8] 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 [9]. The module should manage to load the voxel file formats and data structures that the Voxelizer engine 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 cross-platform [10] 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 engine [2]. Also, it should automatically update itself when a new release of the application is published. The application should provide an intuitive GUI, it should be possible to view both the original 3D model and the voxelized result.

5.2.4 Voxelizer CLI

The Voxelizer CLI should be a cross-platform [10] CLI application. It should function on MacOS, Windows, and Linux. The application should be able to voxelize 3D models with the use of the Voxelizer engine [2].

5.2.5 JSDoc Action

The JSDoc GitHub Action should be an installable GitHub Action [11], available from the GitHub marketplace [12]. It should automate the process of generating JavaScript documentation with the help of JSDoc [13].

5.2.6 Automation

Automation should be used to ease maintenance of the various software projects. Firstly, JavaScript projects need to have the documentation automatically generated with JSDoc [13]. Secondly, the process of publishing new versions should be automated to the greatest extent.

5.3 Methodology

The method that will be utilized in this project is the agile methodology Scrum [14]. 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 [15].

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 adjustments along the way.

For this project, each sprint will be two-week long. After each spring, a review of the completed sprint will be done. This will be an opportunity to reflect on the process, and see which goals were completed and which was not. Further, this review will set the basis 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 [16] will be an important source for JavaScript documentation. For Node.js related work, the Node.js API Docs [17] will be put to good use. Also, documentation from the third party library Three.js [18] will be essential. Further, Stack Overflow [19] 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 [20].	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

Figure 5.1 presents a gantt diagram for the planned time scheduling. This includes all activities listed in section 5.6. These activities primarily include writing and software development. Activities A1 is concerned about writing the preliminary report and the thesis. Activity A2, A3, A4, A5, and A6 are 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.

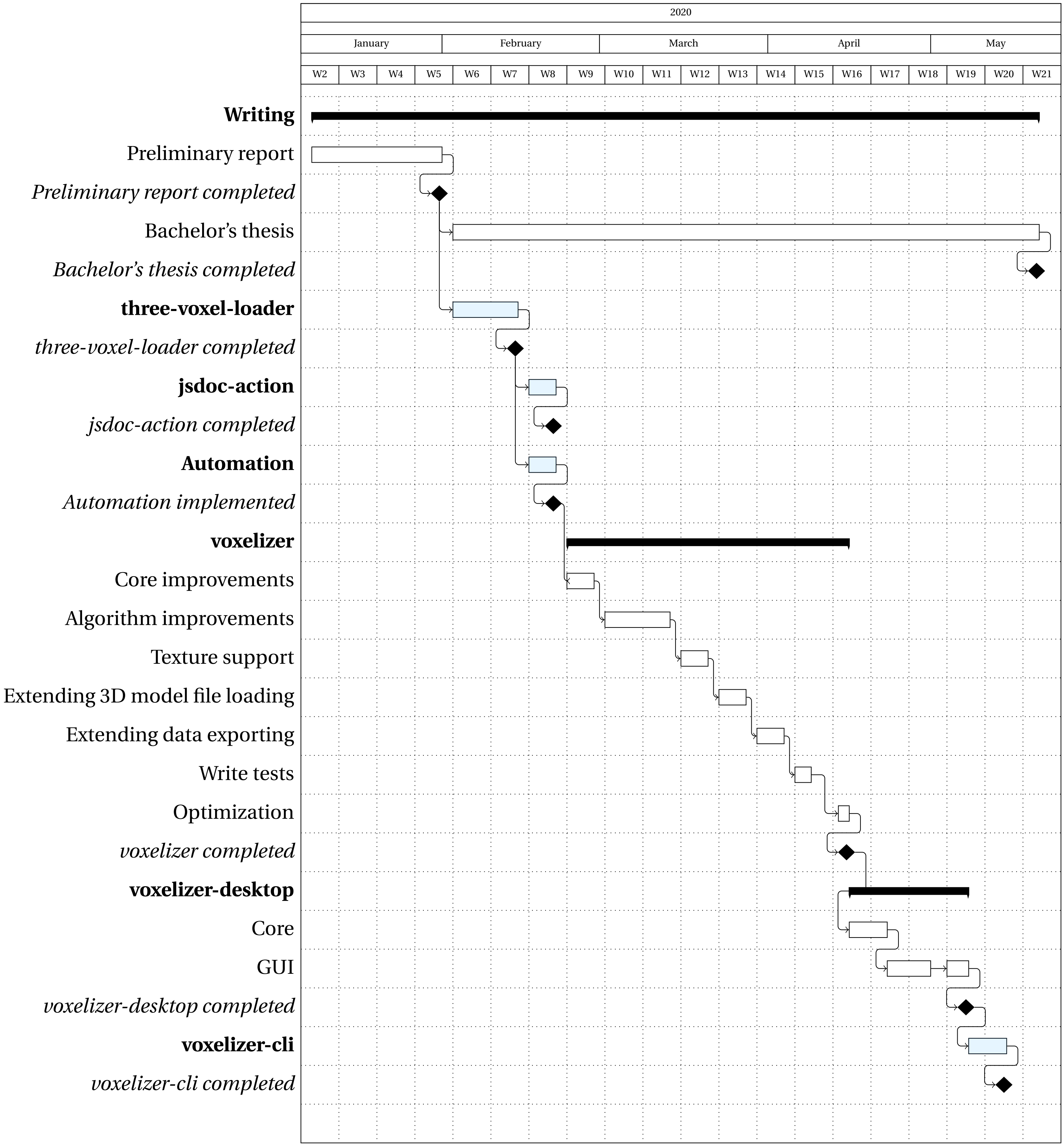

Figure 5.1: Gantt diagram of progress plan.

5.7.2 Project control assets

In order to keep the project on track, Jira [21] 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 [22] 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 [23]	Editor for writing and debugging code.
Blender [24]	3D modeling software.
Git [25]	Version control.
GitHub [26]	Hosting of git repositories.
SourceTree [27]	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 necessary.

The requirements specification will serve as the primary criteria in order to decide whether 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 project, 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 publicly 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.

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. 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 need 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 build 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 schedule 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] Wikipedia. (Jan. 2020). “Open source,” [Online]. Available: https://en.wikipedia.org/wiki/Open_source (visited on 01/26/2020).

[2] A. Storhaug. (2019). “Voxelizer,” [Online]. Available: https://github.com/andstor/voxelizer (visited on 01/26/2020).

[3] three.js. (). “Three.js,” [Online]. Available: https://threejs.org (visited on 01/27/2020).

[4] K. Group. (). “Webgl overview,” [Online]. Available: https://www.khronos.org/webgl/ (visited on 01/27/2020).

[5] Wikipedia. (). “Polygon mesh,” [Online]. Available: https://en.wikipedia.org/wiki/Polygon_mesh (visited on 01/31/2020).

[6] ephtracy. (). “Magicavoxel,” [Online]. Available: https://ephtracy.github.io/index.html?page=mv_main.

[7] P. Min. (). “Binvox voxel file format specification,” [Online]. Available: https://www.patrickmin.com/binvox/binvox.html (visited on 01/31/2020).

[8] Wikipedia. (). “Octree,” [Online]. Available: https://en.wikipedia.org/wiki/Octree (visited on 01/31/2020).

[9] three.js. (). “Mesh,” [Online]. Available: https://threejs.org/docs/#api/en/objects/Mesh (visited on 01/31/2020).

[10] Wikipedia. (Jan. 2020). “Cross-platform software,” [Online]. Available: https://en.wikipedia.org/wiki/Cross-platform_software (visited on 01/31/2020).

[11] GitHub. (). “Github actions,” [Online]. Available: https://github.com/features/actions (visited on 01/31/2020).

[12] (). “Extend github,” [Online]. Available: https://github.com/marketplace (visited on 01/31/2020).

[13] JSDoc. (). “Jsdoc,” [Online]. Available: https://github.com/jsdoc/jsdoc (visited on 01/31/2020).

[14] Scrum.org. (). “Scrum,” [Online]. Available: http://www.scrum.org/ (visited on 01/26/2020).

[15] ——, (). “What is a product backlog?” [Online]. Available: https://www.scrum.org/resources/what-is-a-product-backlog (visited on 01/31/2020).

[16] Mozilla. (). “Mdn web docs,” [Online]. Available: https://developer.mozilla.org/en-US/ (visited on 01/23/2020).

[17] Node.js. (). “Node.js documentation,” [Online]. Available: https://nodejs.org/api/ (visited on 01/23/2020).

[18] three.js. (). “Three.js docs,” [Online]. Available: http://threejs.org/docs/ (visited on 01/23/2020).

[19] S. Overflow. (). “Stack overflow,” [Online]. Available: https://stackoverflow.com (visited on 01/31/2020).

[20] GitHub. (). “Managing vulnerabilities in your project’s dependencies,” [Online]. Available: https://help.github.com/en/github/managing-security-vulnerabilities/about-security-alerts-for-vulnerable-dependencies#alerts-and-automated-security-updates-for-vulnerable-dependencies (visited on 01/26/2020).

[21] Atlassian. (). “Jira,” [Online]. Available: https://www.atlassian.com/software/jira (visited on 01/25/2020).

[22] ——, (). “Confluence,” [Online]. Available: https://www.atlassian.com/software/confluence (visited on 01/25/2020).

[23] Microsoft. (). “Visual studio code,” [Online]. Available: https://code.visualstudio.com (visited on 01/25/2020).

[24] B. Foundation. (). “About blender,” [Online]. Available: https://www.blender.org/about (visited on 01/25/2020).

[25] Git. (). "About git," [Online]. Available: https://git-scm.com/about (visited on 01/25/2020).

[26] (). "About github," [Online]. Available: https://github.com/about (visited on 01/25/2020).

[27] Atlassian. (). "Sourcetree," [Online]. Available: https://www.sourcetreeapp.com (visited on 01/25/2020).