Real-time Weather Rendering System

Requirement specification

BSc Thesis
The goal of this thesis is to research and implement a real-time, procedural, weather rendering system. The following document reveals the process of creating such a system from both a technical and mathematical perspective, considering different algorithms for techniques like noise generation and raymarching.

Field of Studies:	BSc in Computer Science
Specialization:	Computer perception and virtual reality
Author:	Matthias Thomann
Supervisor:	Prof. Urs Künzler
Date:	March 15, 2021
Version:	0.1

Contents

1 General

1.1 Purpose

This document serves the purpose of defining and clarifying the goals, which the thesis 'Realtime Weather Rendering System' is supposed to achieve. Furthermore, the requirement specification allows for a more accurate evaluation of the achievement of objectives and of the result itself.

1.2 Revision History

Version	Date	Name	Comment
0.1	February 27, 2021	Matthias Thomann	Initial draft
0.2	March 03, 2021	Matthias Thomann	Updated project schedule
0.3	March 11, 2021	Matthias Thomann	Determined requirements
0.4	March 13, 2021	Matthias Thomann	Reworked vision chapter
0.5	March 14, 2021	Matthias Thomann	Added system overview diagram
0.6	March 15, 2021	Matthias Thomann	Added UI mockups

2 Vision

2.1 Weather Rendering System

This section defines a high-level vision for the desired outcome of this thesis and potential future work. As listed in the primary goals, the weather rendering system will be based on compute shaders. Compared to the prototype from the previous project, this is expected to result in a much better performance. That in turn, allows for a more complex and realistic model.
With the incorporation of real-time weather data and the use of topological landscape data, any given weather scenario could be simulated and rendered. The desired outcome ideally looks similar to the image depicted in Figure 1. A rendered version of such a cloud system can look elusively realistic compared to an actual photograph, like in Figure 2.

Figure 1: A rendered image of volumetric clouds [1].

Figure 2: A photographic reference of clouds [2].

The first thought about the practical use of a fully-fledged volumetric cloud system might be a video game, since clouds are often a significant part of outdoor scenery in games. However, for this thesis it is intended that the knowledge and results acquired during the given period will be used to recreate a lifelike weather system instead.

2.2 System Overview

To get a better understanding of how such a system could be implemented, this diagram shows all the involved components and their processes.

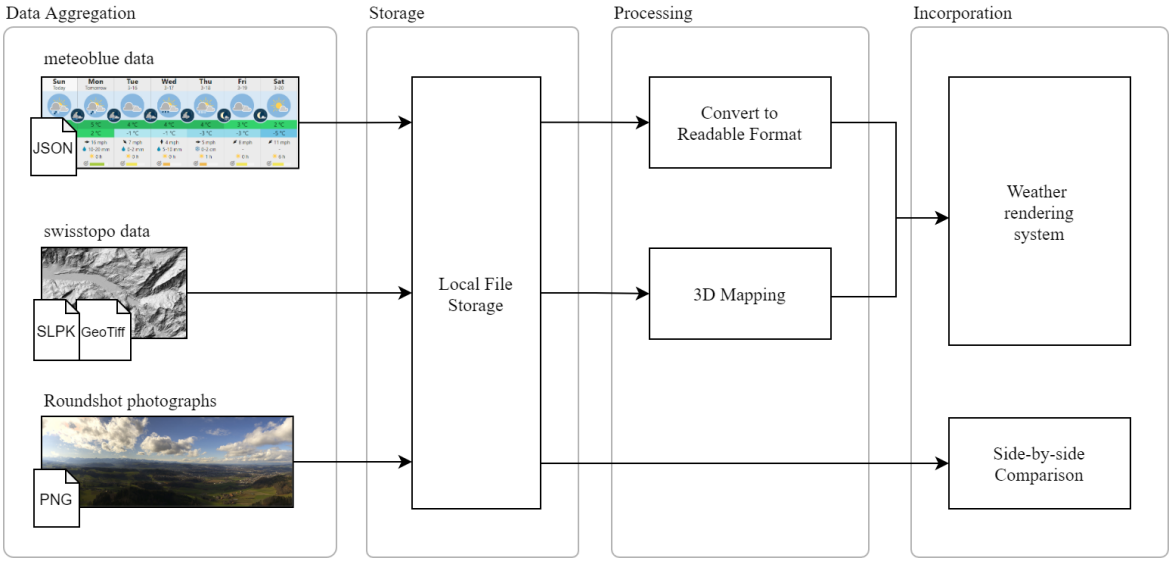


Figure 3: System overview diagram.

All external data is retrieved regularly and stored on the local file system. Their respective file formats are denoted in the bottom left corner of each data source. After aggregation, the data is processed into a readable and compatible format for the Unity Engine. From there, the weather rendering system can make unrestricted use of the data. The resulting output of the system can then be compared side-by-side with the collected real-life photographs.

2.3 External Data

2.3.1 Meteoblue Weather Data

To accurately reflect a weather system, conditions like precipitation, wind and cloudiness will be considered. Fortunately, the company *meteoblue* offers this data in form of different data packages [3]. As an additional bonus, the license costs are drastically reduced for student projects and educational work.
From all available data packages, the "basic_1h" [4] offer seems the most fitting for this thesis. It includes the most common weather variables only, but this will clearly suffice for the planned project. Some of the crucial variables are wind speed, wind direction, temperature, sea level pressure, and a pictocode.
The weather data will be requested for the following locations:

- Bern, Switzerland
- Fribourg, Switzerland
  (to account for the weather in the background of the photographs)
- Solothurn, Switzerland
  (to account for the weather in the background of the photographs)

This data is retrieved on a daily basis and stored on a local file system for the duration of the thesis. The file format is Java-Script Object Notation (JSON).

2.3.2 Roundshot Photographs

The weather data from *meteoblue* gives detailed information about the weather at a specific time and date. But to be able to compare the rendered result of the weather system with the actual weather of that period, real photographs of the same time should be used. That is why images taken by the *Roundshot* camera system from the company *Seitz* [5] are stored periodically.
There are many installations of those systems across the country. For this project, the following two locations are used:

- Roundshot camera Bantiger, Switzerland [6]
- Roundshot camera Gurtenpark, Switzerland [7]

This data is retrieved on a weekly basis and stored on a local file system for the duration of the thesis. The file format is Portable Network Graphic (PNG).

2.3.3 Swisstopo Height Models

The last part of a convincing weather rendering system is the landscape. For that, the topologically accurate 3D height model data from the company *swisstopo* will be used [8]. As of March 2021, *swisstopo*'s height models and landscape data are available free of charge [9]. The goal is to download and convert this data into a Unity-compatible 3D model and use it as a base for the scenery.
This data is retrieved once or whenever an update is due and stored on a local file system for the duration of the thesis. The file format is ESRI Scene Layer Package (SLPK) or any othersuitable data format by the company ESRI.

2.4 UI Mockup

As for the User Interface (UI), there will be two possible modes to choose from. The first one is a "real-life" mode that lets the user choose time and date, for which the weather is then recreated with the *meteoblue* data from that period. The other mode will be a sandbox mode, where all influencial weather variables can be controlled manually.
The following mockups only serve as a general guideline and are not final.

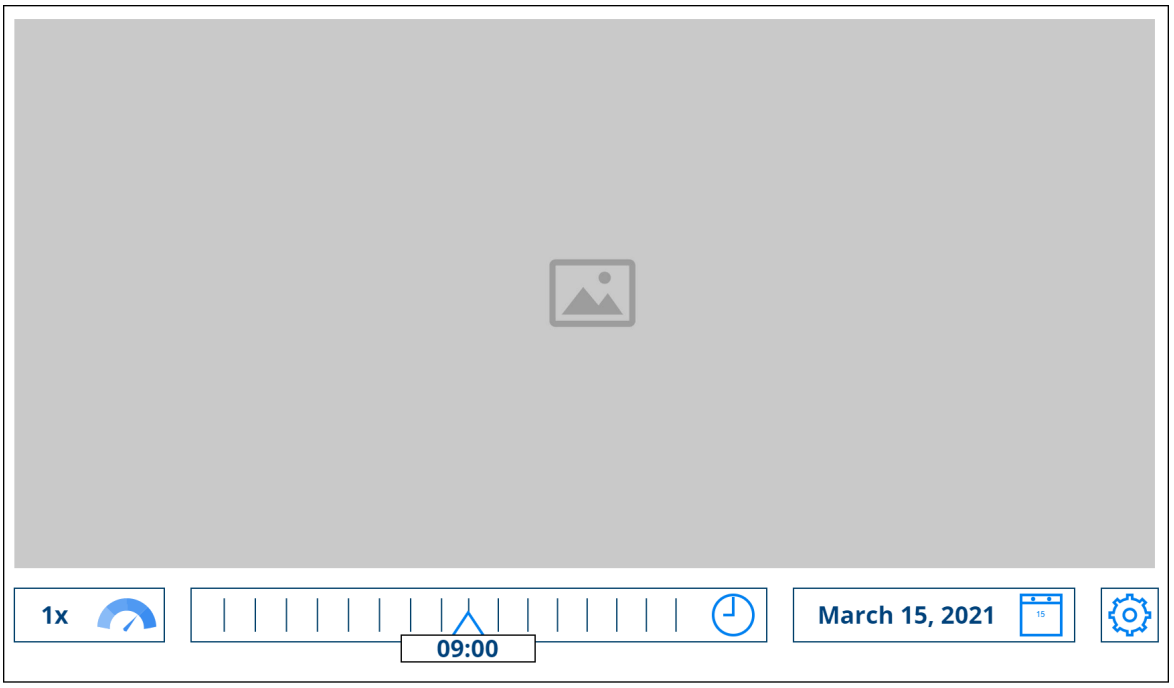


Figure 4: UI mockup of the "real-life" mode.

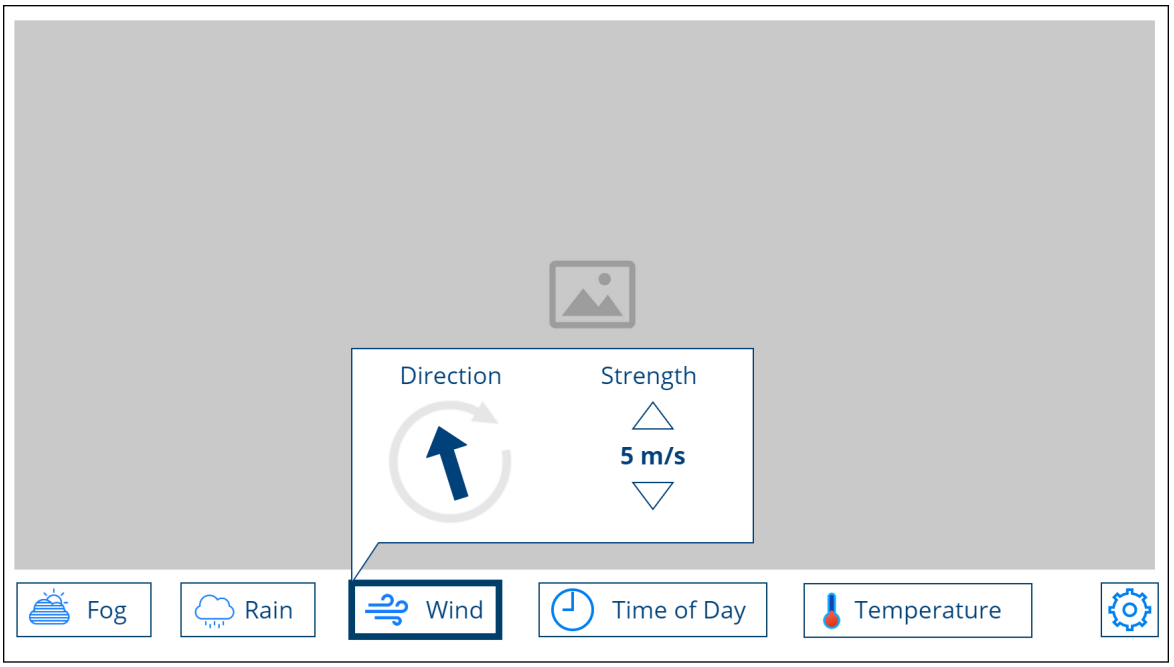


Figure 5: UI mockup of the "sandbox" mode.

2.5 Future Work

In a future project, the weather system could evolve into a complete flight simulation game where the player cruises through the clouds, like in Microsoft's *Flight Simulator*. Another interesting idea is to expand the system into a whole ecosystem with living creatures and animals. Their behaviour would be weather-dependent, making it one big symbiotic environment.

3 Scope of Work

3.1 Initial Situation

In computer graphics, especially in games, an astonishingly large group of features are reccurring across all programs and genres. With the most obvious ones being water surfaces, cloudscapes and fire effects, they are present in almost any game. Naturally, those features grew in complexity, customizability and computational demands over time.
One of the core mechanics for achieving realistic results is called a *volumetric shader*. A prototype such a shader has been created in a previous project and will be used as base.

3.1.1 Previous Work

In a previous project, the process of creating a volumetric shader has already been researched and implemented in a prototype. Thanks to its high flexibility, different cloudscapes could be rendered by the same shader.

Figure 6: Result of the previous work's shader (Evening).

Figure 7: Result of the previous work's shader (Day).

During that project, some other important topics have been researched. Among those were volumetric rendering, Perlin and Voronoi noise generation algorithms, and a technique called *ray marching*.
The implementations of those algorithms and methods will most likely be reused in this thesis and will be adapted and improved accordingly.

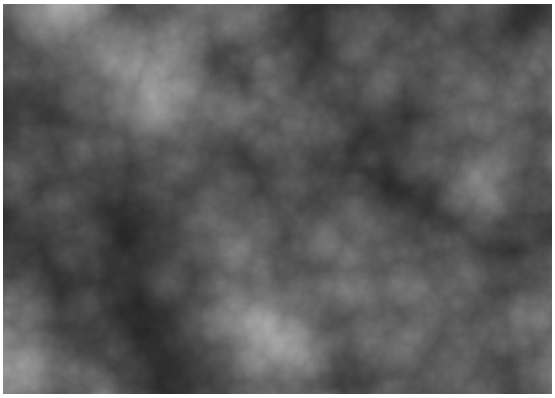


Figure 8: A generated noise texture with Voronoi's algorithm.

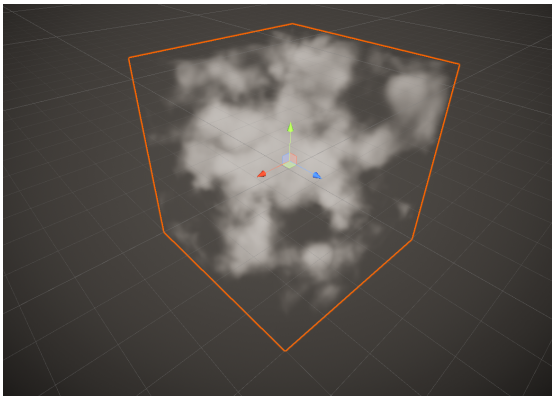


Figure 9: A screen capture of an image rendered with ray marching.

3.2 Goals

As the title of the thesis suggests, this work will primarily focus on clouds and cloudscapes. The primary goal of the project is to research and implement rendering techniques for a real-time procedural weather rendering system.
The goals will be split into two distinct groups: mandatory and optional. However, this section only defines high-level goals. A detailed specification of all requirements can be found in section 4.

3.2.1 Mandatory Goals

The following tasks must be accomplished during the project:

- Understanding the basic nature of clouds
- Understanding of different layers of clouds
- Understanding of compute shaders
- Implement a weather rendering system
- Incorporate real-time weather data from *meteoblue*
- Incorporate topological landscape models from *swisstopo*

3.2.2 Optional Goals

For further optional work, these tasks can be looked into:

- Automatic validation of realism of rendered cloudscapes
- Automatic comparison of rendered cloudscapes and photographs
- Automatic categorization of rendered cloudscapes
- Performance optimization

Most of the tasks labeled with "automatic" could be solved using a neural network. As for performance optimization, it is meant to optimize shader code, find early exits for looping algorithms and reduce the overall workload of importing the external data into the weather rendering system.

3.3 Educational Objectives

Educational objectives include shader programming, knowledge about compute shaders, rendering techniques, common algorithms used in computer graphics like noise generation, a general understanding of aspects needed to create a complete weather system and finally the incorporation of real-time data from a third party.

3.4 Used Software and Tools

All documentation will be written in LaTeX with Visual Studio Code. The shader will be implemented in Unity. The chosen shader language is High Level Shading Language (HLSL). For the presentation, Microsoft PowerPoint will be used.

4 Requirements

All requirements are grouped by type. This results in two major groups, which are research and development requirements. Each one of the requirements is derived from a goal listed in subsection 3.2.

4.1 Research requirements

Each research requirement is denoted with the letter "R" followed by its number. They are sorted according to priority, from most important to least important.

Number	R.1
Name	Understanding the basic nature of clouds
Description	In order to be able to recreate a realistically looking cloud shape, one has to examine and understand the way a cloud forms and disperses again first.

Number	R.2
Name	Understanding of different layers of clouds
Description	Among other characteristics, altitude, humidity and atmospheric pressure dictate the look and genus of a cloud. The goal is to decide which cloud types are required for a believable weather system.

Number	R.3
Name	Understanding of compute shaders
Description	Compute shaders proved to be a highly efficient tool when it comes to heavy calculations, like simulations. To improve performance and therefore allow for more a sophisticated weather system, compute shaders have to be researched.

4.2 Development requirements

Each development requirement is denoted with the letter "D" followed by its number. They are sorted according to priority, from most important to least important.

Number	D.1
Name	Noise generation based on compute shaders
Description	To make full use of the power of compute shaders, it is best to let them execute computationally demanding tasks. In this case, specifically noise generation.

Number	D.2
Name	Incorporation of real-time weather data from *meteoblue*
Description	In order to achieve a high degree of realism, real-time weather data will be used. *Meteoblue* offers different data package contracts, of which the "basic_1h" is to be acquired. The usage of the data package requires physical locations. the chosen locations are: • Bern, Switzerland • Fribourg, Switzerland • Solothurn, Switzerland

Number	D.3
Name	Incorporation of height model data from *swisstopo*
Description	The 3D height model data from *swisstopo* will be downloaded and mapped into a compatible format for Unity. This is then used as a base for the scenery. For texture layers, the satelite image data from *swisstopo* will be used and mapped onto the 3D model.

Number	D.4
Name	Periodcally store photographs of 360-degree cameras
Description	A comparison of real-time weather data and an actual photographic reference from that date and time will prove to be useful. This is why the images from such cameras will be stored on a local file system. There are many camera systems that offer 360-degree footage free of charge. The chosen system is that of the company *Seitz* called *Roundshot* [5], with these locations: • Roundshot camera Bantiger, Switzerland • Roundshot camera Gurtenpark, Switzerland

Number	D.5
Name	Implement a weather rendering system
Description	Finally, the weather system has to implemented, incorporating the external data sources and rendering images of cloudscapes. The system should be controllable via practical parameters, like point in time.

5 Testing

The project will be implemented and tested in Unity. For testing, the following test cases can be used to verify and evaluate the implementation. They are split into two groups, separating the incorporation of external data with the implementation of the system.

5.1 External data testing

Case	Test case	Expected result
T.1	Weather data	The data from *meteoblue* is incorporated into the weather rendering system. The data directly controls all related variables.
T.2	Terrain data	The data from *swisstopo* is incorporated into the weather rendering system. The height model defines the terrain height map. The satellite images are used for texturing.
T.3	Photographic data	There is a feature that allows to overlay the *Roundshot* photograph of the same time and date as the rendered image was created for.

5.2 Functional testing

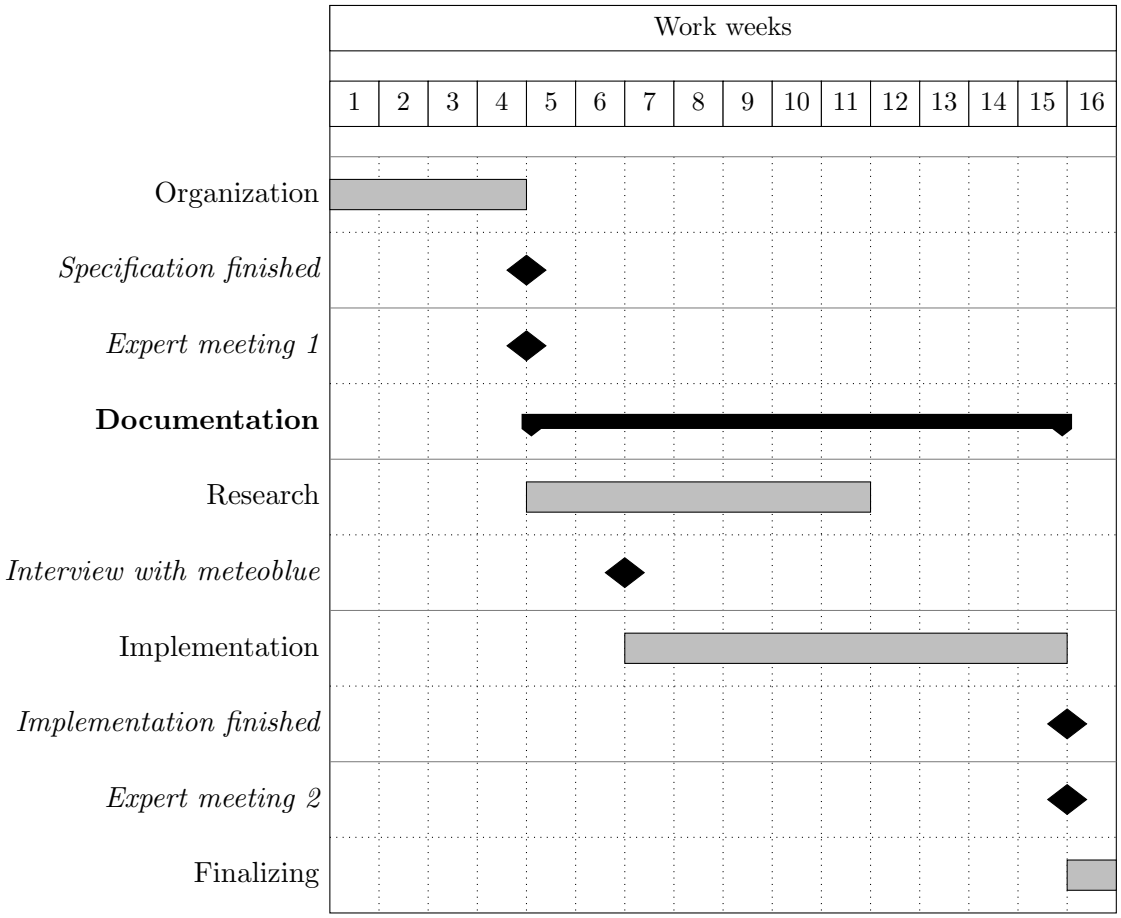
Case	Test case	Expected result
T.4	Code functionality	The code for the weather rendering system compiles and runs without error.
T.5	User interface	The user is able to switch between the two modes, "real-time" and "sandbox". The user is also able to control the weather system over the user interface accordingly.
T.6	Performance	The shader code should run with reasonably good performance and should not show visible stutters or frame drops.

6 Project management

6.1 Schedule

The time frame of the semester spans over exactly 16 weeks. Being worth 12 ECTS points, this project assumes a maximum work load of 22.5 hours per week, resulting in a total of 360 hours.

Over the course of the term, the project will be split into four primary task groups, namely organization, research, implementation and finalization. Put into relation with the duration of the project, the estimated schedule looks like this:



6.1.1 Task Groups

For each task group, the following distribution of time and effort is estimated:

Task group	Predicted effort
Organization	10%
Research	35%
Implementation	50%
Finalizing	5%

The task groups are defined as follows:

- **Organization**
  The first task group focuses on creating and finishing the project specification. This also includes the first meeting with the examination expert, Dr. Eric Dubuis.
- **Research**
  The research spans over the course of almost two months. It also continues being active during the first half of the implementation. This is necessary, as the topics will be further investigated when implementing them, which results in more research. Also, a technical interview with the firm *meteoblue* will be scheduled and held during the first couple of weeks of the research task.
- **Implementation**
  After researching each relevant topic thoroughly, the implementation can begin. In this task, the weather system will be created in Unity. Also, during research and implementation, the documentation will be continuously updated.

6.2 Project Organization

There are two kind of meetings during the project. They will be thoroughly documented in the project's journal. Should a physical meeting be impossible for some reason, an online meeting via Microsoft Teams will be held instead.

6.2.1 Weekly meetings

A meeting will be held on a weekly basis to discuss the progress of the thesis, possibly arisen issues as well as planned work for the upcoming week.

Name	Role	Participation
Matthias Thomann	Author	Mandatory
Prof. Urs Künzler	Tutor and reviewer	Mandatory

6.2.2 Expert meetings

Additionally, both before the research task begins and after the implementation task has ended, a meeting with the external examination expert will be held.

Name	Role	Participation
Matthias Thomann	Author	Mandatory
Dr. Eric Dubuis	Examination expert	Mandatory
Prof. Urs Künzler	Tutor and reviewer	Optional

6.3 Project Results

The project results are the following items:

- **Documenation**
  The documentation includes this document as well as the thesis' scientific paper.
  - Requirement specification
  - Thesis paper
- **Implementation of the System**
  The Unity project, including all implemented shader code, will be managed and stored in the given Gitlab repository [10]. This will also serve as a form of submission for grading.
- **Presentation**
  A public presentation will be held on the second last friday of the term, June 9, 2021.
- **Defense of the Thesis**
  The bachelor's thesis defense will be held after the term, on a day between June 21, 2021 and July 14, 2021. The exact date is yet to be announced.

6.3.1 Submission Terms

The following items must be submitted.

Item	**Description**	**Due Date**
Specification	This document	March 19, 2021
Book entry	An advertising one-page description of the thesis	to be announced
Poster	An advertising poster of the thesis (A1 format)	June 7, 2021
Video clip	An advertising one-minute video clip of the thesis	June 17, 2021
Thesis	The thesis paper and all of the source code	June 17, 2021
Thesis print	The printed thesis including a CD with all source code	June 21, 2021

Glossary

Compute shader A shader which runs on the GPU but outside of the default render pipeline. 8

HLSL High Level Shading Language. 8

JSON Java-Script Object Notation. 3

LaTeX A high-quality document preparation system designed for the production of technical and scientific documentation. 8

Neural network A series of algorithms that can recognize and categorize certain patterns in a given set of data. 8

Noise generation Noise generation is used to generate textures of one or more dimension with seemingly random smooth transitions from black to white (zero to one). 7, 8, 10

PNG Portable Network Graphic. 4

Procedural Created solely with algorithms and independant of any prerequisites. i, 8

Ray marching Ray marching is a type of method to approximate the surface distance of a volumetric object, where a ray is cast into the volume and stepped forward until the surface is reached. 7

Shader A piece of software which runs on the GPU, rendering geometrically defined objects to the screen. 7, 8

SLPK ESRI Scene Layer Package. 4

UI User Interface. 5

Volumetric This describes a technique which takes a 3D volume of data and projects it to 2D. It is mostly used for transparent effects stored as a 3D image. 7

References

[1] *Rendered image of volumetric clouds.* [Online]. Available: `https://www.gosunoob.com/guides/change-live-weather-problems-in-flight-simulator-2020/`.

[2] *Photographic reference of clouds.* [Online]. Available: `https://bantiger.roundshot.com/`.

[3] *Meteoblue: Weather data packages.* [Online]. Available: `https://docs.meteoblue.com/en/apis/weather-data/introduction`.

[4] *Meteoblue: Basic 1h packages.* [Online]. Available: `https://docs.meteoblue.com/en/apis/weather-data/packages-api#basic`.

[5] *Seitz: 360° roundshot camera system.* [Online]. Available: `https://www.roundshot.com/`.

[6] *Roundshot camera: Bantiger*. [Online]. Available: `https://bantiger.roundshot.com/`.

[7] *Roundshot camera: Gurtenpark*. [Online]. Available: `https://gurtenpark.roundshot.com/`.

[8] *Swisstopo: Height models*. [Online]. Available: `https://www.swisstopo.admin.ch/en/geodata/height.html`.

[9] *Swisstopo: Free geodata udpate*. [Online]. Available: `https://shop.swisstopo.admin.ch/en/products/free_geodata`.

[10] *Gitlab: Thesis repository*. [Online]. Available: `https://gitlab.ti.bfh.ch/cpvr-students/cloud-shader`.