Philipp Hafner, B. Sc.

Interactive Visualization of Sentinel-2 Satellite Data

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. Johanna Pirker, BSc

Co-Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, June 2019

Affidavit

Abstract

Satellites are an elemental part of our modern-day life. Without them, many of our daily comforts would perish, such as mobile communications, navigation or precise weather forecasts. Remote sensing satellites are used to monitor and survey our planet, revealing valuable information about our climate, urban development and much more. In order to infer new knowledge about Earth, scientists use many different utensils, such as geoinformation systems which are used for the retrieval and analysis of remote sensing data. Launched by the European Space Agency, the Sentinel-2 satellites are a recent addition to our starlit sky. To the best of our knowledge, no distinct management and analysis tool exists specifically for this platform.

The proposed *Sentinel-2 Explorer* framework is a system that manages integral parts of working with data from the aforementioned satellites, by providing an interactive visualization for satellite images. For this purpose, two distinct tools have been implemented. The first allows searching and downloading Sentinel-2 datasets, while the second offers numerous features for adjusting the visualization and conducting analyses on retrieved datasets by utilizing a modern 3D engine.

This thesis provides an in-depth description of the design and implementation with respect to the proposed system. A study testing for usability has been conducted to ensure the efficiency and usefulness of the framework. The evaluation according to System Usability Score and Computer Emotion Scale shows that the implemented system not only offers a high level of usability for experts but can also be quickly learned and subsequently used efficiently by non-experts.

Acknowledgements

First and foremost, I would like to express my utmost gratitude towards my supervisor, Johanna Pirker. Her tireless efforts and constructive feedback made this work what it is today.

Further I would like to extend my thanks to Michael Holly, providing valuable help in understanding the more fiddly parts of Unity3D.

Gratitude is also extended to Michael Schiller, who never became tired in helping me find nimble formulations and fancy words such as *henceforth*.

I would also like to thank Lukas Schabler, who, together with Michael Holly and Michael Schiller, provided endless input and helped proof reading this thesis.

Finally I would like to thank my parents, Johanna and Bernd and my brother Thomas for supporting me throughout my master thesis.

Contents

1. Introduction

During the mid 1800s, scientists conducted the first experiments in the field of photography. Almost 200 years later, countless satellites orbit our planet, taking thousands of pictures every year. They have many different purposes, ranging from weather forecasting and climate monitoring to mapping and land cover analysis and even espionage. Without such satellites, many things that we take for granted would not be possible. Commonly used (consumer) frameworks such as Google Earth rely on aerial photography and satellite imaging data (Google, 2018), precise weather forecasts would not be a thing without Europe's METEOSAT or the United States' GOES weather satellite projects. All those satellites generate vast amounts of data that have to be processed, monitored and archived. Analyzing satellite (image) data is crucial for scientists, as it allows the inference of new knowledge, helps in forming theories and even proves them. Remote sensing experts make use of many tools that help them in understanding and analyzing this data. Such tools provide a wide array of functions, such as filtering, manipulating, querying, mosaicing or batch processing datasets. However, different satellites create different images, sensed with different instruments and provided in various formats. Launched in 2015 and 2017 (Esa, 2018a), ESAs Sentinel-2 satellites provide coverage of the Earths surface every five days, with each dataset containing twelve different spectral bands. Although there are many different applications that focus on visualizing and analyzing large-scale remote sensing data such as ArcGIS[1] or QGIS[2], there is no dedicated tool that is specifically tailored to the Sentinel-2 satellite platform. Frameworks that are developed for handling satellite imagery have to provide an efficient way to visualize large quantities of data, preferably in an interactive manner. Manipulating the visualization by

[1]http://www.arcgis.com/index.html
[2]https://www.qgis.org/en/site/

adjusting parameters of how the representation is created enables experts to discover new knowledge about our planet.

1.1. Goals & Motivation

The main goal of this thesis is the design, implementation and evaluation of a visualization framework that is specifically designed for the Sentinel-2 satellite platform. The implementation of the *Sentinel-2 Explorer* includes:

- A tool that allows searching for available datasets according to various parameters. This tool includes a downloading component that automatically retrieves all datasets that are of interest for the user.
- The design of a component that enables experts to examine and analyze downloaded datasets. This includes various different tools that perform operations on the dataset and make use of Sentinel-2 specific conditions.

The implemented framework has to provide a multitude of features relevant to expert users that are not necessarily experts with respect to the Sentinel-2 satellites. Therefore, the usefulness of the provided tools will be evaluated. This study will asses:

- The usefulness and effectiveness of the implemented tools.
- The usability and user-friendliness of the provided tools.
- The overall system performance regarding loading times and how this affects the users' perception of the system.

1.2. Methodology & Structure

This thesis is structured into three main parts, outlined in Figure 1.1. The first part focuses on the background (Chapter 2) of this work, namely the science of remote sensing and various interactive visualization techniques. This background defines the purpose of the *Sentinel-2 Explorer*. The second part examines the provided datasets acquired by the satellites (Chapter 3), as they in part define how the framework has to be designed, which will be discussed in Chapter 4. Together with the definition of the stakeholders and requirements, the final design is presented. Chapter 5 describes in detail how the *Sentinel-2 Explorer* is implemented. The third part of this thesis focuses on the evaluation, which will determine the effectiveness and usefulness of the design and implementation (Chapter 6).

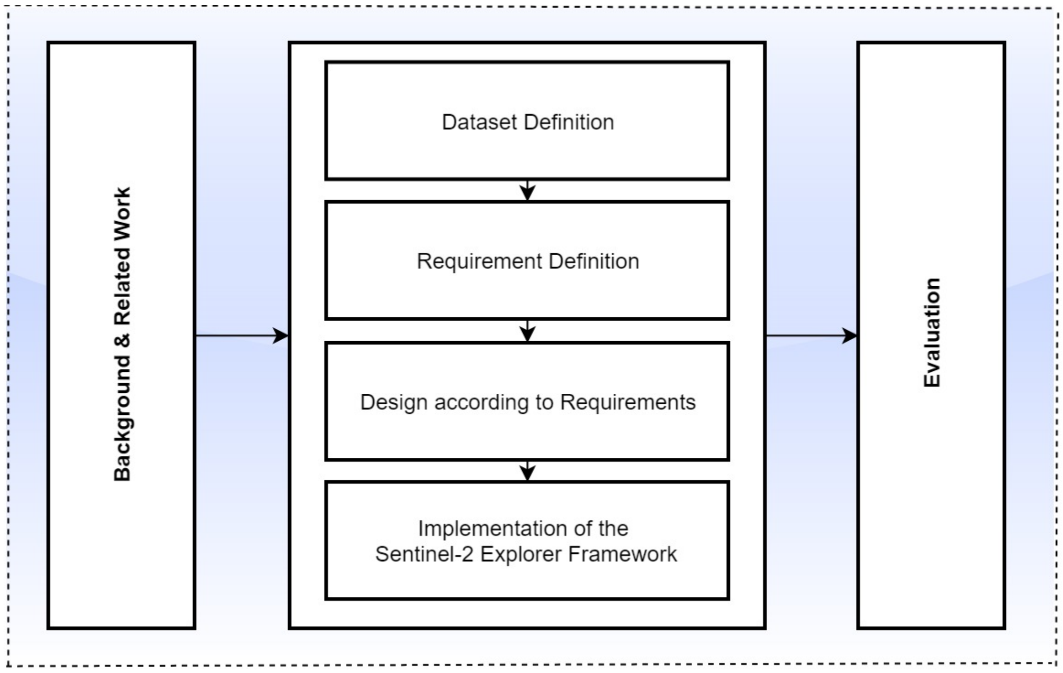

Background & Related Work
Dataset Definition
Requirement Definition
Design according to Requirements
Implementation of the Sentinel-2 Explorer Framework
Evaluation

Figure 1.1.: The structure of this thesis.

Chapter 2 will focus on the background and related work regarding remote sensing and visualization systems, respectively. First, a brief history of remote sensing is given. In succession, the instruments, sensors and use of them on satellites is explained in more detail. We then discuss interactive data visualization and techniques on how to design such a system and what benefits are to be expected when working with an interactive system. The chapter closes by presenting related work in the field of analysis and visualization of satellite data.

Chapter 3 introduces the Sentinel satellites by defining their respective field of usage. The process of how Sentinel-2 data is acquired and pre-processed is explained in detail. After explaining how this is done, the specific data format is explained.

Chapter 4 starts by defining the core functionality of the framework. This is followed by identifying the target user group. We then specify all functional and non functional requirements and present the design strategy which serves as a basis for the implementation.

In Chapter 5, the actual implementation of the framework is described in detail. Following the same structure as in the design chapter, showing how the specific design elements were incorporated to create the envisioned system. This includes a description of all implemented analyzing tools that form the core of the *Sentinel-2 Explorer*.

Chapter 6 deals with the evaluation of the framework. This is done by explaining the selection of participants and which tasks they had to fulfill. The analyzed results of all questionnaires are presented and discussed. This chapter will answer several questions, such as whether Unity3D proves to be a good foundation for implementing such a system and how well the implemented analyses tools perform when operated by experts.

Chapter 7 describes problems that have occurred during the design and implementation of the framework. This is followed by possible improvements for this framework in Chapter 8, before summarizing research results and their implications.

2. Background

This chapter serves as an introduction to the topic of remote sensing and interactive visualization techniques. After a primer to the history of satellite imaging, a brief definition of remote sensing is given. The applications and uses as well as challenges are explained. Further, the visualization of large-scale data in general and the need for interactivity is explained. The description of selected other projects and related work associated with visualization and data-extraction of remote sensing imagery will conclude this chapter.

2.1. A Brief History of Space Photography

The first picture of Earth was taken on October, 24th 1946 (see Figure 2.1), almost ten years before the launch of the first satellite, Sputnik 1. Taken by a camera mounted on a captured V-2 rocket, it was the first time humans saw Earth from space taken at an altitude of approx. 105km. Until the aforementioned date, the record for the highest altitude at which a photo was taken, was held by the Explorer II balloon in 1936. The image from this V-2 rocket beat this record by over 80km. There where dozens of V-2 launches in the late 1940s, where engineers tried to refine their knowledge and skills in rocketry and scientists mounted all sorts of instruments inside the noses of these captured rockets to measure temperature, pressure or magnetic fields. In total, over 1000 images of Earth have been taken during these trials, the highest one taken at an altitude of 160km (Reichhardt, 2006).

The true era of satellite remote sensing began with the launch of Sputnik-1 in October 1957, which orbited around Earth every 96 minutes while trans-

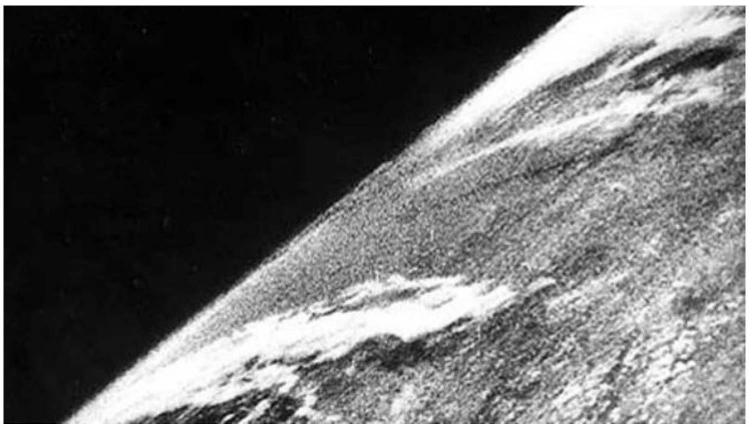


Figure 2.1.: The first photo from space. (U.S. Army, White Sands Missile Range/Applied Physics Laboratory, 1946)

mitting radio signals. This event marked the beginning of a true rush in space satellites. After a month, Sputnik-2 was launched. The US followed with the launch of Explorer-1 in January 1958 and Vanguard-1 in March 1958, which still orbits the Earth, marking it the oldest man-made object in space.

The first TV images from space where recorded by TIROS-1 (1960), which is considered to be the world's first weather satellite. It was equipped with two cameras which allowed for precise weather forecasts, marking a milestone in remote sensing. In parallel with the TIROS program, NASA launched the NIMBUS project, consisting of eight satellites. The primary objective was the monitoring of oceans and gas exchanges between oceans and the atmosphere. The last satellite of the mission, Nimbus-7 was launched in 1978 and ended its operational life in 1994. It was the first NASA satellite used for global environmental monitoring. In the 1970s, the declassification of military satellites, which used infrared and microwave sensors for Earth observation, allowed many new developments. For land observation, NASA and the US National Academy of Sciences launched Landsat-1 in 1972 (Tatem, Goetz, and Hay, 2008). It was the first satellite solely built for the observation and study of Earth. The on-board multi-spectral-scanner (MSS) recorded on four different spectral ranges: green, red and two infrared bands. It provided repetitive images in a resolution of approximately 80m (Irons, 2018).

During the 1980s, significant advances in sensory equipment for satellites were made. Hyper-spectral sensors, which combined the information from several spectral bands and multi-angle spectrometers that combine views from several azimuths were only one of the new inventions for remote sensing. New sensors also have been incorporated, for example altimeters for accurate sea level measurements that can deliver accuracies up to a few millimeters.
Tatem, Goetz, and Hay (2008) note, that today, more than 45 years after the launch of Landsat-1, many countries operate smaller satellites based on the same design for their own scientific purposes. One of the most recently launched projects is the Sentinel program. Part of European Space Agency's Earth observation program Copernicus, it consists of nine satellites (six of which have been launched) which focus on different tasks, such as radar monitoring, passive-optical monitoring and ocean surveying (ESA, 2018f).
In all those years, remote sensing satellites vastly increased our understanding of weather, climate and our Earth. Accurate weather forecasts or long-term monitoring of changes in our climate and atmosphere would not be possible if it were not for the developments of the last 60 years. This has brought mankind from the first image in space, which barely showed the curvature of the Earth to highly-advanced multi-spectral imaging instruments which allow spatial resolutions of up to 0.3m (Blau, 2016).

2.2. Satellite Remote Sensing

The process of observing nature and making measurements from afar is called remote sensing (Jensen, 2013). According to Colwell (1985), it is the acquisition of information of some property of an object or phenomenon, by a recording device that is not in physical or intimate contact with the object or phenomenon under study. In the field of Earth observation, remote sensing is usually done by aircraft or satellites. Spacecraft-based sensing instruments (which of course include satellites) have numerous advantages over aircraft based systems, such as broader spatial (potentially global) coverage as well as increased temporal resolution. This is opposed to the much higher operational costs necessary for orbiting sensors in contrast to aircraft equipped with the same instruments (Rencz, 1999, chap. 6 p. 4).

2.2.1. Active & Passive Instruments

There are two types of sensing instruments: active and passive. Active instruments emit some sort of radiation towards the observed object and measure the reflected/backscattered signal, such as radar (radio detection and ranging) or Laser (light amplification by stimulated emission of radiation). Passive instruments only measure reflected/emitted energy from the viewed area or object. Visible, infrared and near-infrared light is measured by the backscattered sunlight from the earths surface or clouds. This wavelength-range between $0.4\mu m$ and $3\mu m$ is called solar-reflective range because the sun-supplied energy on the Earth's surface exceeds the energy emitted by the planet itself. Above wavelengths of $5\mu m$ the Earth itself emits more thermal radiation than the sun. Since these measurements do not require direct exposure to sunlight, accurate thermal infrared (TIR) readings can be acquired during day and night (Schowengerdt, 2006, chap. 1 p. 10).

2.2.2. Applications & Use

As described by Schowengerdt (2006), remote sensing has various different applications. Deployed on satellites, different sensing instruments usually provide a repetitive view of the planet which allow monitoring of short- and long-term changes. Some of the more important fields are:

- **Mapping:** topography, land use, roads
- **Meteorology:** atmospheric changes, weather prediction
- **Environmental Assessment:** population growth, urban monitoring
- **Change Detection and Monitoring:** deforestation, glacier-melting, climate change

There are different parameters for remote-sensing, such as temporal, spatial and spectral resolution. Meteorologists, for example, desire more frequent coverage and do not rely on very high spatial resolutions, whereas mapping applications need the highest possible spatial resolution and do not require frequent observations at all (Schowengerdt, 2006, chap. 1 p. 2).

Graham (1999) provides a short overview of the most commonly found instruments used on satellites. Naturally, different applications call for different instruments. There are different methods for sensing that are used on satellites. For example, an imaging radiometer is used to measure the intensity of electromagnetic radiation in a specific spectrum. This spectrum may be ultraviolet, visible, infrared or even microwave. The result is an image containing the reflectance values in a specific spatial resolution. A radar uses a transmitter operating at a specific frequency. Depending on the application, a radar may be used for topography mapping or observation of wind speeds.

Naturally, remote sensing satellites produce enormous quantities of (image)-data, regardless of the specific application area. This information has to be visualized in order to make analyses by scientists possible. Humans acquire more information with their two eyes than with all other senses combined. Our visual system is a flexible pattern finder paired with an adaptive decision-making mechanism (Ware, 2004, chap 1. p.2). The next section provides an introduction to interactive data visualization, which provides the cognitive support that allows humans to understand this complex and high-dimensional data.

2.3. Interactive Data Visualization

The typical amount of data created by remote sensors is very large. To understand this data, it is crucial to find a way to create a meaningful visualization of this data. Since this is true for most sensors or other data sources, data visualization has become essential in today's research. As Weiskopf (2019) puts it, an efficient visualization method is a prerequisite for an interactive visualization. This in turn allows for increased effectiveness. This means that in the field of visualization, effectiveness and efficiency are closely intertwined.

2.3.1. The Visualization Pipeline

According to Haber and McNabb (1990), (data)-visualization can be defined as the use of imaging technology as a tool for understanding data provided by a simulation or a measurement. It includes other technologies, such as computer vision and graphics as well as more human-centered sciences, for example perceptual psychology and user interface studies. They (Haber and McNabb, 1990) describe a visualization pipeline, which represents a basic workflow for visualizing data. It is shown in Figure 2.2.

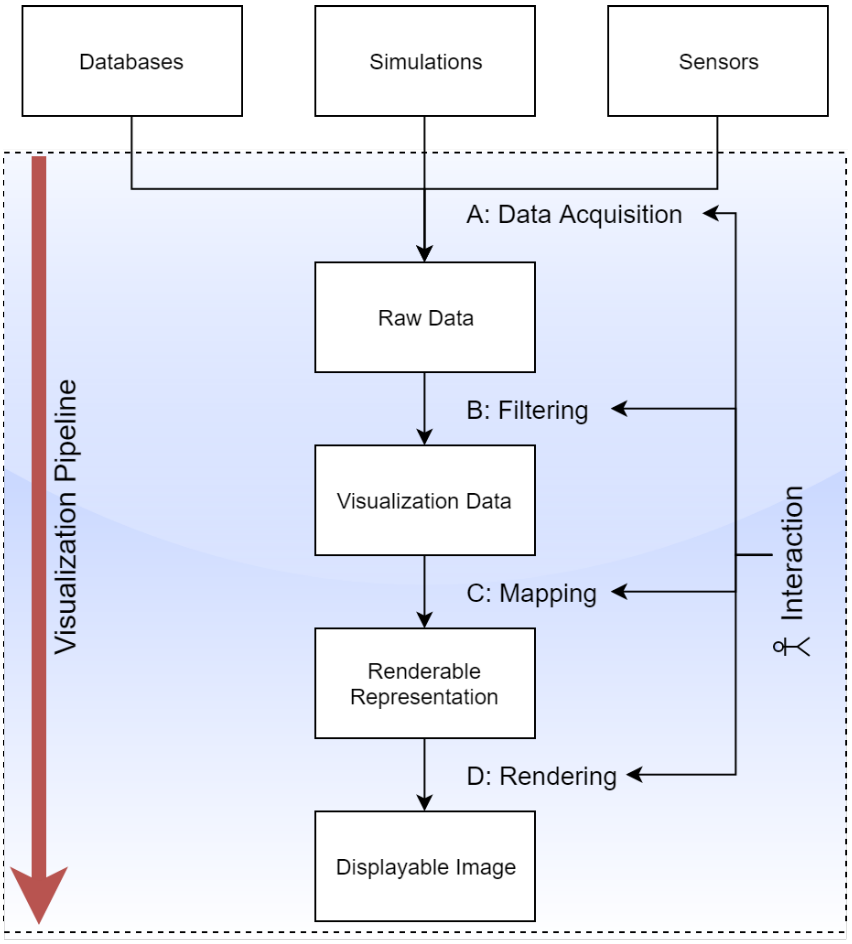


Figure 2.2.: Visualization pipeline, adapted from Weiskopf (2019).

A: Data Acquisition

As explained by Weiskopf (2019), the input for a visualization framework is acquired from an arbitrary data source. This may be from a simulation or in this case the sensor of a remote sensing instrument. The structure of the data used for this work will be explained in detail in Chapter 3. These datasets can be seen as the raw input for the designed visualization system. However, large-scale data (such as remote sensing data) needs a data model that allows efficient usage. Rhodes, Bergeron, and Sparr (2003) argue, that identifying a suitable model is crucial for managing and processing large-scale data. One method is using a multi-resolution approach to allow fast and efficient analysis and use of the data set. Freitag and Loy (1999) explain that post-processing large data sets is a prerequisite for interactivity. They elaborate, that this is usually done by building a hierarchical, multi-resolution version of the data to be visualized. This includes a series of coarse representations. The level of detail is controlled by metrics such as error tolerance or user input, such as field of view. In terms of satellite data visualization, this means if the user has zoomed out, less detail has to be shown, given a fixed screen-resolution. According to Heckbert and Garland (1994), the appropriate level of detail within the model for a given view is the coarsest level that looks identical to the original dataset. For this, they provide a metric to measure the difference between an image approximation $f(x,y)$ and the original input image $\hat{f}(x,y)$. The best error model would include the model of the human visual system. Since this is arguably too complex, a useful result can be achieved by simply calculating the sum of squared distances in the RGB color space, as seen in Eq. 2.1.

$$E(f,\hat{f}) = \sum_{x,y} \left\| f(x,y) - \hat{f}(x,y)) \right\|^2 \tag{2.1}$$

They continue that this error metric could be improved by including different weighting factors to account for the different sensitivities of the human eye to different color channels.

B: Filtering Stage

During the filtering stage, the raw data is transformed to abstract visualization data. This operation may include de-noising, detection of erroneous measurements or applying segmentation algorithms. Although much of the segmentation and filtering work in the case of Sentinel-2 data is already done by ESAs pre-processing, such as identifying and labeling pixels or classifying erroneous and missing entries. Still, the user might want to apply custom filters, such as vegetation indices (see Section 3.3).

C: Mapping Stage

After the filtering stage, the visualization data is mapped to construct a renderable presentation. This is one of the crucial steps in a visualization framework. When dealing with image data, the mapping stage includes the assignment of color values to the input data values (Weiskopf, 2019, chap. 1 p.3). The data now has attributes such as color, transparency or color mappings, which maps colors to different labels in the classified data. This color mapping has to be chosen carefully, as for example Sentinel-2 data provides a radiometric resolution of 12-bit (Suhet, 2015, p. 53). This makes it necessary to normalize each pixel color value $p_{x,y}$ in an image P a to an $\hat{p}_{x,y}$ value in an 8-bit (0-255) image $\hat{P}$. Furthermore, when visualizing vegetation indices (see Section 3.3), which contain floating point values ranging from $[-1 \leq p_{x,y} \leq +1]$, an intuitive color scale has to be defined. Normalization can provide an easy solution, by simply re-ranging the values to an 8-bit space. A general formula for this transformation is as follows:

$$\hat{p}_{x,y} = \frac{\left(p_{x,y} - \min\limits_{p_{x,y} \in P}(p_{x,y})\right) * \left(\max\limits_{\hat{p}_{x,y} \in \hat{P}}(\hat{p}_{x,y}) - \min\limits_{\hat{p}_{x,y} \in \hat{P}}(\hat{p}_{x,y})\right)}{\max\limits_{p_{x,y} \in P}(p_{x,y}) - \min\limits_{p_{x,y} \in P}(p_{x,y})} + \min\limits_{\hat{p}_{x,y} \in \hat{P}}(\hat{p}_{x,y}) \tag{2.2}$$

This can be more simplified for the case where an arbitrary range is remapped to the range $[0 \leq p_{x,y} \leq 255]$:

$$\hat{p}_{x,y} = \frac{\left(p_{x,y} - \min\limits_{p_{x,y} \in P}(p_{x,y})\right) * 255}{\max\limits_{p_{x,y} \in P}(p_{x,y}) - \min\limits_{p_{x,y} \in P}(p_{x,y})} \tag{2.3}$$

Ward, Grinstein, and Keim (2015) note, that mapping changes have to be communicated to the user to avoid confusion or errors. Users might want to specify a custom mapping interactively, which allows for a higher value range in a specific spectrum. By truncating the range, the variation of the shortened range is perceived more easily. Normalization should therefore include bounding values, which indicate that specific values exceed or undercut the (user-specified) range.

The visual identification of missing or erroneous measurements is crucial when handling satellite data. This can happen frequently, even depending on the users needs. For example, one might want to exclude detected clouds from the visualization. Therefore, cloudy areas have to be color coded somehow, conveying the fact that the currently viewed region does not contain usable data.

Ward, Grinstein, and Keim (2015) describe multiple methods of handling missing or erroneous data. The aforementioned color coding of cloudy areas is described as setting a so-called sentinel value (see Figure 2.3). When applied to a numerical data set, a sentinel value simply is a placeholder to preserve the structure of the data. In the field of remote-sensing data, a sentinel value might be a user-specified color, such as black or red.

Other methods include interpolating over missing data-entries. While this method might be considered useful when dealing with many sensory data, this approach is not reasonable when dealing with remote sensing data.

D: Rendering stage

The rendering stage creates the final output to obtain a displayable image. This may include transformations for 3D-data or shading and illumination effects. In our visualization framework, the rendering stage is handled by

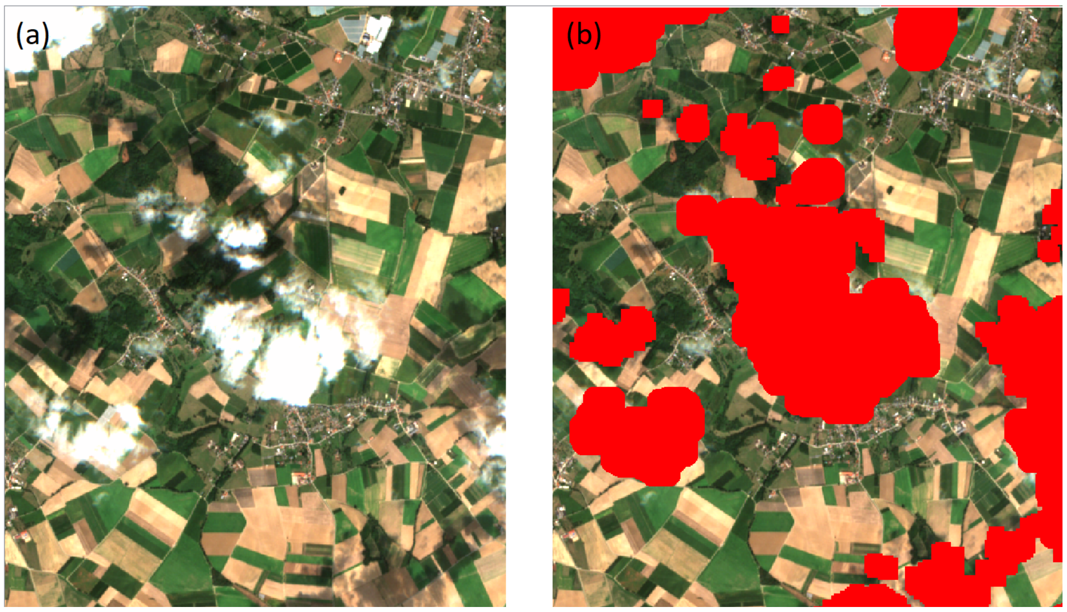


Figure 2.3.: Example for sentinel values. (a): Original image. (b): Image with assigned sentinel value (red) for classified clouds and cloud-shadows.

the Unity Engine and controlled through custom shaders, which will be explained in Section 4.4.2.

Satellites provide multi-dimensional datasets. These dimensions not only include the position X and Y, but also their values for each spectral band. In the case of Sentinel-2, in addition to the coordinates, there are twelve bands, resulting in a total of 14 dimensions, 17 if the scene classification and snow/cloud-masks are also taken into consideration. If vegetation indices are also counted, which will be explained in Section 3.3, we can increase the dimensions to well above 20. It is possible to map at least five dimensions (X, Y and three arbitrary dimensions such as R, G, and B) in an image, but this does not even visualize 50% of the available raw data.

Parallel coordinates (||-coords) can provide a solution on the problem of how to render this high-dimensional data. As explained by Inselberg (2008), they transform multivariate relations into 2D patterns that can be explored and analyzed, making them an interactive visualization technique. ||-coords are defined through placing N lines (labeled $\overline{X}_1, \overline{X}_1, ..., \overline{X}_N$) equidistant and

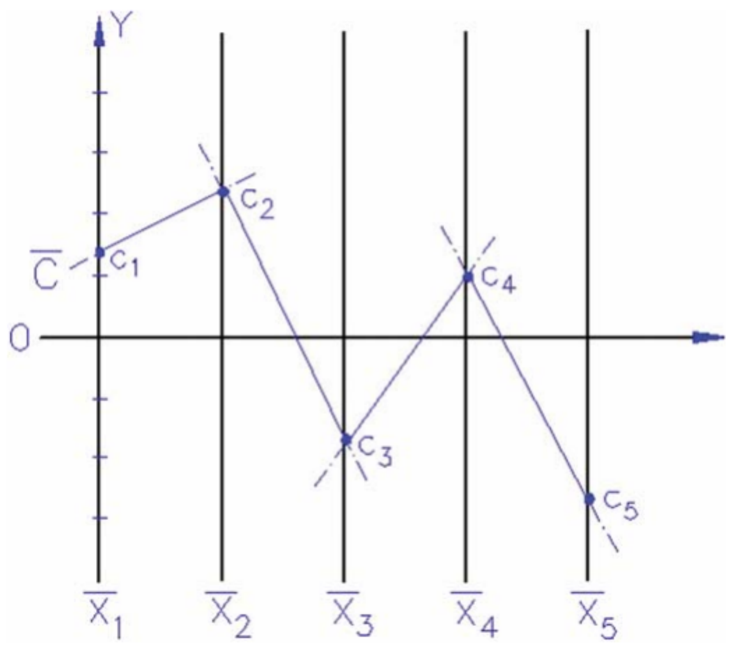


Figure 2.4.: A possible implementation for interactive handling of time-variant data (Inselberg, 2009).

perpendicular to the x-axis. They themselves are axes for the ||-coords system in N-dimensional Euclidean space $\mathbb{R}^N$. A point C with the coordinates $c_1, c_2, ..., c_n$ is constructed by the polygonal line $\overline{C}$. More specifically, the point is constructed by the line segments between the axes, whose vertices c_i are their values in the $i - th$ dimension of the dataset. Through this, a 1:1 relationship between the points in $\mathbb{R}^N$ and the planar polygonal line is established. Figure 2.4 illustrates the explained principle.
The attentive reader will undoubtedly see the possible applications towards satellite data. By mapping all dimensions of satellite-image to such a ||-coord system, it is possible to get an idea of how the dataset is structured in terms of the spectral values contained in it (see Figure 2.5).

Inselberg (2009) has demonstrated thoroughly how this can be interactively used for data exploration. A appropriately designed GUI-widget would allow the user to restrict (pinch) the visualization to the most relevant values of his desire, by limiting the range for one dimension.

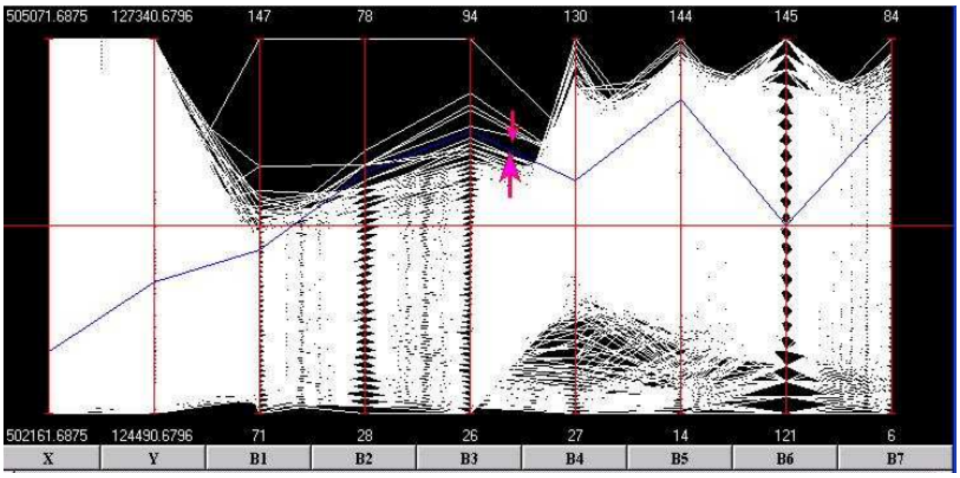


Figure 2.5.: All spectral bands of a satellite-image visualized in a ||-coord system (Inselberg, 2009).

2.3.2. Interactivity in Visualization Frameworks

The previous description of the visualization stages point out that user-interaction may be desired or even required in each individual stage. For example, the user might want to label pixels with certain properties, prompting an interaction with the filtering stage. As explained in detail, providing the ability to apply a custom color-mapping is an interaction with the mapping stage. Even during rendering, the user might want to change which value-ranges are actually rendered in the final image. This further shows that visualization and interactivity are closely related. This subsection further highlights interaction mechanisms which are useful for an interactive visualization framework and interact with multiple stages of the visualization process at once.

Non-interactive (static) visualizations offer only pre-calculated data. This means, that different "layers" (i.e. band combinations) have to be stored independently. This may be fine when alternate views are not needed. An example for such a visualization might be a plain print medium. Interactive and therefore dynamic visualizations motivate people to explore data for themselves, encouraging interest (Murray, 2017). To achieve this, Shneiderman (2003) has proposed a visual information seeking mantra which identifies seven abstract high level tasks for interactive information systems:

- Overview: Allowing the user to gain a basic understanding of the entire collection.
- Zoom: Items of interest can be enlarged.
- Filter: Uninteresting items can be hidden.
- Details on demand: Details about a dataset should be shown only when needed.
- Relate: View relationships between different items.
- History: Keep a record of performed tasks to allow actions such as redo or replay.
- Extract: Allow extraction of sub-sets and queries.

Craft and Cairns (2005) point out, that although these points are to some extent vague, their methodological suggestions are accepted. They can be seen more as an inspiration and guideline rather than a fixed approach. Yi, Kang, and Stasko (2007) seek to identify the cohesion between visualization systems and the benefits of interactivity. They conclude that the two are deeply intertwined. Even a static, non interactive visualization is often interactively used by the user itself. A poster or photograph can be used interactively by rotating or holding it closer/further to the eyes, making the process somewhat interactive albeit the poster being a static representation of information. As pointed out by Ware (2004) on p. 382, (interactive) visualization is a two-way interface and highly asymmetric. Much more information flows from the system to the user than in the other direction. Yi, Kang, and Stasko (2007) add, that this indicates, that interaction techniques are more designed for changing and adjusting the visualization in contrast to entering actual data into the system. Combining different views can also be an interactive process. The user might want to overlay different datasets onto each other to easily see correlations or differences. Ware (2004) (p. 205) identifies this as a desireable feature of satellite visualization tool-kits. He points out, that there are many pitfalls in doing this, as the different layers will always interfere with one another to some degree. Sometimes the two layers will fuse perceptionally together, making it hard to distinguish which layer the information belongs to.

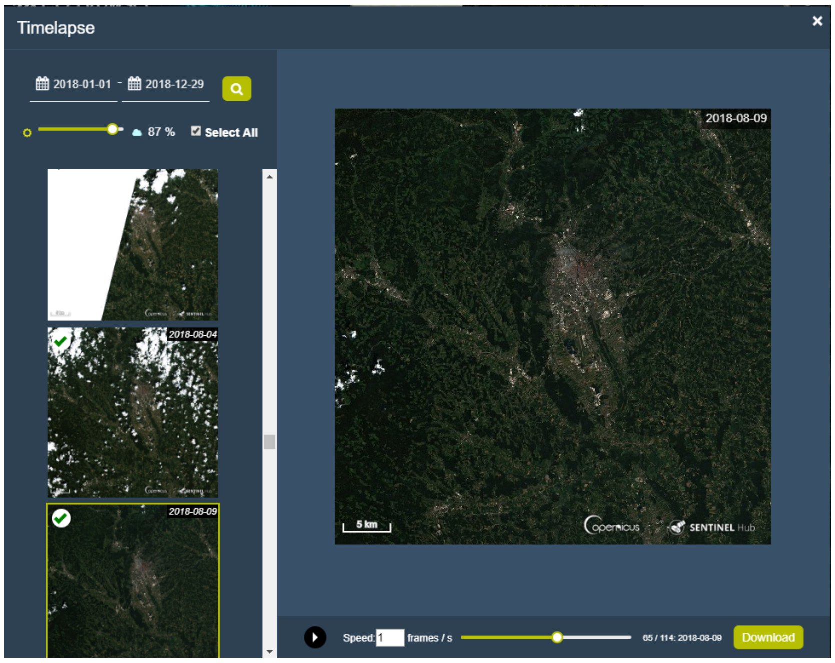


Figure 2.6.: A possible implementation for interactive handling of time-variant data (Sinergise, 2018a).

Another important task is to allow the integration of temporal data visualization and the ability to allow for interactive adaptions to the resulting representation. Ward, Grinstein, and Keim (2015) (p. 249) write, that advanced geoinformation Systems (GIS) often integrate mechanisms that allow the modeling of relations between datasets that change over time. Sinergise (2018a) incorporated a feature in their EO browser, which allows to create an animation of datasets from different sensing times. An example is shown in Figure 2.6

2.4. Visualization of Satellite Data

This section focuses on the generation and visualization of satellite data. This includes remote sensing data derivatives (Digital Elevation Models and Vegetation Indices). In addition, a brief overview to existing satellite data visualization projects is shown.

2.4.1. Remote Sensing Data Derivatives

Smith and Clark (2005) investigated methods for the visualization of digital elevation models (DEM) for land-form mapping. They reviewed twelve visualization methods and discussed their advantages as well as disadvantages. For topographic analysis, they consider methods such as gradient-

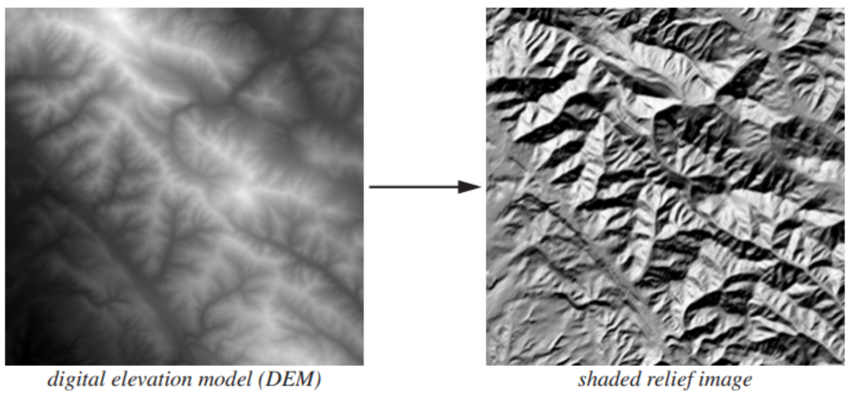


Figure 2.7.: Example of a raw DEM (left) and after processing (right) (Schowengerdt, 2006).

calculation, local contrast stretching and slope curvature computation among the best available techniques. In addition, they found that most investigated datasets feature a strong northwest to southeast lineament. This should be considered when shading or lighting DEMs (See Figure 2.7).

Jackson and Huete (1991) investigated vegetation indices (VI), which are used to identify and classify different types of vegetation in satellite images. This is done by combining, fractioning and summing the values of different

spectral bands. They found, that while VIs are useful for classifying vegetation (see Figure 2.8), they depend on atmospheric correction, taking into account sensor view and solar zenith angles. Different types of indices will also be covered in the next chapter.
Loranty et al., 2018 wrote a review article about indices, where they compared different indices to one another, showing that classification relies strongly on the used indices types. They argue that with new multi-spectral sensors, the role and importance of indices will grow in the future.

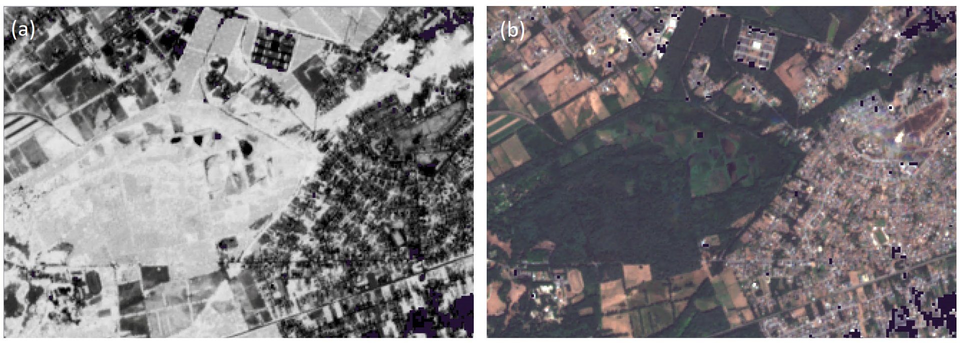


Figure 2.8.: Example for the NDVI (see Section 3.3). The left (a) image shows the NDVI values for the original scene on the right (b).

2.4.2. Sentinel-2 Data Processing

In the field of Sentinel-2 data processing, Muller-Wilm et al. (2013) provide a description of the Sen2Cor processor, which computes atmospheric correction to single-date Level-1C data in order to obtain Level-2A data, which will be described in Chapter 3. The processor also calculates a scene classification (SCL), which is used to classify each pixel. These classifications include, among others, vegetation, bare soil and various clouds types. An example of this scene classification can bee seen in Figure 2.9.

Louis et al. (2016) provide further insights in the development state of the Sen2Cor processor and the improvements made since its initial release, showing that the scene classification provides a mean overall accuracy of approximately 80%.

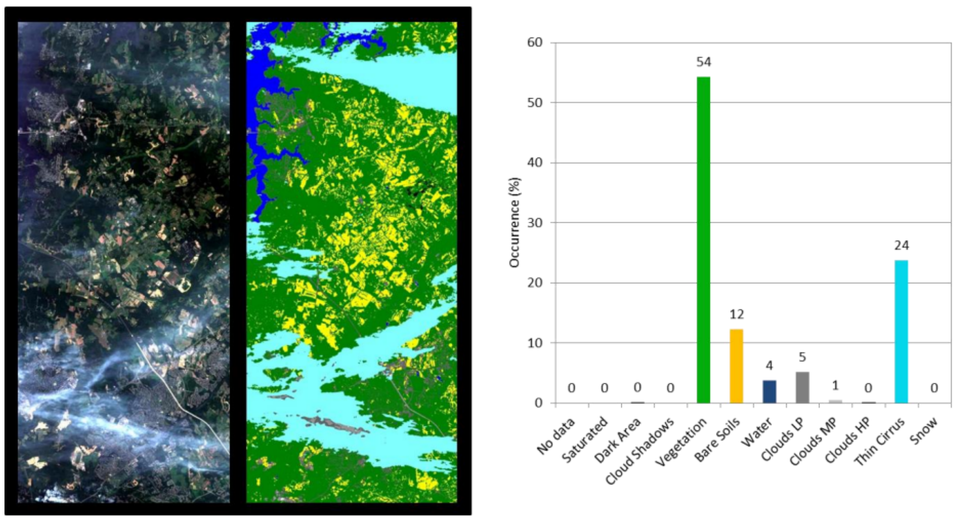


Figure 2.9.: Example of an original L1C image (far left) and the resulting scene classification (left and graph on the right), adapted from Muller-Wilm et al., 2013.

Another approach to deliver atmospheric correction to a time-series was presented by Hagolle et al. (2015), where the original aerosol optical thickness (AOT) was estimated. They have compared multi-spectral and multi-temporal approaches with ground-truth data provided by the Aerosol Robotic Network (AERONET) and found that a combination of both approaches yields better results and increases robustness.

2.4.3. Visualization Projects

Dobashi, Yamamoto, and Nishita (2009) have presented an interactive near-realtime rendering system for Earth-scale clouds. This is achieved by using volumetric data with different levels of detail (LOD), compensating for the viewpoint distance. Realistic results are achieved by taking into account atmospheric effects and cloud shadows as well as light scattering inside the clouds (see Figure 2.10). To increase performance, the initial volume data set is subdivisioned and divided into different classes, depending on the viewing-distance. The source data is derived from IR-satellite data.

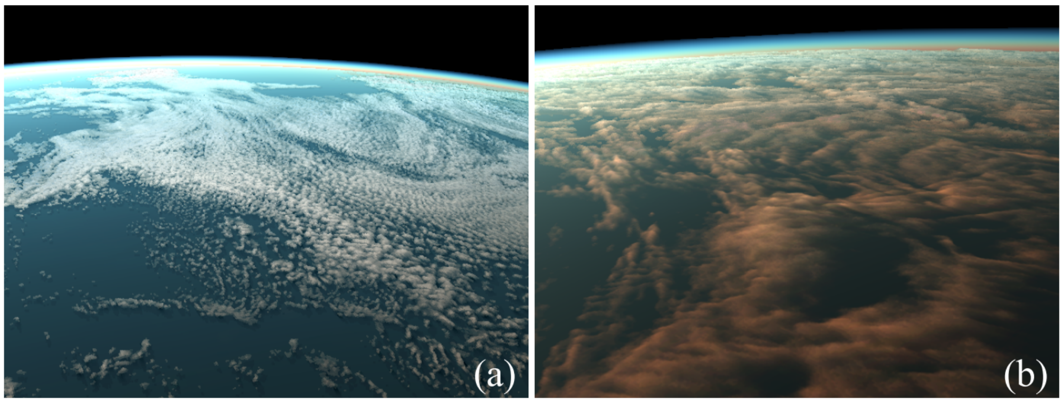


Figure 2.10.: Near Realtime Cloudmapping: (a) shows an example of 3D clouds. In (b) the clouds are red because the lightning conditions are set to evening (Dobashi, Yamamoto, and Nishita, 2009).

Further work on cloud-models has been done by Liang and Yuan (2016), using IR-satellite data to derive time-varying cloud animations. This is done by combining five spectral bands from the Chinese weather satellite FY-2F. According to the detected cloud type, different filtering algorithms are applied, including cloud height estimation and thickness computation. From this data, a volume can be constructed. In combination with a time-series, a 3D-animation can be created to allow analysis of weather phenomena, such as typhoons. Due to the spectral range of the used bands, the algorithm can be applied to many different satellites.

In addition, Nishita et al. (1993) have proposed an algorithm for physically-based image synthesis of Earth to allow photo-realistic results, taking into account light-scattering due to particles in the atmosphere. Beyond that, radiative transfer of water molecules is taken into consideration to ensure realistic colors and reflections. This results in a physically correct representation suitable for use in weather forecast or flight simulators.

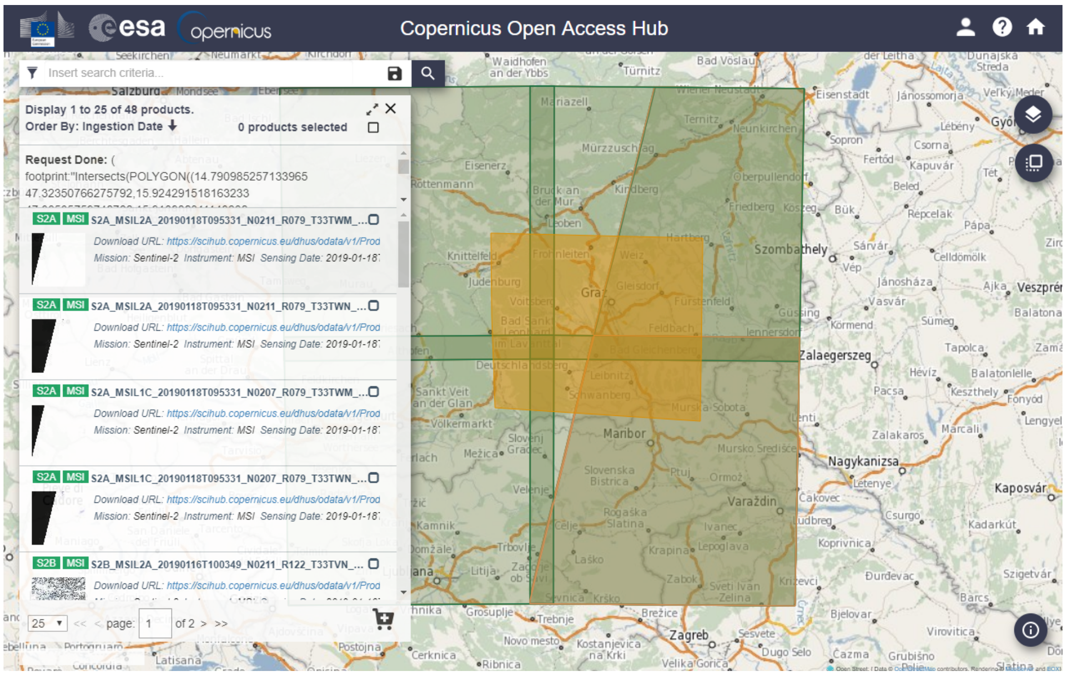
In the field of geo information systems (GIS), Foresman (2004) has investigated different 3D-geobrowsers and identified their strong suits and weaknesses as well as key-requirements for such systems. These include the capability to handle different data types from different satellites as well as LOD handling and general hardware requirements. They conclude that no definitive system represents a panacea for the challenges provided by science and industry.

ArcGIS is a powerful system for displaying, mapping, and analyzing spatial data. It is developed and sold by Environmental Systems Research Institue, Inc. (ESRI). Originally developed for large mainframes it has evolved into a desktop system with a graphical user interface (GUI) (Geology, 2015). An extension, ArcGlobe is specifically designed for the use of very large datasets and allows seamless visualization of raster (image) or feature data. It incorporates level-of-detail handling and tiling for increased efficiency. ArcGlobe operates on a global view (esri, 2019).

Dodge, McDerby, and Turner (2008) explore the state of the art of geographic visualization, primarily relevant to social scientists. They divide visualization techniques into three epistemological classes. *Looking* is the presentation, thematic mapping and overall displaying of spatial data. This includes not only 2D-data, but also 3D-layered information. *Querying* includes the visual interfaces designed for information access. It enables the user to navigate through complex information to retrieve relevant subsets of information. The third category, *Questioning*, describes the process of filtering and probing the displayed data to answer the fundamental "what" and "why" questions. This means that the system involves interactive mechanisms, such as zooming, linking, etc. These operations should be highly responsive. One of their conclusions is that modern geographic visualization concepts struggle with the handling of missing or uncertain data.

Built on the open-source JavaScript framework Cesium and WebGL, Müller et al. (2016) have built an interactive visualization tool named "GPlates Portal" which provides a series of virtual globes specifically designed for end-users. They argue that web-based applications still suffer from limitations compared to their desktop counterparts. Particularly tools which might be useful to power-users are not available.

The Copernicus Open Access Hub is a web-service which provides complete and open access to all available Sentinel User products. In the case of Sentinel-2, both Level-1C and Level-2A products are available. This hub allows for a full-text search of datasets or querying the archive according to different parameters, such as acquisition date, cloud coverage etc. Data can be inspected online as preview before downloading. The Data Hub uses the Open Data Protocol (OData) which allows accessing the data directly through batch scripting via a HTTP/REST protocol (ESA, 2018a). Figure 2.11 shows a screenshot of the Open Access Hub.


Copernicus Open Access Hub
Insert search criteria...
Display 1 to 25 of 48 products.
Order By: Ingestion Date
0 products selected
25
page: 1 of 2

Request Done: (footprint:"Intersects(POLYGON((14.790985257133965 47.32350766275792,15.924291518163233

S2A MSI S2A_MSIL2A_20190118T095331_N0211_R079_T33TWM_...
Download URL: https://scihub.copernicus.eu/dhus/odata/v1/Prod
Mission: Sentinel-2 Instrument: MSI Sensing Date: 2019-01-18

S2A MSI S2A_MSIL2A_20190118T095331_N0211_R079_T33TWN_...
Download URL: https://scihub.copernicus.eu/dhus/odata/v1/Prod
Mission: Sentinel-2 Instrument: MSI Sensing Date: 2019-01-18

S2A MSI S2A_MSIL1C_20190118T095331_N0207_R079_T33TWM_...
Download URL: https://scihub.copernicus.eu/dhus/odata/v1/Prod
Mission: Sentinel-2 Instrument: MSI Sensing Date: 2019-01-18

S2A MSI S2A_MSIL1C_20190118T095331_N0207_R079_T33TWN_...
Download URL: https://scihub.copernicus.eu/dhus/odata/v1/Prod
Mission: Sentinel-2 Instrument: MSI Sensing Date: 2019-01-18

S2B MSI S2B_MSIL2A_20190116T100349_N0211_R122_T33TVN_...
Download URL: https://scihub.copernicus.eu/dhus/odata/v1/Prod

Figure 2.11.: Screenshot of the Open Access Hub.

The Sentinel Hub is an engine for large scale processing of remote-sensing data designed by Sinergise. it is capable of analyzing and displaying Sentinel, Landsat and other satellite data (Sinergise, 2018a). The hub allows for online inspection of all Sentinel-2 bands as well as different band combinations such as true color or custom combinations for different applications such as geology, agriculture, false color etc. In addition, a multitude of indices can be examined (Sinergise, 2018b). Figure 2.12 shows a screenshot of the

Sentinel Hub.

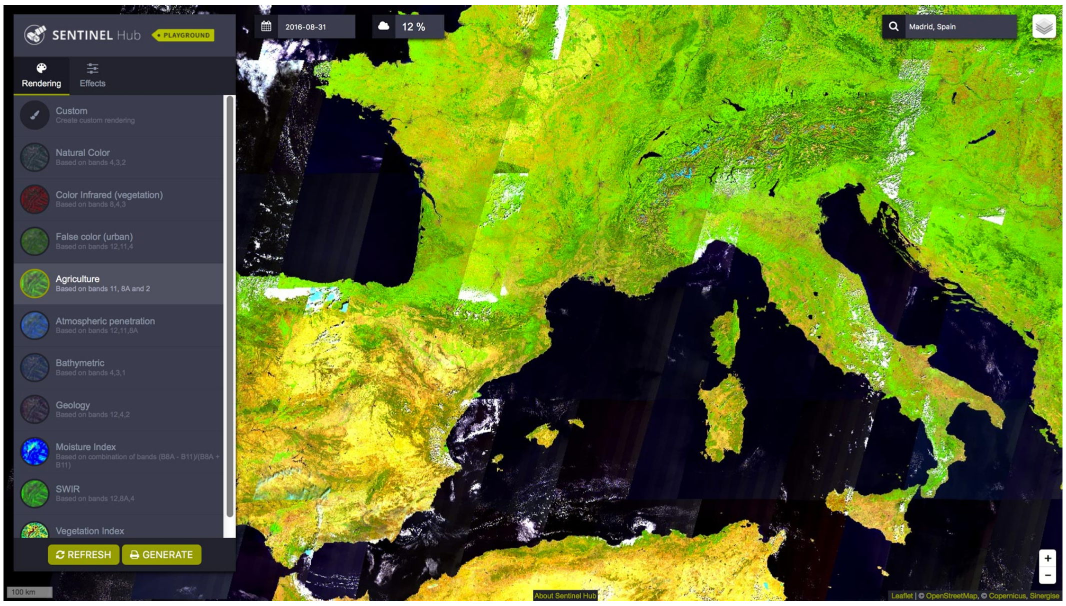


Figure 2.12.: Screenshot of the Sentinel Hub (Sinergise, 2018a).

The Sentinel Hub also features an Earth Observation (EO) Browser, which provides a complete archive of various different Satellites, including Landsat, Sentinel, Envisat, MODIS and others. It provides time-series capabilities which allow to monitor specific areas over long time periods. Land-cover data can be observed from 1984 onward (Sinergise, 2018c).

2.5. Summary

This chapter provided an introduction to the topics of remote-sensing and the basics of interactive data visualization. The uses and applications of remote sensing have been explained. It has been shown, that visualization of this high dimensional data is crucial for processing and understanding this huge amounts of data. After highlighting some of the most important steps when dealing with visualizations, a look was taken at related satellite projects and already existing work in the field of visualization and satellite data processing. As far as we know, no visualization framework utilizes the power of modern 3D-engines such as Unity. Such engines are capable of utilizing the vast rendering powers of modern Graphical Processing Units (GPUs), allowing for faster and more streamlined visualizations.
In the next chapter we will provide an introduction to the Copernicus project, the Sentinel satellites and the generation and structure of the datasets.

3. Dataset

This chapter provides insights on the Copernicus project, its Sentinel satellites and specifically the Sentinel-2 data offer as it is the primary data provider of datasets used in this project. After an introduction to ESAs Copernicus Project, the computation and uses of aforementioned vegetation indices is described in depth, because they are necessary for the description of Level-2A processing for Sentinel-2 image data. The provided format is described as well as the contents of each acquired dataset. Subsequently, the two main data offers, Level-1C and Level-2A are compared and discussed.

3.1. The Copernicus Project

The Copernicus program, formerly known as Global Monitoring for Environment and Security (GMES) initiative started in 1998 by the European Space Agency (ESA), the European Commission (EC) and the European Organization for the Exploitation of Meteorological Satellites (EUMETSAT). The Copernicus program is split into a service and a space component. The service component provides essential information for atmospheric, marine and land domains. The atmospheric monitoring component is aimed at providing data, which is essential for monitoring and observing climate changes, such as carbon dioxide levels, ultraviolet (UV) radiation levels and air quality. The land domain focuses on core mapping services to allow for observation of seasonal or annual changes for more directed policy integration concerning agriculture and regional development. Finally, the marine component is vital for water-based research such as oil and pollution tracking (Aschbacher and Milagro-Pérez, 2012, p. 3, 4). The space service of Copernicus is responsible for providing the actual data which is processed by the service component. Currently, the main focus lies on the following instruments:

- Synthetic aperture radar (SAR) sensors, which can be used under all weather conditions, including day and nighttime.
- Lower resolution (between 300m and 30m per pixel) optical sensors for agricultural and land monitoring purposes.
- High resolution (above 30m per pixel) optical sensors for accurate land coverage information
- Very high resolution (above 4m per pixel) optical sensors for urban emergency and security applications.
- Thermal radiometers for monitoring impacts of climate changes on land and ocean temperature.
- Atmospheric chemistry spectrometers for measurement of air quality and atmosphere composition.

ESA member states decided to invest in the building of this dedicated space infrastructure through a set of satellites, called the Sentinels, which are described in the next Section (Aschbacher and Milagro-Pérez, 2012, p. 5).

3.2. Current Satellites in Use

The Sentinels are the space component of the Copernicus program. Currently (Dec. 2018), there are seven satellites in orbit, six more are planned (ESA, 2018b).

3.2.1. Sentinel-1

Sentinel-1 is designed to image global landmasses, coastal zones, sea-ice zones, polar areas and shipping routes at high resolution. The two satellites, S1A and S1B offer a six-day exact repeat and conflict-free operation. The on-board SAR instrument operates in the C-band (3.95-5.8GHz) using horizontal and vertical polarization for transmitting and receiving (Fletcher, 2012a). Sentinel-1A was launched in April 2014, Sentinel-1B followed in April 2016 orbiting 180°apart from its identical twin (Esa, 2018b).

3.2.2. Sentinel-2

As specified by Fletcher (2012b), Sentinel-2 provides systematic global coverage of land surfaces from 56°S to 84°N, including coastal areas and selected calibration sites. The satellites have a temporal re-visit frequency of five days at the equator and provide images in 10m, 20m and 60m resolution with a field of view of 290km. The MultiSpectral Instrument (MSI) provides twelve spectral bands which extend from Very-Near Infrared (VNIR) to Short-Wave Infrared (SWIR):

- 10m Resolution: blue (490nm), green (560nm) and red (665nm)
- 20m Resolution: four narrow bands in the vegetation red-edge domain (75nm, 740nm, 783nm and 865nm) and two SWIR bands (1510nm and 2190nm)
- 60m Resolution: mostly used for atmospheric correction, such as aerosol retrieval (443nm), water vapor retrieval (945nm) and cirrus cloud detection (1375nm).

A visual representation of the bands is shown in Figure 3.1 (Fletcher, 2012b).

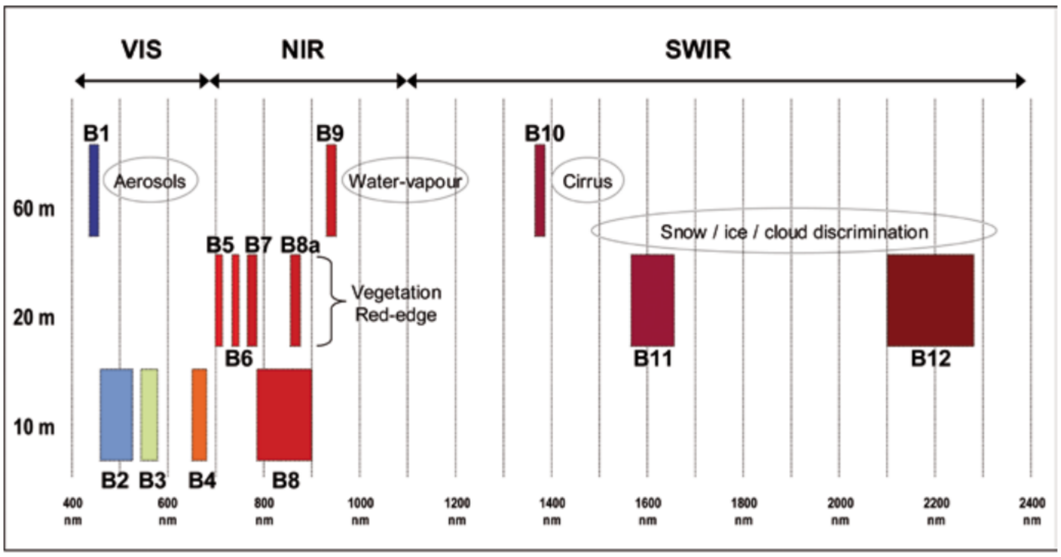


Figure 3.1.: Sentinel-2 bands visualized in the spectral band (Fletcher, 2012b).

The MSI has a mass of 290kg and provides a swath-width of 290km. Acquired data is labeled Level-0 and has to be radiometrically corrected. Atmospheric corrections and geometrical models have to be applied in order to use the data (Fletcher, 2012b). This will be further explained in Section 3.4. Sentinel-2A was launched in June 2015, Sentinel-2B almost two years later in March 2017 (Esa, 2018a).

3.2.3. Sentinel-3

According to Fletcher (2012c), the purpose of Sentinel-3 is the study of sea surface topography, such as wave height and wind speeds. Sea, ice and land temperature is measured as well as ocean color and land surface reflectance. This is accomplished by using five different instruments on each of the two satellites:

- Ocean and Land Color Instrument, an imaging spectrometer
- Sea and Land Surface Temperature Radiometer
- Synthetic Aperture Radar Altimeter
- Microwave Radiometer
- Global Positioning System (GPS) receiver and Laser-Retro-Reflector for precise orbit determination.

Sentinel-3 provides a re-visit time of four days. Sentinel-3A was launched in January 2016, Sentinel-3B followed in April 2018 (Esa, 2018c).

3.2.4. Sentinel-5p

Sentinel-5p is a precursor mission to the Sentinel-4 and Sentinel-5 program. The satellite was launched in October 2017 (Esa, 2018d), which will provide daily global atmospheric readings to deliver better forecasting on atmospheric constituents. Measured components include Ozone (O_3), nitrogen dioxide (NO_2), carbon monoxide (CO), UV levels and aerosols (Fletcher, 2012d).

3.3. Vegetation Indices

In order to understand the pre-processing of Sentinel-2 data, one has to understand the underlying algorithmic background. Since Vegetation Indices (VIs) are crucial for the processing of Sentinel Level-2A data, a basic explanation of their purpose is given in this section. VIs are an algebraic combination of spectral bands. They can reveal valuable information such as vegetation structure, leaf density, water content and even evidence of parasite attacks. This means that the index should be sensitive to the property of interest (Yengoh et al., 2015). More specifically, a vegetation index should maximize its sensitivity to biophysical parameters with a preferably linear response. It should normalize or model external effects such as sun angle, viewing angle and atmospheric effects for consistent comparisons. Furthermore, the index should normalize effects such as topography (slope) and soil variations. According to Jensen (2013), many indices make use of the inverse relationship between red and near-infrared reflectance, which is associated with healthy green vegetation (See Figure 3.2). The structure of leaves determines how plants interact with sunlight. The main pigments of vegetation, chlorophyll, carotenoids and water absorb specific wavelengths of energy. For example, green plants absorb most of the red and blue light, causing them to appear as green. Higher absorption means lower reflectance (Yengoh et al., 2015).

Since the NDVI and NDSI are especially important for deriving the Sentinel-2 Level-2A product, these two will be explained in more detail.

3.3.1. Normalized Difference Vegetation Index

The Normalized Difference Vegetation Index (NDVI), developed by Rouse Jr et al. (1974), is the the difference between the near-infrared band and the red band, divided by the sum of the two bands (Yengoh et al., 2015).

$$NDVI = \frac{\rho_{NIR} - \rho_{RED}}{\rho_{NIR} + \rho_{RED}}$$

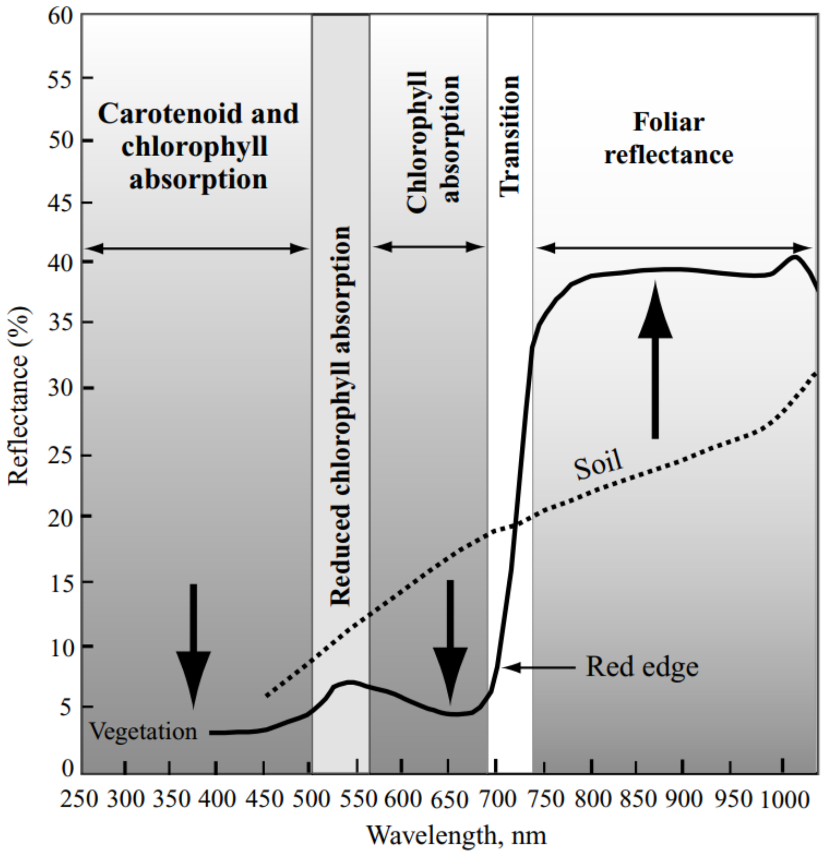


Figure 3.2.: Chlorophyll strongly absorbs light between 400nm and 650nm. In contrast, the reflectance values in the near infrared (700nm - 1000nm) are very high. This builds the base for many vegetation indices. Adapted from Jensen (2013).

Adapted to Sentinel-2 bands, the equation reads as follows:

$$NDVI = \frac{B08 - B04}{B08 + B04}$$

The NDVI uses the fact that green vegetation reflects less visible light and more near-infrared. Sparse or less green vegetation reflects a greater amount of visible and less NIR light. The NDVI values range between -1.0 and $+1.0$, where only positive values correspond to vegetation. The higher the values, the greater the chlorophyll content of the selected pixel (Yengoh et al., 2015).

The NDVI has been used to identify a range of phenology metrics which describe plant-life cycles and show how these are influenced by seasonal and annual variations in climate. The duration of the of photosynthetic activity can be interpreted as the length of the growing season (Yengoh et al., 2015).

3.3.2. Normalized Difference Snow Index

The Normalized Difference Snow Index (NDSI), originally developed for the Landsat project by Hall, Riggs, and Salomonson (1995), is used to map snow, which is highly reflective in the visible part of the spectrum and highly absorptive in the near or short infrared part. It is defined by the following equation:

$$NDSI = \frac{\rho_{VIR} - \rho_{SWIR}}{\rho_{VIR} + \rho_{SWIR}}$$

This can be adapted to be used with Sentinel-2 bands:

$$NDSI = \frac{B03 - B11}{B03 + B11}$$

As described by Jensen (2013), especially when observing scenes where both snow and clouds are present, the NDSI plays a vital role of distinguishing one from another. When looking at the reflactance of snow and clouds in the VIR and NIR spectrum, both approximately reflect the same percentage. However, as can be observed in Figure 3.3, this changes drastically above wavelengths of $1.2\mu m$, where clouds continue to reflect lots of energy, while snow increasingly absorbs it. This property is used to differentiate cloud and snow in generation of the Sentinel-2 cloud- and snow-masks, which will be described in Section 3.4.2.

3.4. Sentinel-2 Image Data

This section describes the various different processing levels involved in the generation of an end-user friendly product.

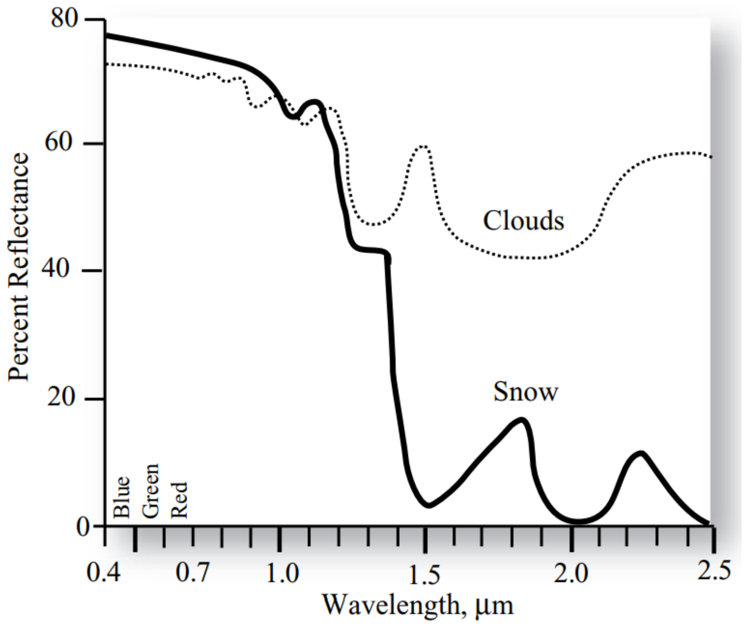


Figure 3.3.: Reflectance of snow and clouds in the spectral range of 0.4μm to 2.5μm (Jensen, 2013).

3.4.1. Level-0 and Level-1 Product

As described by Suhet (2015), the continuous acquisition of Sentinel-2 image data is called a datatake. At maximum the datatake has a length of 15.000km. The datatake can be divided into smaller sub-takes called granules or tiles. The raw image data is transferred to ground stations in so called Instrument Source Packets (ISP). This is called the Level-0 product. It is the basis for the later processing and has a size of 25km across track by 23km along track. An average orbit contains approximately 3500 of such products. To allow further processing, annotated ancillary source packets are stored within those products, containing meta-data such as time correlation information and thermal data. After decompressing the acquired data, a geometric model is developed which allows to spatially reference each pixel in the image. This represents the processing Level-1A (Suhet, 2015, p. 44).

Level-1B processing includes dark signal and cross-talk correction as well as the identification of defective pixels. High spatial-resolution bands are deconvoluted and denoised (ESA, 2018c). In addition, the geometric model is refined using ground control points. This refined model is appended to the product but not applied (Fletcher, 2012b, chap 7. p 43).

For the Level-1C product, a Digital Elevation Model (DEM) is used to project the image data in cartographic coordinates (UTM/WGS84). A single Level-1C product covers an area of 100x100 km (illustrated in Figure 3.4). This allows for mapping each pixel in the projected image back to its native geometry coordinates in the raw source image. For each spectral band, the Top-Of-Atmosphere (TOA) reflectance is calculated (ESA, 2018d). This is a unit-less ratio that describes the amount of light leaving a certain area to the amount of light received by that area (Fletcher, 2012b, chap 7. p 43). In addition, this product contains cloud, land and water masks. The cloud mask classifies if a certain pixel contains no clouds, cirrus or opaque clouds.

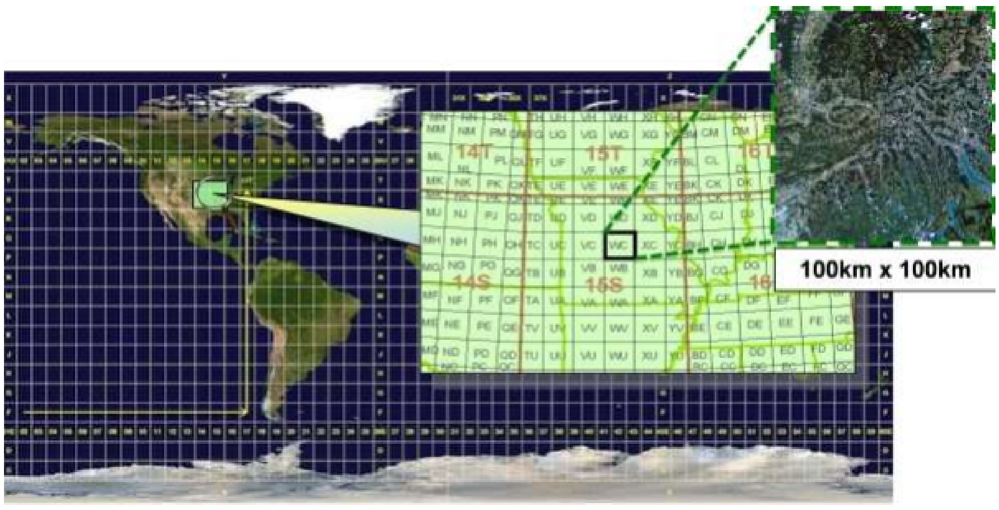


Figure 3.4.: Product tiling (Suhet, 2015).

3.4.2. Level-2A Product

The Level-2A product provides Bottom-of-Atmosphere (BOA) reflectance values. The conversion from TOA to BOA is done by an atmospheric correction of the Level-1C reflectance values (Suhet, 2015, p. 48). This correction relies on a model developed by Mayer and Kylling (2005), which builds a lookup table (LUT) of sensor specific functions such as direct and diffuse transmittances, solar fluxes and albedo. Those and other parameters account for most atmospheric conditions. This LUT has a resolution of 0.6nm. Applied to the TOA reflectances, BOA reflectances are calculated (ESA, 2018e). An overview of the Level-2A classification process can be seen in Figure 3.5. It is applied to Level-1C data resampled to 60m spatial resolution. Additional outputs of the Level-2A processing are described in the subsequent sections.

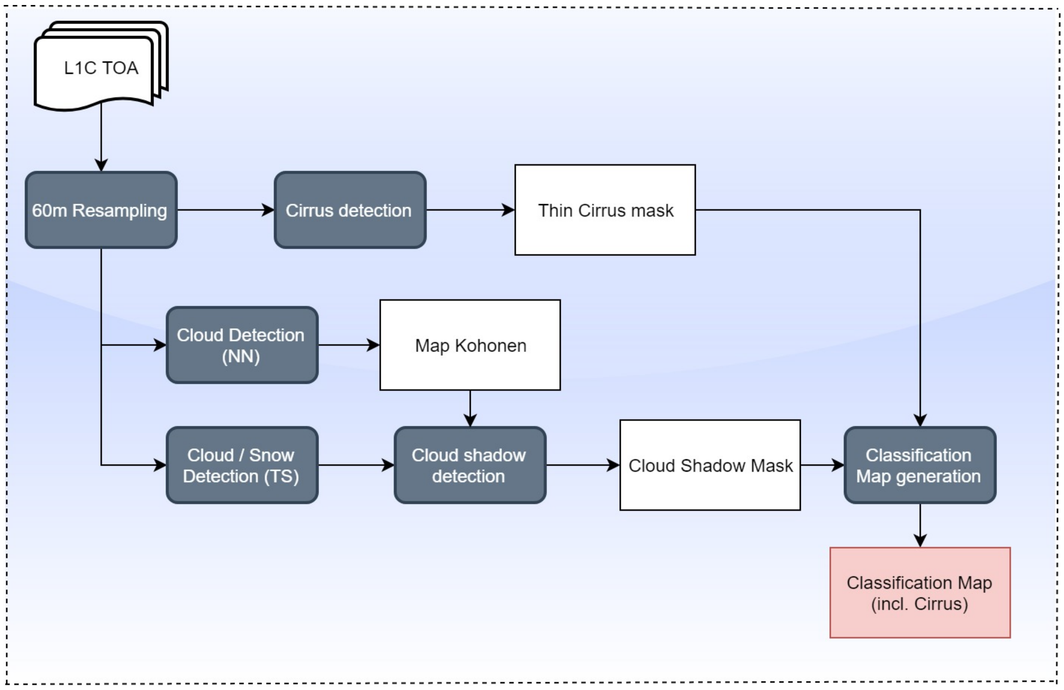


Figure 3.5.: Classification modules for the SCL. (NN: Neural Network, TS: Threshold algorithm) Adapted from ESA (2018e).

Cloud/Snow Detection

These masks contain the probability of cloud or snow cover for each pixel, respectively. In order to obtain these probabilities, a series of threshold filters is applied to different bands (see Figure 3.6).

The following description summarizes the classification algorithm described by ESA (2018e):
First, a threshold is applied to the brightness values of B04 (red). Values outside of the thresholds are considered cloud-free and are set accordingly in the cloud mask. Otherwise, the NDSI is applied. As mentioned before, the NDSI is the radio between VIR (B03) and SWIR (B11). Snow absorbs light in the SWIR range but reflects VIR. Clouds are generally reflective in these bands. Pixels with strong negative values are considered cloud-free. Pixels above a certain threshold are considered potentially cloudy and proceed to the next stage.

In this decision stage, the primary goal is to determine if the pixel in question represents snow. This multi-stage filtering for B02, B03, B08 and B11 is only applied to pixels for which snow has been detected during the last ten years. After this process, snow boundary zones have to be processed. This removes false cloud detection on the borders of snow regions, where snow and ground could be detected as clouds. Those regions are dilated and threshold filtered in B12. Pixels exceeding that threshold are classified as "not snow". For those pixels that have been classified to not represent snow, NDVI filtering is applied. This removes highly reflective vegetation. Pixels which exceed a certain threshold are considered not to be clouds. All other pixels proceed to the next stage. This step identifies bare soil pixels. First, the pixels are thresholded using the ratio between B02 and B11. This ratio is lower for soil than every other scene class, including clouds. With the same ratio, water can be detected as the reflectance ratio for water is higher than any other class, again, including clouds. The last step is to identify rocks and sands in deserts. Rocks, as well as sand, exhibit higher reflectance in B11 than in B08. The opposite is true for clouds. This marks the final step in the cloud classification process. Pixels that have a high enough cloud probability after this step can be spatially filtered. This compensates for slight miss-registrations of bands and helps to combat miss-classified pixels on the borders of rivers and shorelines. To distinguish between cirrus and other clouds types, the reflectance of B10 ($1.375 \mu m$) is thresholded. This detection relies on the strong water vapour absorption at $1.38 \mu m$. If the value is below the thick-cloud threshold and above the clear-sky threshold, a cirrus cloud pixel is detected.

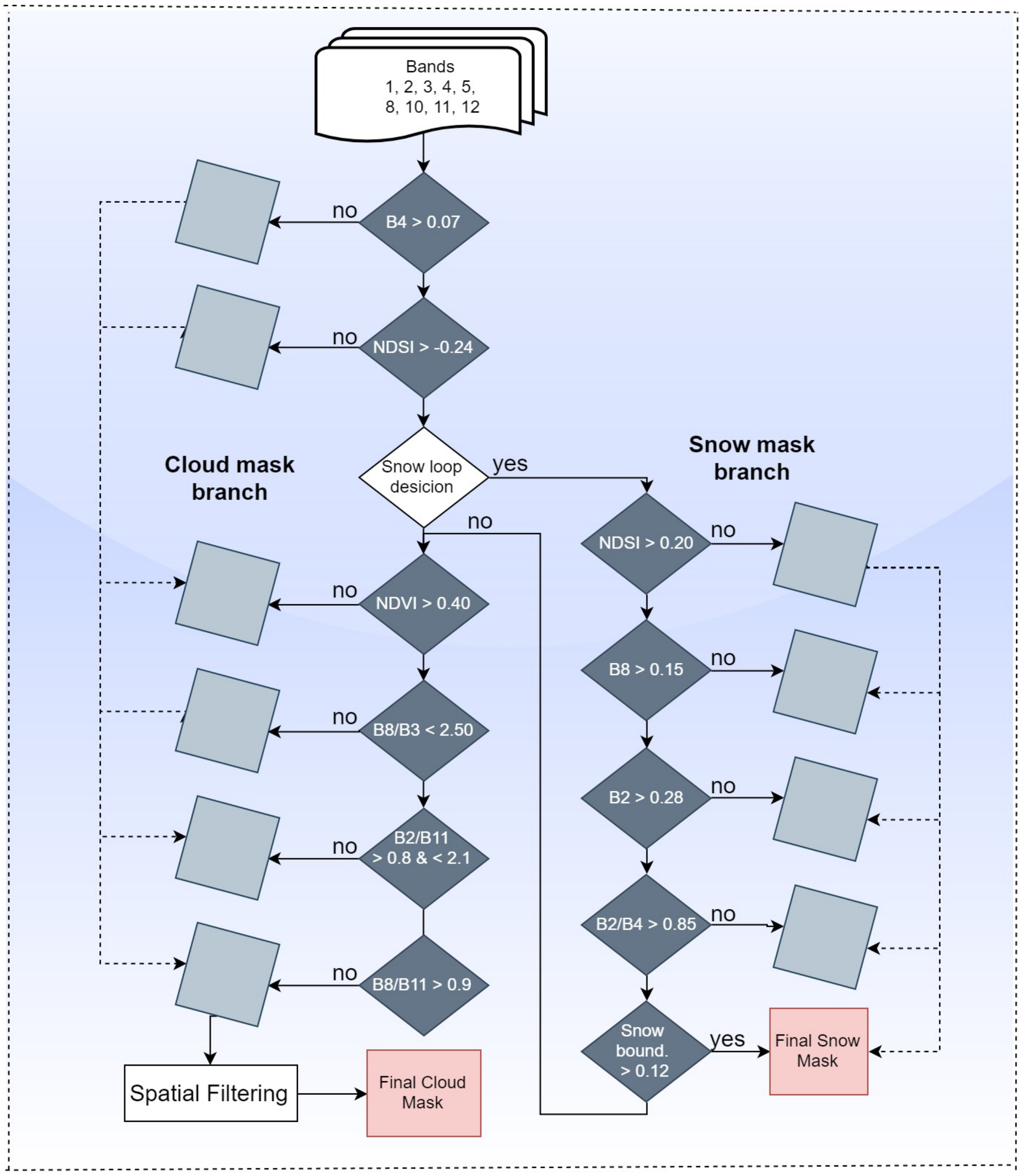


Figure 3.6.: Level-2 cloud/snow detection algorithm sequence. Adapted from ESA (2018e).

Cirrus Cloud Detection

As explained by Liou (1986), cirrus clouds contain a significant amount of large, nonspherical ice crystals. They are optically thin and non-black, regularly covering about 20% of the globe. In order to detect cirrus clouds, B10 ($1.375\mu m$) is threshold filtered. This detection relies on the strong water vapour absorption of cirrus clouds. Only if the pixel is above a certain clear-sky theshold but below the thick-cloud threshold, a cirrus cloud is detected. In addition, the pixels in questions are cross-checked with the cloud mask obtained by the aforementioned detection process. Only if the cloud probability in the cloud-mask is below a certain threshold, the pixel is classified as containing cirrus clouds (ESA, 2018e).

Cloud Shadow Detection

This map is built by multiplying the "geometrically probable" cloud shadows, which are derived from the final cloud mask, the suns position and cloud height distribution and the "radiometrically probable" cloud shadow derived from the neural network "dark areas" classification (which are based on a Kohonen map[1]). Only if the pixels exceed a certain probability threshold, a classification as "cloud shadow" is applied. Otherwise, the pixel is classified as either a medium or high probability clouds, depending on the exact thresholding result (ESA, 2018e).

Scene Classification

Level-2A products include an enhanced scene classification (SCL) which classifies each pixel in the scene. As seen in Table 3.1, it includes four different classes for clouds and classes for shadows, vegetation, soil and others.

[1]Self-organizing map, a special type of artificial neural network. See Kohonen (1995).

It is generated from the TOA reflectance values of the Level-1C product and available as 60m mask. It uses the cloud and snow detection algorithm outlined in the previous subsection. Cloud shadows are detected using the geometrically probable shadows, derived from the final cloud mask, taking into account the cloud height distribution and the sun position (ESA, 2018e). The scene classification is provided in a resolution of 60m. In addition, the product contains an up-scaled 20m version. Users can generate Level-2A data from Level-1C datasets themselves, using the stand-alone Sen2Cor processor (ESA, 2019).

Label	**Classification**
0	No Data
1	Saturated or Defective
2	Dark Area
3	Cloud Shadow
4	Vegetation
5	Bare Soil / Not Vegetated
6	Water
7	Cloud: Low Probability
8	Cloud: Medium Probability
9	Cloud: High Probability
10	Cloud: Thin Cirrus
11	Snow

Table 3.1.: Scene classification values, color coded.

3.5. Dataformat

As of December 2016, the naming convention of Sentinel-2 scenes is as follows (ESA, 2016):

```
<MID>_MSI<PL>_<DTST>_N<PBN>_R<ON>_T<TID>_<PD>.SAFE
```

- `<MID>`: Denotes the Mission ID. Either `S2A` or `S2B`.
- `<PL>`: Identifies the product level. Either `L1C` or `L2A`.
- `<DTST>`: The start time-stamp of the datatake, in `YYYYMMDDTHHMMSS` format, with the `T` separating date and time.
- `<PBN>`: Used processing baseline version, in major and minor version number, dot seperated, for example `02.04`.
- `<ON>`: Relative orbit number (001 - R14).
- `<TID>`: Identifies the tile (see Figure 3.4) Tiles are uniquely identified by two digits and three letters.
- `<PD>`: Product discriminator, uses the same format as `DTST` and is used to differentiate between different end-user products from the same datatake.

Hence the filename:

```
S2A_MSIL2A_20170105T013442_N0204_R031_T53NMJ_20170105T013443.SAFE
```

describes a Level-2A product which has been acquired on the 5th of January 2017 at 01:33:42, during relative orbit 031, containing data from tile 53NMJ, processed with baseline version 02.04 on the 5th of January 2017 at 01:34:43. This various identifiers may be used by the operator to select and query the available datasets according to his needs. Especially the sensing timestamp is of vital importance since in many remote sensing applications, the observation of an area in a given time frame is a requirement.

As described by ESA (2018g), Sentinel-2 products are made available in the Sentinel SAFE (.SAFE) format. This package includes image data in the JPEG2000 format, quality indicators such as defective pixel masks and

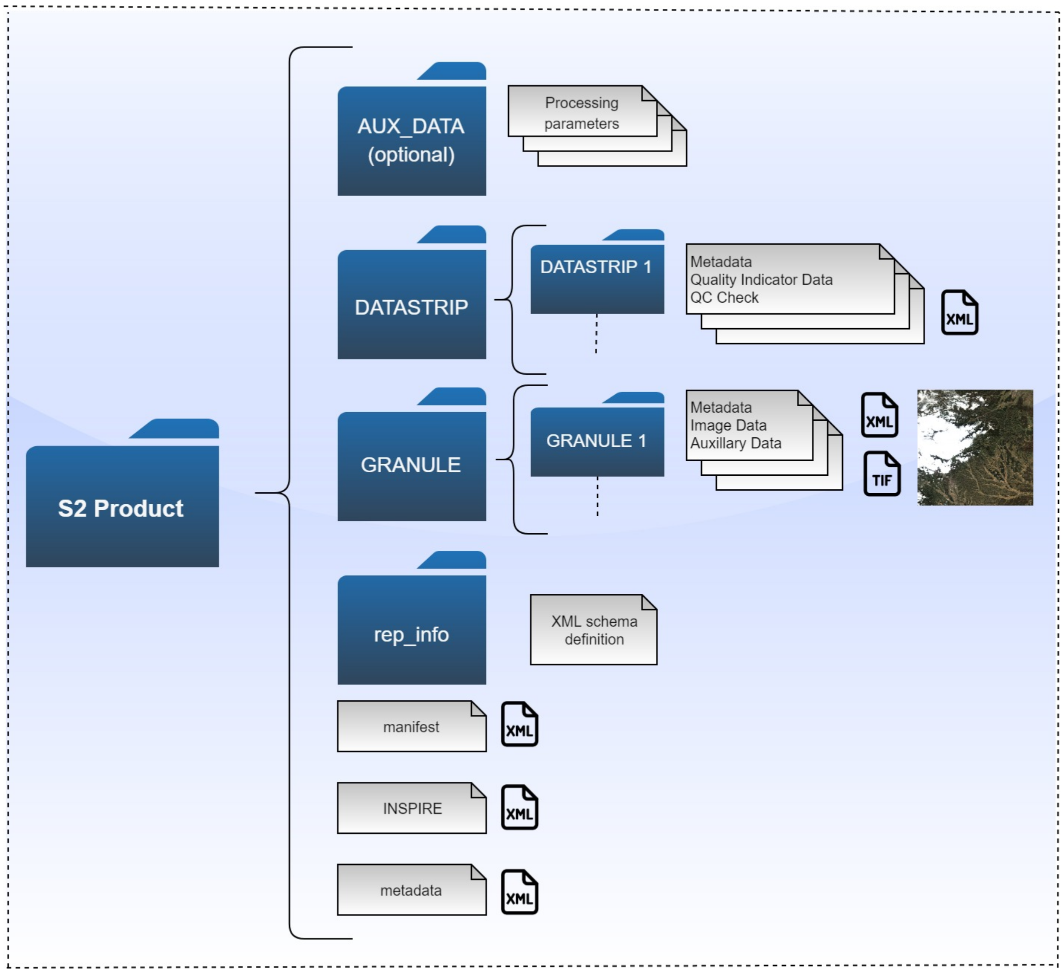


Figure 3.7.: Sentinel-2 physical format description, based on ESA (2018g).

auxiliary metadata. The structure (see Figure 3.7) of the container is the same throughout different processing levels.

Specifically the Level-2A product contains the following data:

- **Manifest/Metata:** XML file which holds general product information (acquisition date/time, location, etc.)
- **Quicklook:** a (low-resolution) preview image in JPEG2000 format.
- **Subfolder for Auxillary Data:** Processing innformation and parame-

ter maps.

- **Subfolder for Datastrip:** Level information and quality control (QC) Reports.
- **Subfolder for Granule:** Actual image data in geocoded JPEG2000 format. Includes auxillary data such as cloud and snow masks.

3.6. Summary

In this chapter, we have outlined the history and goal of the Copernicus Project. After a brief explanation of each of the satellites, vegetation indices have been explained. They are useful in classifying various different properties and necessary for the processing of Sentinel-2 data. We then analyzed the various different data products which Sentinel-2 has to offer. Publicly available are only the Level-1C and Level-2A product. We decided on using pre-processed Level-2A products for this work, as they already provide BOA reflectance values, thus removing the necessity of batch-processing Level-1C data. The packaged Level-2A products are uncompressed .zip files (the contained image data is wavelet-compressed) and contain all necessary information to allow geo-referencing the scene. The provided scene classification in 20m resolution can be used for masking specific pixel classes, such as clouds or snow. Not all bands are available in full 10m resolution, making an on-the-fly resampling necessary (see Table 3.2). The next chapter describes the design of the implemented system as a primer to the implementation chapter.

Band Name	10m	20m	60m
B01	🟥	🟥	🟩
B02	🟩	🟨	🟨
B03	🟩	🟨	🟨
B04	🟩	🟨	🟨
B05	🟥	🟩	🟨
B06	🟥	🟩	🟨
B07	🟥	🟩	🟨
B08	🟩	🟥	🟥
B8A	🟥	🟩	🟨
B09	🟥	🟥	🟩
B11	🟥	🟩	🟨
B12	🟥	🟩	🟨
SCL	🟥	🟨	🟩

Table 3.2.: Level-2A band content. Red: not provided. Yellow: provided in non-native resolution. Green: provided in native resolution.

4. Design

Before delving into the implementation chapter, we provide an overview of the design strategy surrounding the *Sentinel-2 Explorer* framework. Implementing such a complex system requires identifying the audience which is the primary target group for this application. Naturally, a system designed for educational use in schools or universities has to implement different use-cases and features than a system targeted at highly trained professional remote sensing specialists. After identifying the user group, functional and non-functional requirements need to be defined. This serves the purpose of differentiating between specific user needs and overall features such a system has to integrate. The chapter concludes by describing the chosen design strategy and the tools used to implement the *Sentinel-2 Explorer*.

4.1. Core Functionality

Before selecting the target user group, the purpose and core functionality of the *Sentinel-2 Explorer* has to be defined. Of course, a system developed within the scope of a master thesis can not aim to be a replacement for a full fledged GIS. Instead, the *Sentinel-2 Explorer* aims to be a system for easing the life of people that work with Sentinel-2 satellite data. Thus, the main purpose of this framework is to provide an intuitive and easy way to retrieve and inspect datasets from this satellite.

4.2. Target User Group

The target group clearly includes scientists and specialists who are interested in the Sentinel-2 satellite, but are not necessarily experts in the field of processing data on this specific remote-sensing platform. This is vital, since different experts in fields such as mapping, climate-research and geology have to work with various satellites and therefore can not be expected to be experts on each and every specific satellite. The included tools for data analysis should not be designed to be only useful to a handful of people, but be meaningful enough to be of interest for everyone. For example, providing an appealing visualization of information that is otherwise hidden away in metadata not only reduces efforts of scientists by manually searching and retrieving this data but also allows processing larger amounts of data in a shorter time span.

4.3. Requirements Analysis

As described by Paetsch, Eberlein, and Maurer (2003), this analysis checks given requirements for necessity (*"Is feature X necessary?"*), completeness (*"Are all constraints satisfied?"*), consistency (*"Are there any requirements that contradict each other?"*) and feasibility (*"Is feature Y implementable within the given time/budget?"*). Requirements can be partitioned in functional and non-functional requirements. As defined by Koelsch (2016), functional requirements describe what functions a system (software) should perform, while non-functional requirements define how the system should behave and what constraints the system has to follow.

4.3.1. Functional Requirements

As already mentioned, functional requirements specify, what a system should be able to perform. Requirements in this category concern themselves with what users want to do with the system or which algorithms have to be implemented. The following list provides a collection of functional requirements for the Sentinel-2 Explorer:

1. Users should be able to download Sentinel-2 satellite data from the Copernicus Open Access Hub:
    a) by optionally specifying a desired cloud coverage range,
    b) by optionally specifying a desired snow coverage range,
    c) by optionally specifying a desired no-data coverage range,
    d) by optionally specifying a date range that returns datasets that have been acquired during this time-period,
    e) by optionally specifying a geographic location (such as a Sentinel-2 tile),
    f) by optionally including/excluding datasets obtained by the S2A or S2B satellite.
2. Geographic querying of available scenes should be possible with a visual selection system.
3. Querying should be possible without knowledge of the employed backend querying system used on Copernicus servers.
4. Queried scenes should be displayed together with their respective metadata (date, cloud coverage, etc.)
5. The user should see a preview image of the scenes returned by the performed query.
6. Datasets that are returned by a query and are already downloaded should be visually highlighted.
7. Users should be able to exclude specific datasets from download.
8. If the user wants to download multiple datasets, pending downloads should be queued and downloaded consecutively.
9. Downloads should be cancelable at any given time.
10. In case the user is downloading multiple datasets, it should be possible to cancel all downloads at once.
11. The download queue and progress should be visible to the user on demand.
12. Downloads should proceed even if the user leaves the downloading component of the application.
13. Temporary files and download-fragments should be cleaned up in case a system error occurs.

14. The original datasets should be stored on the file-system in its original format to provide access for other applications.
15. Download and intermediate data storage directories have to be configurable by the user.
16. It is required, that the system provides a visualization of all bands in their respective maximum resolution.
17. The system is required to allow zooming and moving the viewport to allow up-close inspection of AOIs (Area of Interest).
18. Inspection of downloaded scenes should be possible.
19. The user should be able to inspect multiple scenes concurrently.
20. Inspected scenes should be displayed with respect to their geographic location. Loaded scenes on the same geographic position should be loaded on top of each other with the possibility of temporarily hiding individual scenes.
21. The system is expected to display basic information of a loaded dataset, such as:
    a) Name
    b) Acquisition Date
    c) Geographic coordinates / projection information
    d) Cloud coverage percentage
    e) Snow coverage percentage
    f) No Data coverage percentage
    g) Vegetation coverage percentage
    h) Bare Soil coverage percentage
22. The user is able to choose displaying various presets of band combinations, such as:
    a) True-Color
    b) False-Color IR
23. The user is able to specify a custom band combination.
24. For each individual scene, the user should be able to mask out all available pixel classes.
25. Masked out pixels should either be transparent or set to a distinguishable color.
26. The user should be able to load and close scenes from within the inspection tool, without loosing the current state of the application.

27. The displayed value-spreading of the scenes should be calculated according to mean and standard deviation for each individual band.
28. For each displayed band, the user should be able to mask-out values exceeding a custom threshold. If the pixel values of one band exceeds or falls below the set the threshold, the pixel becomes transparent.
29. For a visual guidance when masking pixels according to thresholds, the user should see the mean and standard deviation of each visible band.
30. The user should be able to inspect the pixel value of each selected band at a given mouse coordinate, either
    a) by clicking on a specific point in the scene. In this mode, a visual indicator should highlight the specific position in the scene,
    b) or by continuously moving the mouse across the scene.
31. The system should display pixel or geographic coordinates of the inspected position in the scene.
32. If scenes overlap at the point of interest, the values of all scenes that cover this point should be displayed.
33. If the user loads more than one scene, the changes of vegetation, soil and other pixel classes should be displayed, revealing the changes in the temporal dimension.
34. As there are many classes, the user should be able to hide the temporal changes for selected graphs.
35. the user should be able to hide GUI elements, allowing for a larger inspection area.

4.3.2. Non-Functional Requirements

In contrast to functional requirements, which specify what the system should do, non-functional requirements specify how the system should accomplish this tasks. This includes attributes such as efficiency, effectiveness, fault tolerance and performance. The non-functional requirements for the *Sentinel-2 Explorer* are defined as follows:

1. The framework should be easy to use for both experts and people with little experience in remote-sensing.

2. The internal dataset structure (metadata, filename conventions, etc.) should be irrelevant to the desired operations of the user.
3. The framework should cache intermediate data (such as scene previews) to reduce load times and conserve bandwidth.
4. The system is not permitted to alter or modify the original datasets. This includes adding any kind of files to the datasets.
5. All generated intermediate data (with the exception of the datasets themselves) should be stored separately from the datasets.
6. Loading a scene should not exceed ten seconds.
7. Data integrity should be preserved in the event of a crash or network outage.
8. The system has to be able to display multiple (at least five) datasets concurrently without compromising speed or stability.
9. If a band is not available in the resolution, the system should perform an automatic resampling to the desired resolution.
10. Resampling should only occur where necessary.
11. Resampling should not employ any interpolation method to guarantee accurate display of the dataset.
12. Switching between band combinations should be possible with loading times less than five seconds.
13. The system is not allowed to use third party software or libraries that are not available freely for educational or academic use.
14. The system should be deployable under *Microsoft Windows 10* and *Debian Linux* operating systems.
15. The system has to be implemented in the time-scope of a master thesis.

4.4. Design Strategy

In order to meet all imposed requirements, a base framework which is capable of handling large quantities of (image)-data is necessary. Since various image-processing tasks have to be implemented, the use of a computers graphical-processing unit (GPU) is beneficial. There are various frameworks available, however, Unity3D was chosen. Unity 3D[1] is a modern real-time

[1]Unity: https://unity.com/

3D engine and development platform. The Unity Editor is a powerful tool for designing and implementing game logic. The Editor features tools for animation, lighting, optimization, physics, audio and scripting support. The engine is developed in C++ and uses C# as scripting language for implementing logic and custom behaviour. Due to Unity's supports for all major native Graphics APIs such as DirectX12, Vulcan, OpenGL and OpenGL-ES, developed products can be deployed on Windows, Linux, Mac, WebGL and mobile platforms (Unity, 2019). Unity employs the concept of scenes, which are environments for a specific element in the application (i.e. a level in a game). The *Sentinel-2 Explorer* will employ three such scenes, depicted in Figure 4.1.

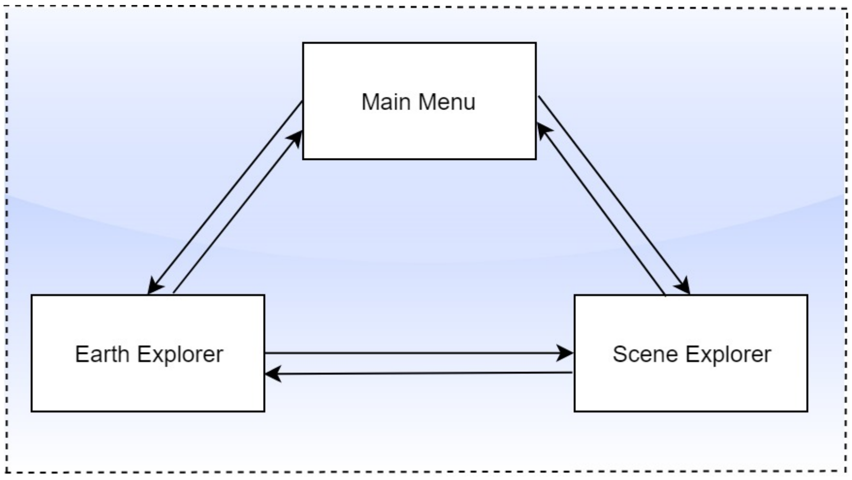


Figure 4.1.: The three scenes of the Sentinel-2 Explorer.

The functional requirements imply two main tasks for the *Sentinel-2 Explorer*: The download and the analysis of satellite data. Downloading and analysis are vastly different tasks, requiring distinct user interfaces and data models, hence two scenes are designed. The user is able to switch between all three scenes at any given time.

4.4.1. Earth Explorer

This scene is focused on querying and downloading Sentinel-2 satellite data. The Copernicus Open Access Hub[2] is the primary data provider of the Sentinel-2 data offer. The hub uses the OpenSearch[3] and OData[4] APIs to allow for batch-scripted data access of third party software.

OpenSearch

OpenSearch is used to query the internal Open Access Hub database for results according to specified search parameters. For this, OpenSearch exposes a REST API, which uses the following format:

```
<hostname>:<port>/search?start=<offset>&rows=<size>q=<query>
```

Each of the format specifiers is explained below:

- `<hostname>`: The service root of the server.
- `<port>`: Optional port specifier.
- `<offset>`: Specifies the starting position from which results are shown.
- `<size>`: Specifies the amount of results shown, at most 100.
- `<query>`: Search parameters.

A query returns at most 100 results at once. In order to access consecutive results, the same query has to be performed multiple times, with the `<offset>` and `<size>` parameter adjusted accordingly. The actual search query is defined by the segment trailing the q= identifier and can be used for filtering results according to user specified parameters. A complete request may look as follows:

```
https://scihub.copernicus.eu/dhus/search?start=0&rows=100&q=
(platformname:Sentinel-2) AND beginposition:
[2019-01-01T00:00:00.0000Z TO NOW] AND
cloudcoverpercentage:[0 TO 10] AND filename:*L2A*33TWN*
```

[2]https://scihub.copernicus.eu/

[3]https://github.com/dewitt/opensearch

[4]https://www.odata.org/

This query retrieves all Sentinel-2 products with an acquisition time between 01.01.2019 and the current day with a maximum cloud coverage of 10%. In addition, only Level-2A products of the tile `33TWN` are returned. A full list of all available search parameters is listed on the *Science Hub User Guide* website[5].

To provide a more convenient way of performing queries, a user interface tailored to create queries has to be designed. All relevant query parameters, defined by the functional requirements in the previous section have to be exposed to be easily configurable by the user. Since the user may want to specify a Sentinel-2 tile, a visual system for finding tiles has to be designed. This will be accomplished by overlaying a cylindrical earth map (see Section 5.1.2). Upon clicking any position on this map, the system will retrieve the corresponding tile that is covering this point. The tile name is then visualized and appended to the query.
The result of the query is a response in `xml` format, containing entries for each returned dataset, along with high-level metadata. This can be used to visualize query results without needing to download actual data. Each entry is visualized by a list-entry in a scroll-able GUI element, along with a low-resolution true-color quicklook (for fast visual inspection). The actual downloading of data is performed with the help of the OData API, described in the next section.

OData

The OData API is a data access protocol that allows for querying search results returned by the OpenSearch request. These entries contain, along with the aforementioned metadata, URIs[6] for downloading actual data associated with the dataset. OData enables the download of specific products contained in the set, such as a quicklook, specific bands or the entire dataset. Each product is identified by a unique ID which allows access to all resources contained within this dataset. To download the actual data, all OData requests have to be made to the following base URL:

[5]https://scihub.copernicus.eu/userguide/FullTextSearch
[6]URI: Unique Resource Identifier

```
https://scihub.copernicus.eu/dhus/odata/v1/
```

Downloading the complete SAFE archive is achieved by appending the following query to the base URL:

```
Products('<id>')/$value
```

where `<id>` denotes the respective dataset ID. Individual resources in the dataset may be accessed by designating them in a subquery. The query below would be used to retrieve the quicklook image associated with the dataset:

```
Products('d080e4b9-e537-4c03-8aa8-6f484cfc542c')
/Products('Quicklook')/$value
```

4.4.2. Scene Explorer

This scene focuses on the inspection and analysis of downloaded products by implementing the visualization pipeline from Section 2.3.1. The dataset is displayed by rendering given bands as texture onto a plane, whereby three bands are combined to create an RGB image. It is up to the user, which bands he wants to display. Since the drawing process should take as little time as possible, the task is performed by a GPU compute shader which can be used for arbitrary calculations and makes use of the GPUs inherent parallelism (Wolff, 2018). Recalling Section 2.3.1, Freitag and Loy (1999) argue, that a multi-resolution approach is useful when dealing with large quantities of data. Therefore, the input bands should only be loaded in the lowest necessary resolution. Since the resulting image is drawn onto a plane in an orthographic view-space, it is trivial to check which resolution is necessary to satisfy the argument from Heckbert and Garland (1994), that the coarsest level of detail which looks identical to the original dataset is appropriate. Accomplishing this makes it necessary to calculate the size on the screen that is occupied by the rendered image. If this size exceeds the lowest resolution level of 60m ($1830px * 1830px$), bands are reloaded in 20m resolution ($5490px * 5490px$). If the occupied space exceeds the 20m resolution, 10m bands ($10980px * 10980px$) are loaded. This works both ways, hence if the plane size falls below the respective resolution, the bands are reloaded in the next lower level of resolution. In addition, only portions of the image that are actually visible should be drawn. This makes it necessary to calculate the region which the user currently observes, disabling rendering on invisible areas. This further increases performance and reduces load on the GPU and overall system memory since less data has to be loaded. The output image is calculated by a shader program executed on the GPU, marking it the main component of both the mapping and the rendering stage in the visualization pipeline. The image has to be recalculated each time the resolution changes, or if the user changes settings which determine what pixel classes are to be masked out. The image has to be re-drawn in real time, since the effect of changing values should give immediate feedback to the user. To accomplish this task, the shader needs the following inputs:

- Three bands in uncompressed 16 bit unsigned integer format.
- Mean and standard-deviation values for each band.
- Upper and lower value threshold for each band.
- Scene classification band in 8 bit unsigned integer format.
- List of pixel classes which have to be masked.
- RGB color values which denote the output color of masked pixels.

The output of the shader is a four-channel 32 bit unsigned integer image. (8 bit for each channel: red, green, blue and alpha (transparency).

The shader has four components, each devoted on performing a specific task:

- **Threshold check**: Mask pixels which exceeds or fall below user-set thresholds.
- **Spread computation and Resampling**: Apply individual mean/standard-deviation spread to all bands and convert pixel values to 8-bit output format.
- **Scene Classification Masking**: Mask all pixels classes according to user needs.

Threshold Check

According to the functional requirements, the user should be able to set an upper and a lower threshold for each displayed band. Values exceeding or falling below that value should be rendered transparent. If the value of one bands exceeds or falls below its respective threshold, the entire pixel should become invisible (see Figure 4.2). Therefore, this component only influences the alpha component of the resulting image. This feature is useful for manually masking out pixels which fulfill certain reflectance criteria and have possibly been falsely classified by the Sen2Cor algorithm.

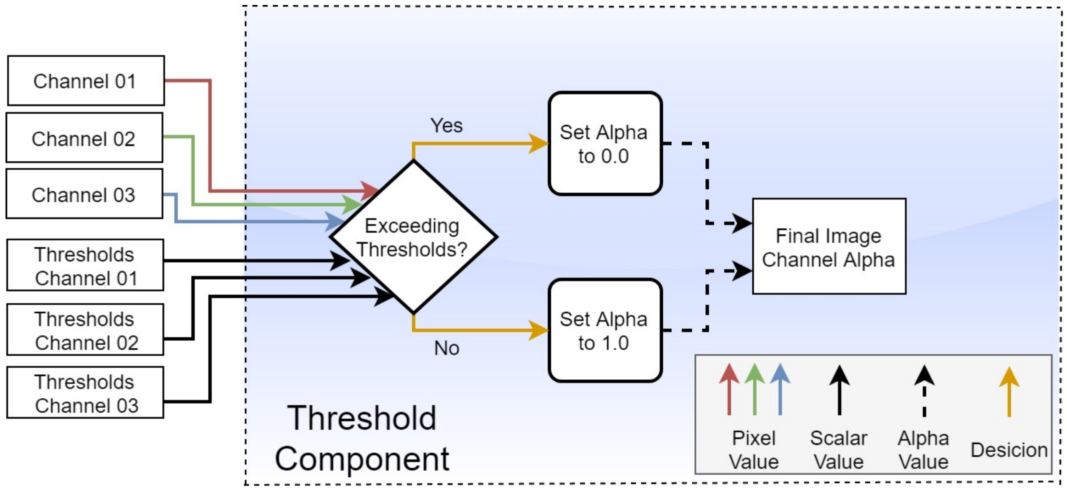


Figure 4.2.: Threshold component of the shader.

Spread computation

Sentinel-2 image data is delivered in 16 bit unsigned integer format, however, the radiometric resolution of the MSI is only 12-bit (Suhet, 2015). Remapping values linearly from a 12 bit image to the specified 8 bit range would result in a dark image, since the values represent the percentage of reflectance for the given pixel (with 10000 representing a reflectance of 100%). Values are therefore spread according to their respective band mean and standard deviation, where $\bar{x} - \sigma$ represents minimum and $\bar{x} + \sigma$ the maximum. Values are than linearly remapped to 8 bit unsigned values, representing the final values displayed on the screen (see Figure 4.3).

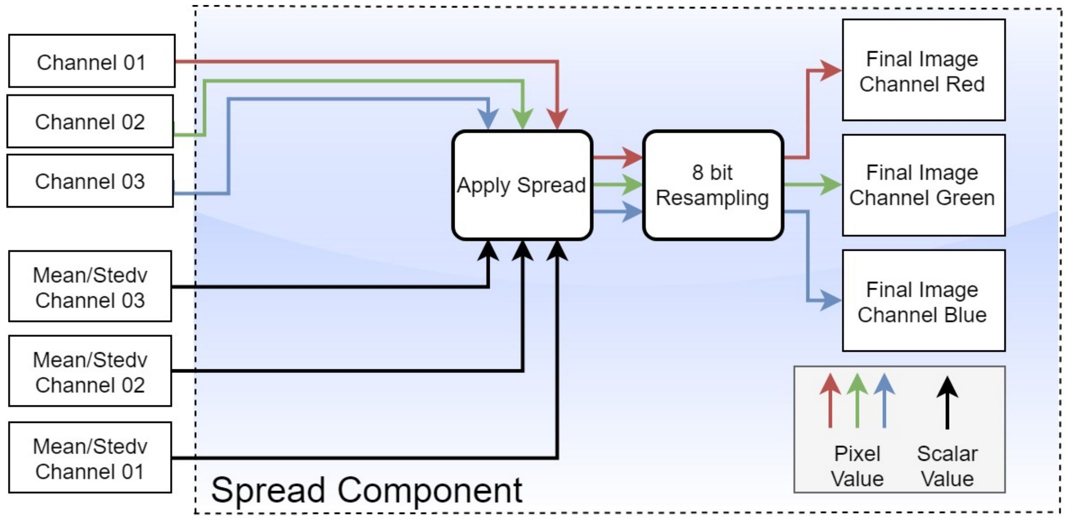


Figure 4.3.: Spreading component of the shader.

Scene Classification Masking

As explained in Section 3.4.2, pixels are classified as one of eleven possible classes. As multiple scenes of the same tile can be loaded simultaneously, masking out classes that are of no interest to the user reveal pixels of other loaded scenes. An arbitrary number and combination of classes can be masked out for each scene.

This allows for a useful tool, enabling the possibility of virtually combining scenes (especially when paired with threshold masking) across different time periods (i. e. replacing bare soil pixels with vegetation pixels from the previous year). In addition, the user is able to select a sentinel color to highlight the pixel class (as explained in Section 2.3.1). The design of this shader component is outlined in Figure 4.4.

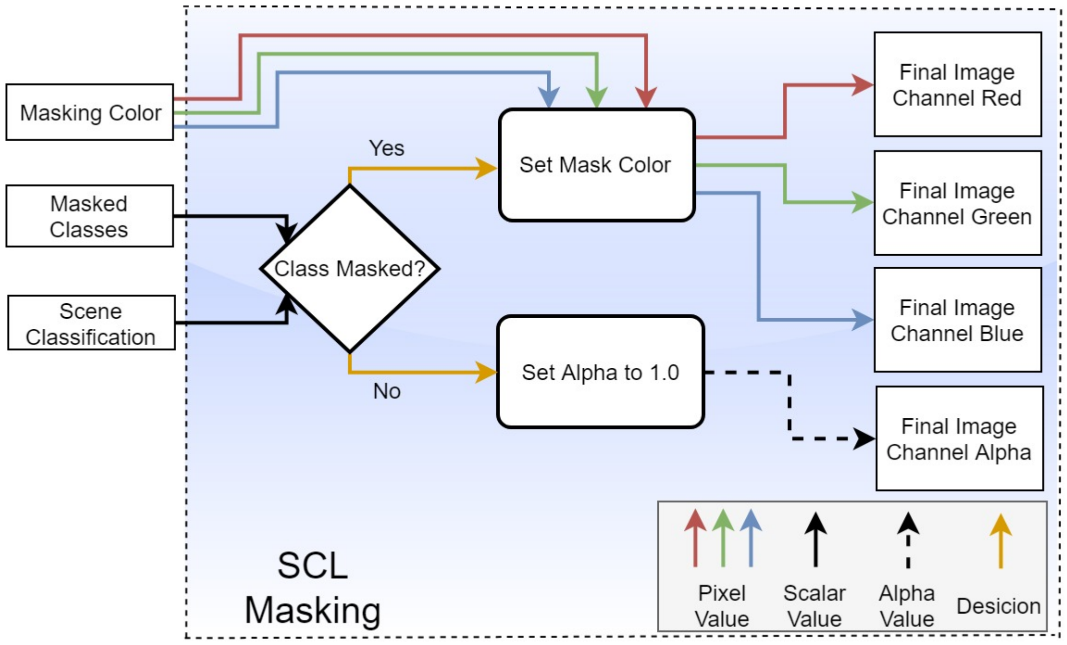


Figure 4.4.: SCL masking component of the shader.

This shader is accessed by virtually all GUI elements present in the scene, as each component is controlled by a different widget, isolating each shader component syntactically to allow a logic-driven workflow. All described parameters can be set for each loaded scene individually. Intuitive controls (such as double-ranged sliders) have to be designed to allow for efficient use.

4.4.3. Main Menu

Serving as the starting point when launching the application, the primary use of the main menu is the navigation to the Earth Explorer and Scene Explorer, respectively. In addition, the main menu provides access to "housekeeping" functions, such as application settings.

4.4.4. Summary

This chapter described the high-level design of the implemented framework. By identifying the target audience, scientists and experts occupied in fields that have a bearing on remote sensing data, we established both functional and non-functional requirements that have to be fulfilled. These requirements dictate what the system has to accomplish. For this, Unity3D was chosen as a base framework. We showed the design of the two main components: The *Earth Explorer* which is responsible for establishing access to the actual data and the *Scene Explorer* which functions as analysis tool for downloaded data. This chapter concludes with a complete schematic overview (see Figure 4.5) of the described shader, combining all individual components described in the last section. The next chapter will focus on the implementation of the *Sentinel-2 Explorer* framework.

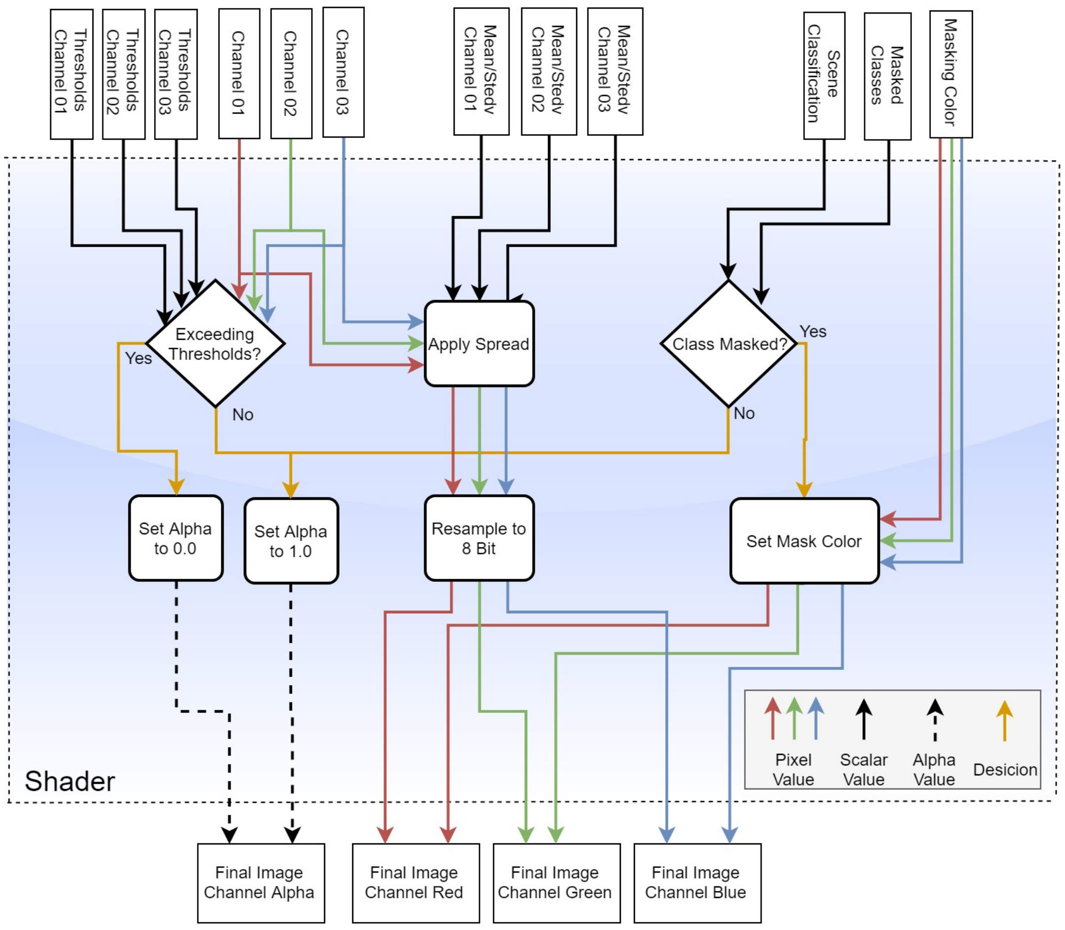


Figure 4.5.: The complete shader design for the *Scene Explorer*.

5. Implementation

Following the structure of the previous chapter, the relevant implementation details of each scene (and contained components) are described separately. This chapter does not aim to be a manual for re-implementing such a system, but serve as a description of how the *Sentinel-2 Explorer* works.

As a short recap from the Design chapter, the *Sentinel-2 Explorer* is divided into three componens:

- **EarthExplorer**: Provides access to the Copernicus servers for retrieving and downloading datasets.
- **SceneExplorer**: Allows inspection of one or multiple scenes with different analysis tools.
- **Main Menu**: Serves as navigational platform and presents a visually appealing starting point when working with the framework.

5.1. EarthExplorer

Recalling Section 4.4.1, this scene is responsible for providing access to querying and downloading Sentinel-2 products. To allow easy querying according to a given geo-location (in the form of the aforementioned tiling grid), the scenes background is a Plate Carrée projected image of earth, provided by the Blue Marble dataset. This image provides a visual aid as upon clicking an arbitrary position on this map, the system retrieves the corresponding tile name and appends it to the search query. To map the clicked position to a tile name, the GDAL/OGR library is used in conjunction with a KML[1] file of the tiling grid[2]. In addition, the user can specify various

[1]KML: http://portal.opengeospatial.org/files/?artifact$_i d = 23689$

[2]Tiling KML:https://sentinel.esa.int/web/sentinel/missions/sentinel-2/data-products

parameter as defined by the functional requirements in Section 4.3.1. Results are shown in a table-like manner and can be downloaded individually.

5.1.1. Graphical User Interface

The user interface of this scene is designed to provide an easy and fast way to interact with the system. There are three GUI components providing interactions with the system in this scene:

- **Query panel**: Provides an interface to form a search query according to the described OpenSearch format in Section 4.4.1.
- **Result Panel**: Shows returned data-sets in a table like manner and provides a low-resolution quicklook for preliminary checks without needing to download the product. In addition, high-level metadata is shown.
- **Downloading Panel**: Indicates the downloading progress of scheduled datasets.

All three components are hide-able to maximize the screen area available to the user for exploring and navigating on the Plate Carrée map.

Query panel

As shown in Figure 5.1, the textual input widget at the top of the panel is used for specifying various search parameters. It can be used to retrieve tiles and enables power users to include OpenSearch keywords to allow querying with parameters that are otherwise not accessible through the user interface. At the top left corner resides a button, which, when clicked, hides the entire panel. The button then changes, indicating an arrow pointing to the left, which, upon a consecutive click, shows the panel again. The begin and end date controls allow for restriction of retrieved datasets according to specified dates. The *today* button provides a convenience tool to set the end date to the current date.

The three sliders provide adjustable levels of cloud, snow and no-data statistics respectively. Additional input controls for specifying an orbit (or an orbit range) and checkboxes for including/excluding Sentinel-2A or Sentinel-2B data are provided. Clicking the *Search* button at the bottom starts the querying process in the `SceneController` (see Figure 5.2). The actual query (`DownloadQuery`) is constructed and passed to the download handler (`S2Downloader`). This handler starts the querying process by accessing the Copernicus server. Upon retrieving the first results (100 entries can be retrieved with one call), concrete dataset-representation objects (`AvailableProduct`) are created and posted to be shown by the user interface. This is done concurrently with retrieving and parsing further query results until no more entries are returned from the server. As retrieving entries can take up to a minute, depending on the number of datasets that have to be accessed, it is crucial to present results as soon as they are available to provide a responsive feedback.

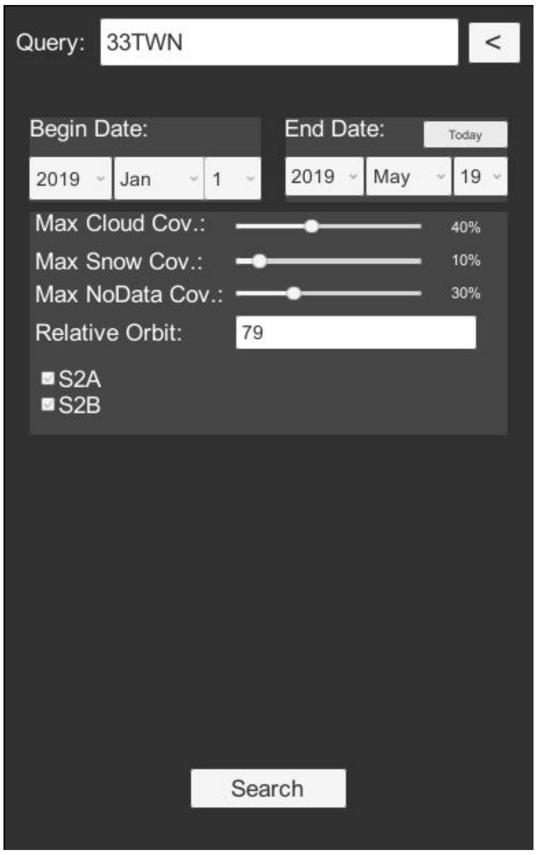

Query: 33TWN
Begin Date: 2019 Jan 1
End Date: 2019 May 19
Max Cloud Cov.: 40%
Max Snow Cov.: 10%
Max NoData Cov.: 30%
Relative Orbit: 79
S2A
S2B

Figure 5.1.: The query panel, filled with example data.

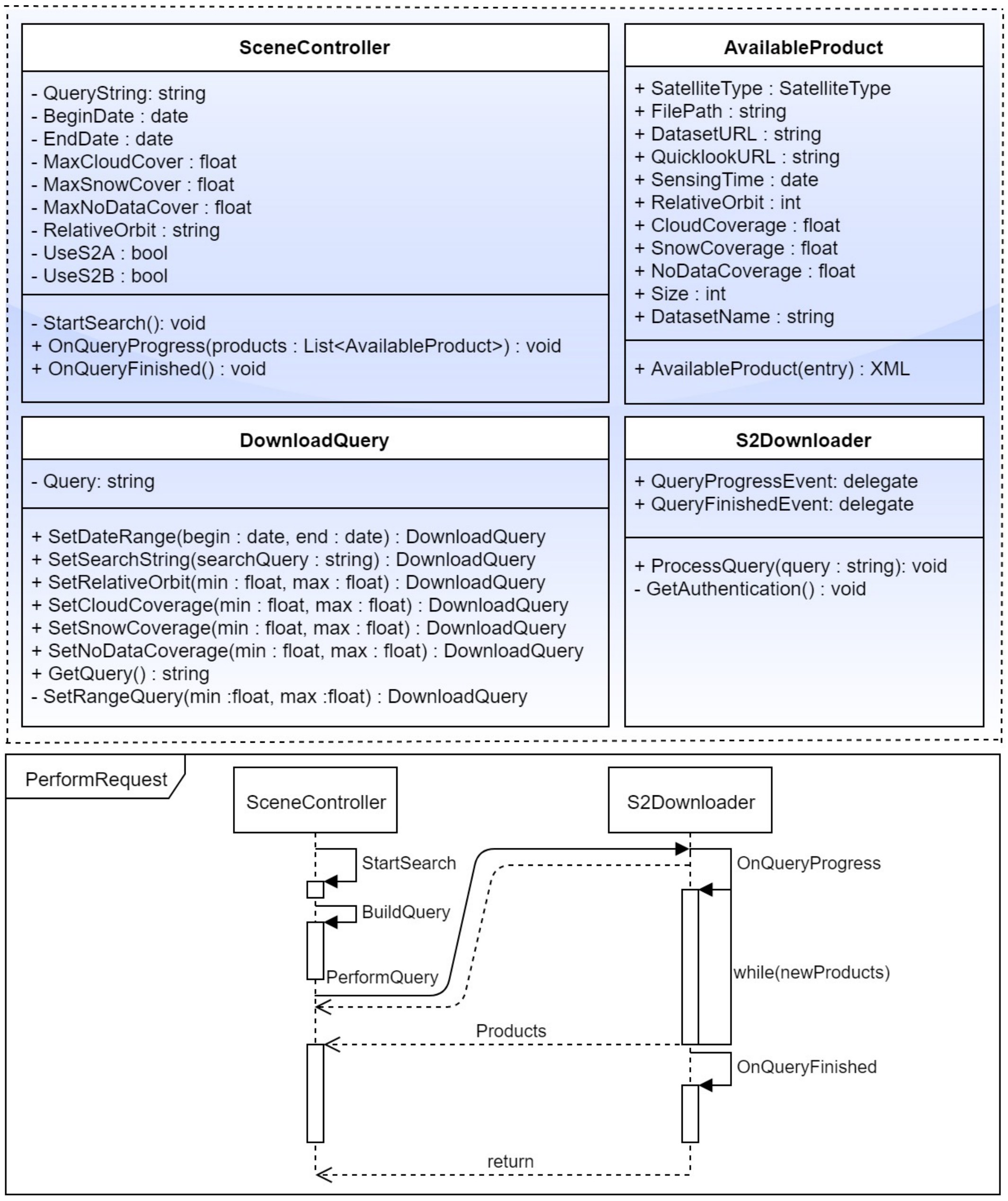
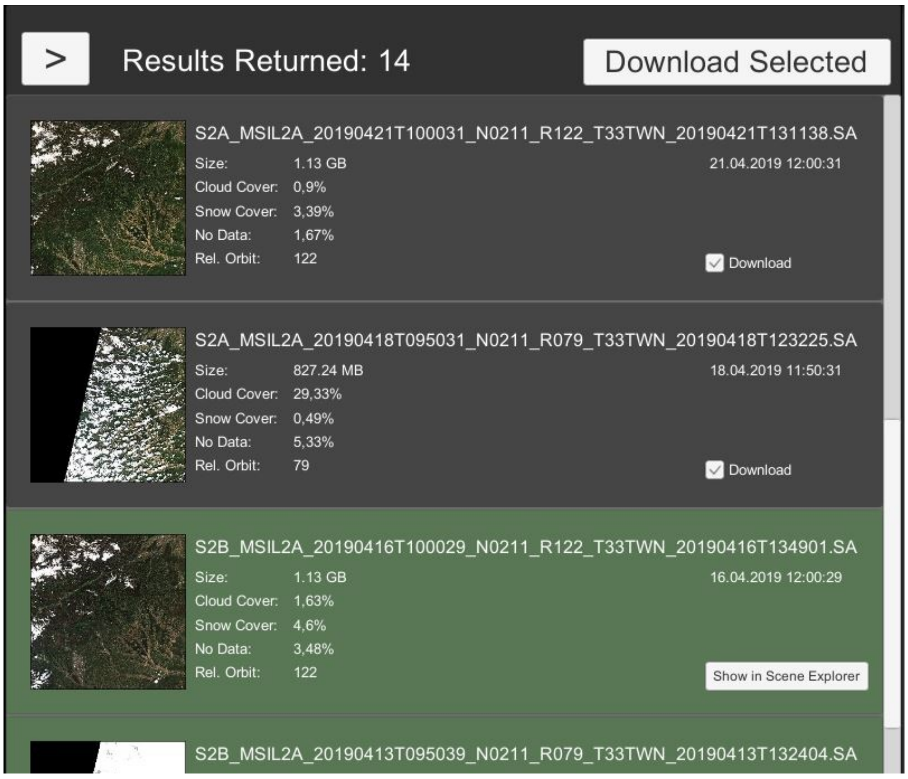

SceneController
- QueryString: string
- BeginDate : date
- EndDate : date
- MaxCloudCover : float
- MaxSnowCover : float
- MaxNoDataCover : float
- RelativeOrbit : string
- UseS2A : bool
- UseS2B : bool
- StartSearch(): void
+ OnQueryProgress(products : List<AvailableProduct>) : void
+ OnQueryFinished() : void
AvailableProduct
+ SatelliteType : SatelliteType
+ FilePath : string
+ DatasetURL : string
+ QuicklookURL : string
+ SensingTime : date
+ RelativeOrbit : int
+ CloudCoverage : float
+ SnowCoverage : float
+ NoDataCoverage : float
+ Size : int
+ DatasetName : string
+ AvailableProduct(entry) : XML
DownloadQuery
- Query: string
+ SetDateRange(begin : date, end : date) : DownloadQuery
+ SetSearchString(searchQuery : string) : DownloadQuery
+ SetRelativeOrbit(min : float, max : float) : DownloadQuery
+ SetCloudCoverage(min : float, max : float) : DownloadQuery
+ SetSnowCoverage(min : float, max : float) : DownloadQuery
+ SetNoDataCoverage(min : float, max : float) : DownloadQuery
+ GetQuery() : string
- SetRangeQuery(min :float, max :float) : DownloadQuery
S2Downloader
+ QueryProgressEvent: delegate
+ QueryFinishedEvent: delegate
+ ProcessQuery(query : string): void
- GetAuthentication() : void
PerformRequest
SceneController
S2Downloader
StartSearch
BuildQuery
PerformQuery
OnQueryProgress
while(newProducts)
Products
OnQueryFinished
return

Figure 5.2.: Top: class diagram (simplified to relevant fields/methods) of all involved components. Bottom: sequence diagram for performing a query and retrieving results.

Result Panel

Upon receiving a list filled with `AvailableProduct` instances, the products are loaded into a list view by the `SceneController`. Each `AvailableProduct` is displayed using a table entry. An example of the *Result Panel* is shown in Figure 5.3. On the left, a (true-color) quicklook is displayed, which, when hovered above with the mouse, enlarges. In addition, the name of the dataset, its size, acquisition date, relative orbit, cloud cover, snow cover and No-Data percentage is shown. A checkbox indicates, if the product is downloaded, once the download process starts. If a product is already stored on disk, it is highlighted in green and can not be downloaded again. Instead of the checkbox, a button is displayed, which, upon clicking, opens the scene in the *Scene Explorer*. All scheduled downloads are started upon clicking the "Download Selected" button at the top right corner.



Results Returned: 14

Download Selected

S2A_MSIL2A_20190421T100031_N0211_R122_T33TWN_20190421T131138.SA
Size: 1.13 GB
21.04.2019 12:00:31
Cloud Cover: 0,9%
Snow Cover: 3,39%
No Data: 1,67%
Rel. Orbit: 122
✓ Download

S2A_MSIL2A_20190418T095031_N0211_R079_T33TWN_20190418T123225.SA
Size: 827.24 MB
18.04.2019 11:50:31
Cloud Cover: 29,33%
Snow Cover: 0,49%
No Data: 5,33%
Rel. Orbit: 79
✓ Download

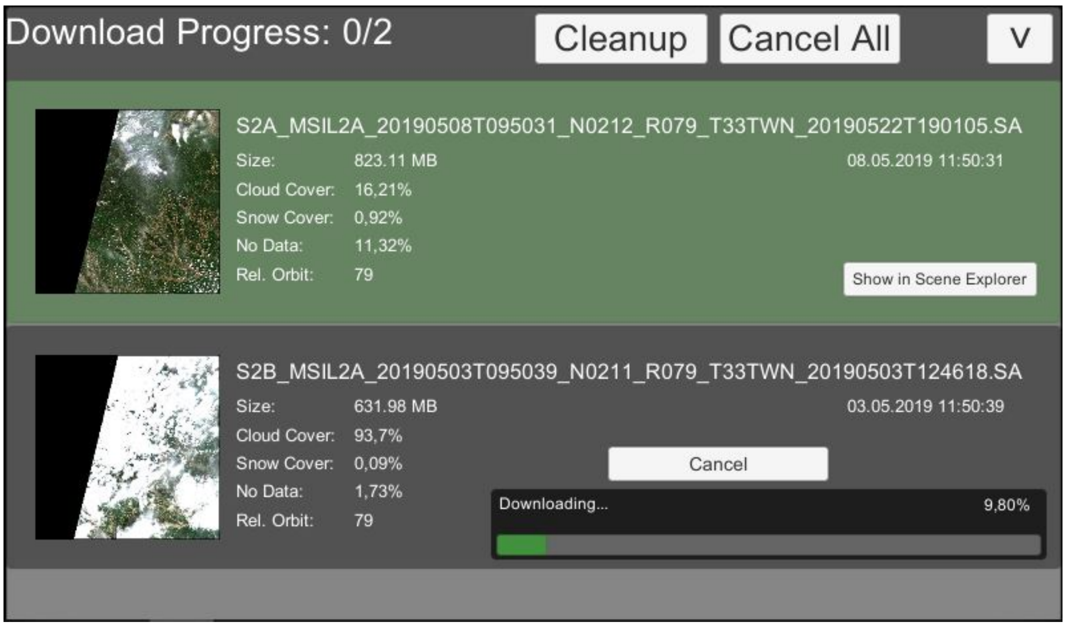
S2B_MSIL2A_20190416T100029_N0211_R122_T33TWN_20190416T134901.SA
Size: 1.13 GB
16.04.2019 12:00:29
Cloud Cover: 1,63%
Snow Cover: 4,6%
No Data: 3,48%
Rel. Orbit: 122
Show in Scene Explorer

S2B_MSIL2A_20190413T095039_N0211_R079_T33TWN_20190413T132404.SA

Figure 5.3.: GUI representation of products in the *Result Browser*.

Download Panel

Downloads are processed in succession in order of adding them to the queue. The `AvailableProduct` instances are the same as in the *Result Panel*. For the currently downloading element, a progress bar indicates the percentage of data that has already been processed. If the user so desires, a download (or all scheduled downloads) can be canceled. Canceled downloads remain in the list, but highlighted in red. Data that has already been processed is deleted from the target directory. Once the "Cleanup" button is clicked, all finished and canceled downloads are removed from the list. Upon completion, the progress bar is hidden, in its place, the "Show in Scene Explorer" button is shown, which allows the dataset to be inspected in the *Scene Explorer*. In case the user leaves the *Earth Explorer*, the *Download Panel* remains active in the background. This enables the user to download data while analyzing already completed datasets in the meantime.



S2A_MSIL2A_20190508T095031_N0212_R079_T33TWN_20190522T190105.SA

Size: 823.11 MB
08.05.2019 11:50:31
Cloud Cover: 16,21%
Snow Cover: 0,92%
No Data: 11,32%
Rel. Orbit: 79

S2B_MSIL2A_20190503T095039_N0211_R079_T33TWN_20190503T124618.SA

Size: 631.98 MB
03.05.2019 11:50:39
Cloud Cover: 93,7%
Snow Cover: 0,09%
No Data: 1,73%
Rel. Orbit: 79
9,80%

Figure 5.4.: The Download panel. The download of the upper dataset has already been completed.

5.1.2. Plate Carrée Map

To aid the user in selecting tiles of interest, the Blue Marble dataset is used as a backdrop image in the scene. To increase the users sense of orientation, national borders are drawn with the use of a shape file[3] and a Unity3D LineRenderer. The shapefile consists of multipolygons for each country (since some countries consist of multiple enclosed terretorial areas such as islands), whose points are denoted in latitude and longitude. These coordinates have to be mapped to pixel coordinates (and consequently to screen coordinates). Unfortunately, Earth is not flat, therefore making it necessary to use projections which allow a two-dimensional representations of our globe.

Projections

As explained by Ward, Grinstein, and Keim (2015) in chap. 6, p. 224 ff., projections are used to map a position on the globe (λ, φ) to a flat (2-D) surface on an image (x, y). It is defined as $(\lambda, \varphi) \rightarrow (x, y)$, where λ denotes degrees in longitude $[-180, 180]$ and φ degrees in latitude $[-90, 90]$. There are different projections with different properties (excerpt):

- **Conformal Projections** retain local angels correctly, which means they preserve shapes. This, however, does not hold true for areas.
- **Equal Area Projections** assure, that a specific area on the globe covers the same area on the map as an equal-sized area on another part of the globe. This comes at the cost of distorting forms and angles.
- **Equidistant Projections** preserve the distance between lines.

Mappings may also be classified by the type of surface onto they are projected (excerpt):

- **Cylindrical Projections** project the surface of the sphere on a cylinder put around the sphere. Most projections of this type preserve local angles.

[3]Country borders: http://www.naturalearthdata.com/downloads/50m-cultural-vectors/

- **Plane Projections** map the surface of the sphere onto a plane that is tangent to the sphere, with the tangent point representing the center point of the projection.

An example for a cylindrical projection is NASAs Blue Marble (Stöckli et al., 2005). An image of the dataset can bee seen in Figure 5.5.

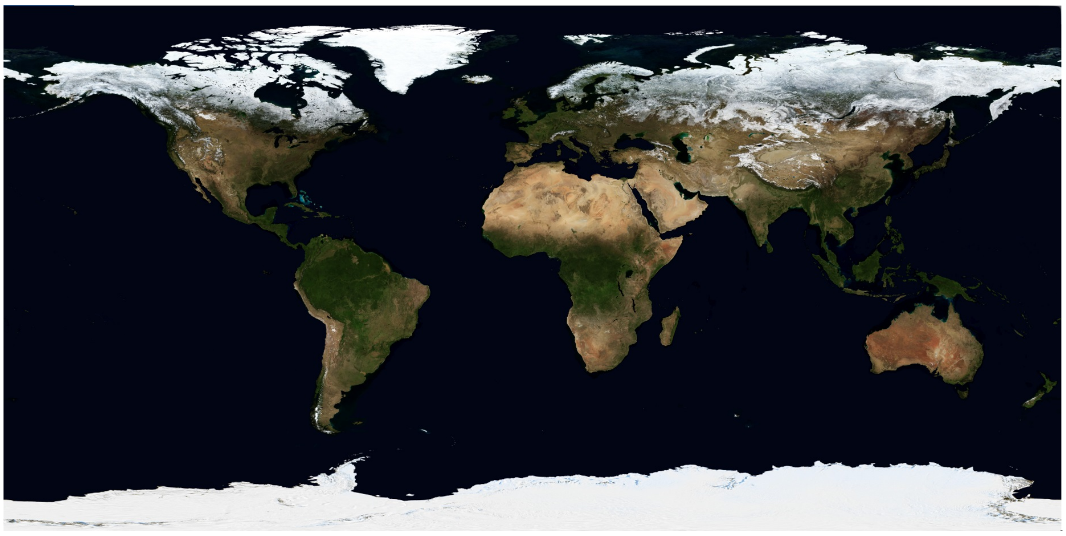


Figure 5.5.: NASAs Blue Marble MODIS mosaic in Plate Carrée projection (Stöckli et al., 2005). Distortions in the polar areas are clearly visible in this type of projection.

Blue Marble is provided in spatial resolutions of 500m, 2km and 8km and uses an equirectangular projection (Plate Carrée). As described by Ward, Grinstein, and Keim (2015) in chap. 6 p. 226, this type of cylindrical projection is one of the oldest and simplest projections. It maps meridians to equally spaced vertical straight lines and circles of latitude to evenly spaced horizontal straight lines. The mapping from a sphere onto a rectangular surface is 1:1.

$$x = \lambda \qquad\qquad y = \varphi \tag{5.1}$$

Although equirectangular projections are neither conformal nor equal-area, they are still used in thematic mapping.

This equirectangular mapping property allows trivial conversion of the polygon-coordinates in the shape file onto the image in the scene, by simply scaling the image to an extent of 360 units in width and 180 units in height. The center of the image has to be at the origin of the scenes coordinate system. The individual points of each polygon are then drawn with a Line Renderer. The aforementioned shape file that contains the tiling grid of the Sentinel-2 satellites denotes its polygons in the same coordinate system (latitude and longitude). When the user performs a left-click anywhere on the map, the shapefile is queried for the coordinates using the GDAL library. GDAL is an acronym for *Geospatial Data Abstraction Library*. It is a translator library for raster and vector geospatial data. It provides a single raster (and vector) abstract data model to the calling application for all supported formats. The OGR (OpenGis Simple Features Reference Implementation) library is part of the GDAL package that provides read/write access to vector file formats (such as shapefiles and KMLs). The GDAL library package is released under an X/MIT style Open Source license by the Open Source Geospatial Foundation (GDAL/OGR contributors, 2019). The shapefile that references the tiling grid contains over 55,000 entries, each with four points denoting the location and orientation on the globe. A raycast is used to receive an exact position on the equirectangular Earth map from the screen coordinates of the mouse click. This location is then set as spatial filter for the shape file. GDAL can then be used to return only features that enclose this point. As shown in Figure 5.6, the resulting tile is highlighted, and the name of the tile is appended to the query. The example below shows the shapefile entry for the tile `33TWN` (excess decimal places are truncated):

```
type: Feature,
id: 31323,
properties:
    name : 33TWN,
    geometry:
        type: Polygon,
        coordinates: [ (14.99, 47.85, 0.0), (16.46, 47.84, 0.0),
                       (16.44, 46.85, 0.0), (14.99, 46.86, 0.0),
                       (14.99, 47.85, 0.0) ]
```

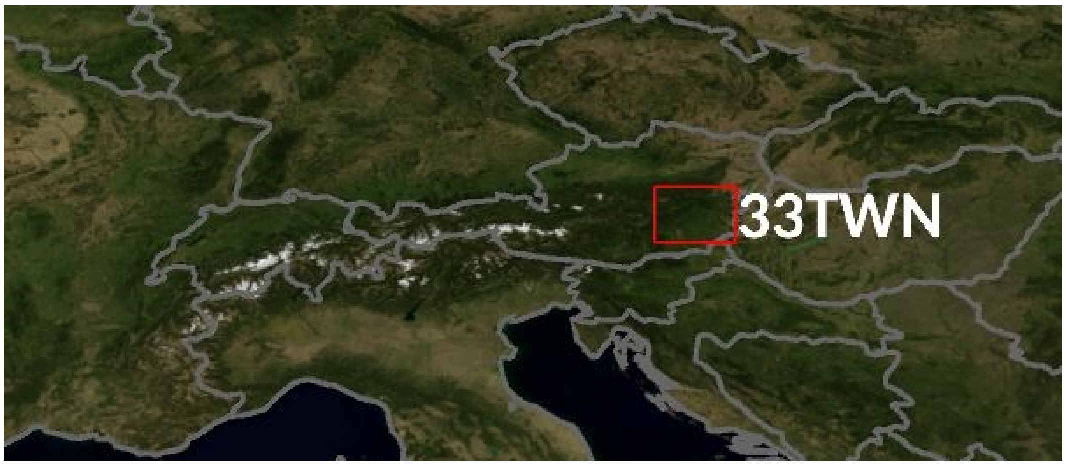
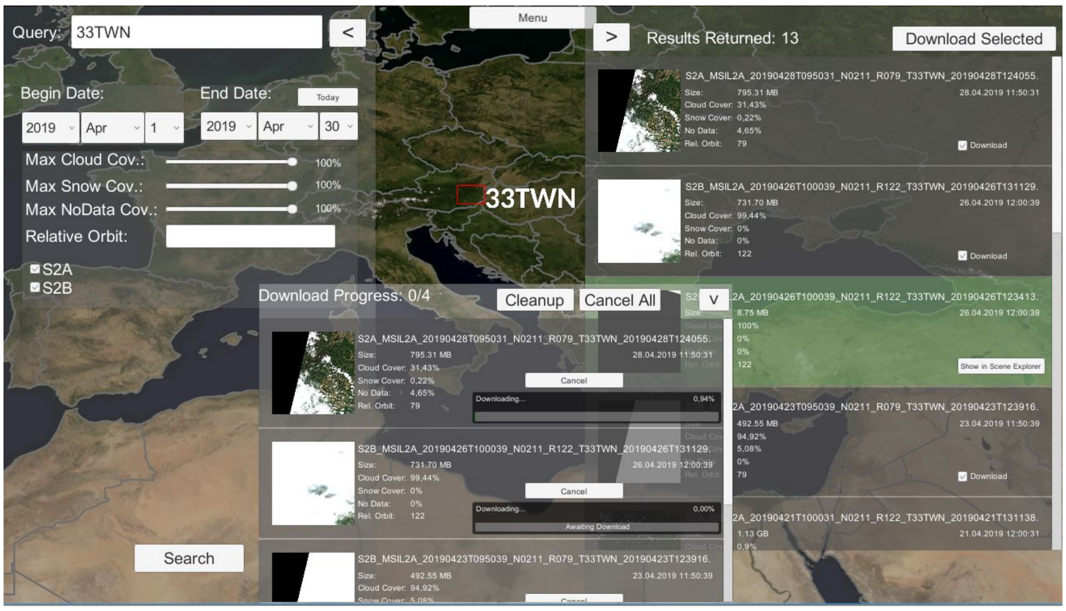


Figure 5.6.: Upon a mouse click, the system highlights the tile that covers the point of interaction.

Concluding this section, Figure 5.7 shows the completed *Earth Explorer*. The next section will describe the implementation of the *Scene Explorer*.



Query: 33TWN

Begin Date: 2019 Apr 1

End Date: 2019 Apr 30

Max Cloud Cov.: 100%

Max Snow Cov.: 100%

Max NoData Cov.: 100%

Relative Orbit:

☑ S2A

☑ S2B

Results Returned: 13

S2A_MSIL2A_20190428T095031_N0211_R079_T33TWN_20190428T124055.
Size: 795.31 MB — 28.04.2019 11:50:31
Cloud Cover: 31,43%
Snow Cover: 0,22%
No Data: 4,65%
Rel. Orbit: 79
☑ Download

S2B_MSIL2A_20190426T100039_N0211_R122_T33TWN_20190426T131129.
Size: 731.70 MB — 26.04.2019 12:00:39
Cloud Cover: 99,44%
Snow Cover: 0%
No Data: 0%
Rel. Orbit: 122
☑ Download

S2 2A_20190426T100039_N0211_R122_T33TWN_20190426T123413.
Size: 8.75 MB — 26.04.2019 12:00:39
100%
0%
0%
122

2A_20190423T095039_N0211_R079_T33TWN_20190423T123916.
492.55 MB — 23.04.2019 11:50:39
94,92%
5,08%
0%
79
☑ Download

2A_20190421T100031_N0211_R122_T33TWN_20190421T131138.
1.13 GB — 21.04.2019 12:00:31
0,9%

Download Progress: 0/4

S2A_MSIL2A_20190428T095031_N0211_R079_T33TWN_20190428T124055.
Size: 795.31 MB — 28.04.2019 11:50:31
Cloud Cover: 31,43%
Snow Cover: 0,22%
No Data: 4,65%
Rel. Orbit: 79
Downloading... 0,94%

S2B_MSIL2A_20190426T100039_N0211_R122_T33TWN_20190426T131129.
Size: 731.70 MB — 26.04.2019 12:00:39
Cloud Cover: 99,44%
Snow Cover: 0%
No Data: 0%
Rel. Orbit: 122
Downloading... 0,00%
Awaiting Download

S2B_MSIL2A_20190423T095039_N0211_R079_T33TWN_20190423T123916.
Size: 492.55 MB — 23.04.2019 11:50:39
Cloud Cover: 94,92%
Snow Cover: 5,08%

Figure 5.7.: The completed *Earth Explorer* with all panels shown.

5.2. Scene Explorer

Recalling Section 4.4.2 of the Design chapter, the *Scene Explorer* has the purpose of allowing inspection and analysis for previously downloaded scenes. A variety of tools are at the users disposal to perform the following tasks:

- **Display Settings**: Used to select the desired band combination. The user can select from a variety of pre-set combinations or specify a custom selection. In addition, this widget enables masking of pixel classes according to the scene classification mask.
- **Value Pinching**: This tool enables the user to mask out pixels falling below or exceeding configurable thresholds for each displayed band.
- **Temporal Analysis**: A graph showing the changes in pixel classes over time, normalized to a percentage value.
- **Dataset Properties**: Shows the user meta-data information of the currently selected dataset. This includes percentages for all pixel classes.
- **Value Viewer**: Shows the pixel values (and pixel class) of individual pixels. Can also be used to retrieve geographic coordinates of a specific location in the dataset.

Before examining the specific details and implementation of each tool, we will discuss how scenes can be loaded efficiently to increase performance and decrease RAM consumption.

5.2.1. Scene Loading

The *Scene Explorer* is designed to make use of all bands provided in a Sentinel-2 dataset. Table 3.2 in the Dataset chapter shows, that many bands are available in multiple resolutions. The Design chapter already established that bands should only be loaded in the sufficiently highest resolution. Upon zooming, a check is performed if the loaded resolution is still high enough to provide the appropriate level of detail. If not, the next higher (or lower) resolution is loaded. Bands that are not available in the desired resolution are up or downsampled on the fly.

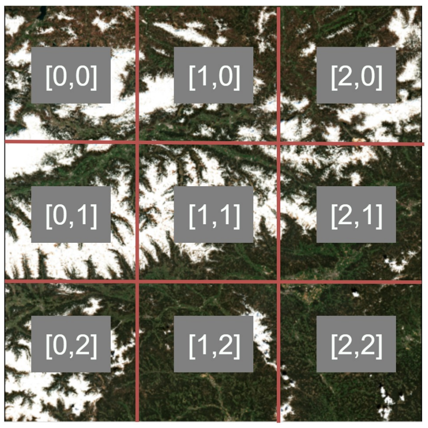


Figure 5.8.: The scene tiled into a 3x3 grid.

Doing this for the entire dataset at once would be quite taxing on the computers hardware. To overcome this, the image is subdivided into sub-sections (see Figure 5.8). Only sections visible to the camera are considered in the resampling/reloading process. When a sub-section of the scene is not visible (either by not being in the cameras frustum or by being overlayed by another scene), its state remains unchanged until it becomes visible again. The `SceneController` holds a list of all loaded datasets. When a new `S2Scene` is loaded at the users request, nine instances of the class `RenderPlaneController` are created, each responsible for one specific tile of the image. The loading of the scene includes unpacking the image data to a designated working directory and loading a standard band combination. This process is done asynchronously (as, depending on the hard drive speed, unpacking may take a few seconds), allowing the user to continue working on already loaded scenes. Upon zooming/panning, each `RenderPlaneController` checks, if the event made reloading image data necessary. If so, new data may be requested from the parent `S2Scene`. A scene may need to reload its content entirely (i.e. a resolution change is necessary) or redraw its content (i.e. when masking or pinching parameters change). A reload is generally slower, as it involves reading data from the disk. In contrast, redrawing only re-applies the shader on the already loaded image-data. Although the user may select an arbitrary band combination for display, the scene classification is always loaded into the alpha channel. The alpha channel is not visible in the final image, but it is used by the shader and other components to provide the pixel type for each position in the image. As multiple scenes (of potentially different tiles) can be loaded at the same time, taking into acount the geoposition of each dataset is necessary. Tiles in the same projection (UTM zone) are loaded at their correct position, with the first loaded scene determining the coordinate system. If two scenes of the same tile (and therefore with the same geographic position and extent) are loaded, the most recently loaded scene is overlayed onto the previous scene. Loading scenes with differing UTM projections is not possible. Masking pixel classes or applying pinching values to a scene may therefore reveal pixels of the underlying dataset.

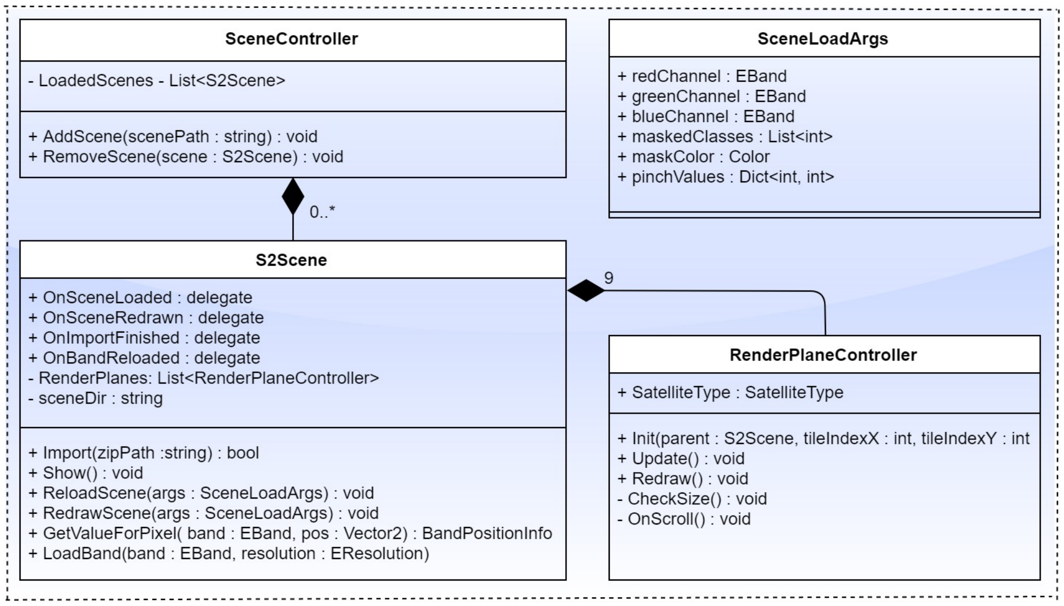

SceneController
- LoadedScenes - List<S2Scene>
+ AddScene(scenePath : string) : void
+ RemoveScene(scene : S2Scene) : void
0..*
SceneLoadArgs
+ redChannel : EBand
+ greenChannel : EBand
+ blueChannel : EBand
+ maskedClasses : List<int>
+ maskColor : Color
+ pinchValues : Dict<int, int>
S2Scene
+ OnSceneLoaded : delegate
+ OnSceneRedrawn : delegate
+ OnImportFinished : delegate
+ OnBandReloaded : delegate
- RenderPlanes: List<RenderPlaneController>
- sceneDir : string
+ Import(zipPath :string) : bool
+ Show() : void
+ ReloadScene(args : SceneLoadArgs) : void
+ RedrawScene(args : SceneLoadArgs) : void
+ GetValueForPixel(band : EBand, pos : Vector2) : BandPositionInfo
+ LoadBand(band : EBand, resolution : EResolution)
9
RenderPlaneController
+ SatelliteType : SatelliteType
+ Init(parent : S2Scene, tileIndexX : int, tileIndexY : int
+ Update() : void
+ Redraw() : void
- CheckSize() : void
- OnScroll() : void

Figure 5.9.: Class diagram (simplified to relevant fields/methods) of all involved components.

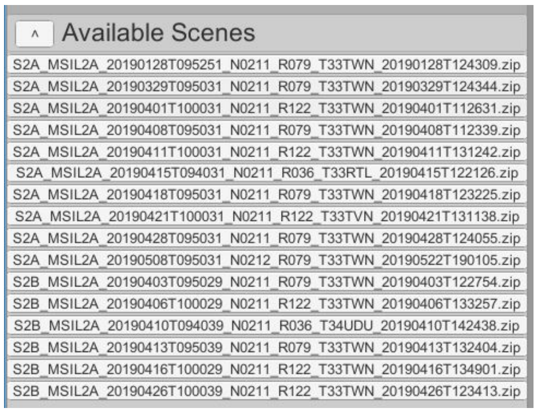
5.2.2. Tools and Widgets

To enable efficient interaction with the loaded scenes, a carefully designed user interface is necessary. The interface is divided into three separate panels on the left, right and bottom, respectively. Each panel is hide-able, following the same principle as in the *Earth Explorer*. The panels hold individual widgets, which can be collapsed and expanded individually. This reduces the risk of distracting the user with widgets he is not interested in, while simultaneously increasing the screen space that is available for other widgets. Hiding entire panels also allows more space to be taken up by the loaded datasets.

On the following pages, all implemented widgets are explained.

Available Scenes Widget

This widget (shown in Figure 5.10) is responsible for interfacing all available downloaded scenes that can be loaded into the *Scene Explorer*. The download directory is scanned for all zip files that match a certain file name pattern. For each found scene, a button is added to a list.



^ Available Scenes

S2A_MSIL2A_20190128T095251_N0211_R079_T33TWN_20190128T124309.zip
S2A_MSIL2A_20190329T095031_N0211_R079_T33TWN_20190329T124344.zip
S2A_MSIL2A_20190401T100031_N0211_R122_T33TWN_20190401T112631.zip
S2A_MSIL2A_20190408T095031_N0211_R079_T33TWN_20190408T112339.zip
S2A_MSIL2A_20190411T100031_N0211_R122_T33TWN_20190411T131242.zip
S2A_MSIL2A_20190415T094031_N0211_R036_T33RTL_20190415T122126.zip
S2A_MSIL2A_20190418T095031_N0211_R079_T33TWN_20190418T123225.zip
S2A_MSIL2A_20190421T100031_N0211_R122_T33TVN_20190421T131138.zip
S2A_MSIL2A_20190428T095031_N0211_R079_T33TWN_20190428T124055.zip
S2A_MSIL2A_20190508T095031_N0212_R079_T33TWN_20190522T190105.zip
S2B_MSIL2A_20190403T095029_N0211_R079_T33TWN_20190403T122754.zip
S2B_MSIL2A_20190406T100029_N0211_R122_T33TWN_20190406T133257.zip
S2B_MSIL2A_20190410T094039_N0211_R036_T34UDU_20190410T142438.zip
S2B_MSIL2A_20190413T095039_N0211_R079_T33TWN_20190413T132404.zip
S2B_MSIL2A_20190416T100029_N0211_R122_T33TWN_20190416T134901.zip
S2B_MSIL2A_20190426T100039_N0211_R122_T33TWN_20190426T123413.zip

Figure 5.10.: The *Available Scenes* widget.

Clicking the button starts the loading process outlined on the previous pages for the associated scene. After the `SceneController` has loaded the new scene, all other widgets are notified with a callback to update their respective user interface. Figure 5.11 outlines the interaction between the `SceneController` and the `AvailableScenesController`.

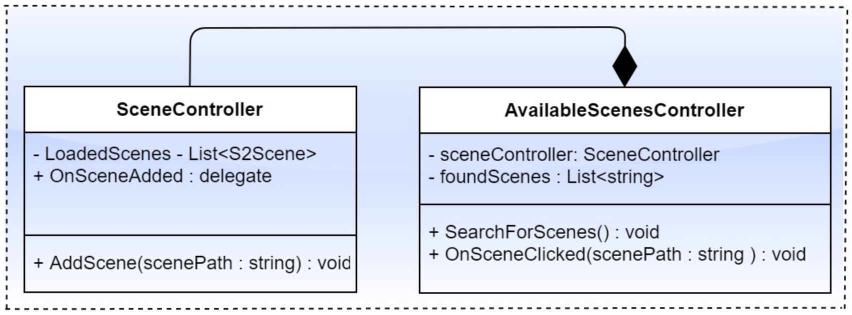
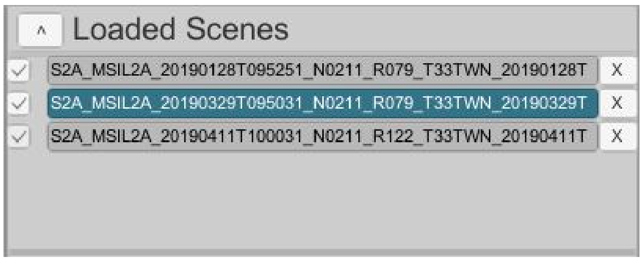

SceneController
- LoadedScenes - List<S2Scene>
+ OnSceneAdded : delegate
+ AddScene(scenePath : string) : void
AvailableScenesController
- sceneController: SceneController
- foundScenes : List<string>
+ SearchForScenes() : void
+ OnSceneClicked(scenePath : string) : void

Figure 5.11.: Class diagram showing the relation between the scene and widget Controller.

Loaded Scenes Widget

All loaded scenes are removed from the *Available Scenes* widget and added to this widget. In addition to indicating which scenes are loaded (see Figure 5.12), this widget also keeps track of which scene is currently selected. If the selected scene changes, all other widgets are notified to update their GUI accordingly. Each loaded scene can be hidden by clicking on a checkbox. Closing scenes is done by clicking the "X" button on the right of each row.



✓ S2A_MSIL2A_20190128T095251_N0211_R079_T33TWN_20190128T X
✓ S2A_MSIL2A_20190329T095031_N0211_R079_T33TWN_20190329T X
✓ S2A_MSIL2A_20190411T100031_N0211_R122_T33TWN_20190411T X

Figure 5.12.: The *Loaded Scenes* widget. The second scene is currently selected.

As with the *Available Scenes* widget, this widget performs neither hiding nor unloading scenes directly. Instead, the `SceneController` is notified of changes and acts accordingly. Figure 5.13 shows the relationship between the `LoadedScenesController` and the `SceneController`.

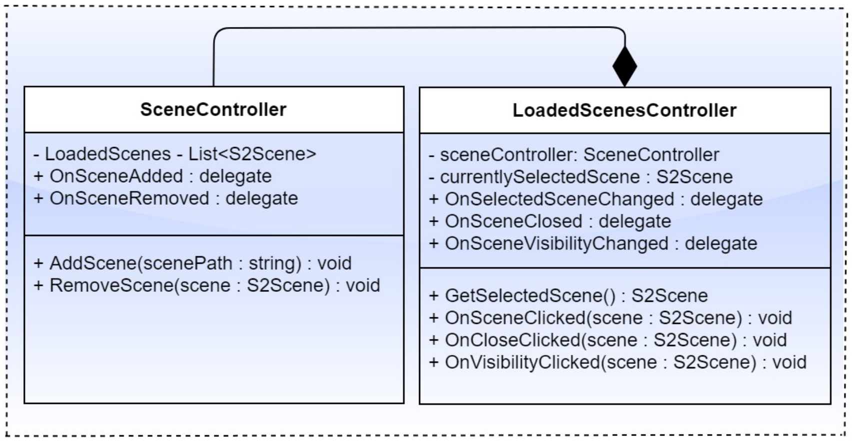

SceneController
- LoadedScenes - List<S2Scene>
+ OnSceneAdded : delegate
+ OnSceneRemoved : delegate
+ AddScene(scenePath : string) : void
+ RemoveScene(scene : S2Scene) : void
LoadedScenesController
- sceneController: SceneController
- currentlySelectedScene : S2Scene
+ OnSelectedSceneChanged : delegate
+ OnSceneClosed : delegate
+ OnSceneVisibilityChanged : delegate
+ GetSelectedScene() : S2Scene
+ OnSceneClicked(scene : S2Scene) : void
+ OnCloseClicked(scene : S2Scene) : void
+ OnVisibilityClicked(scene : S2Scene) : void

Figure 5.13.: Interaction between the main and widget controller.

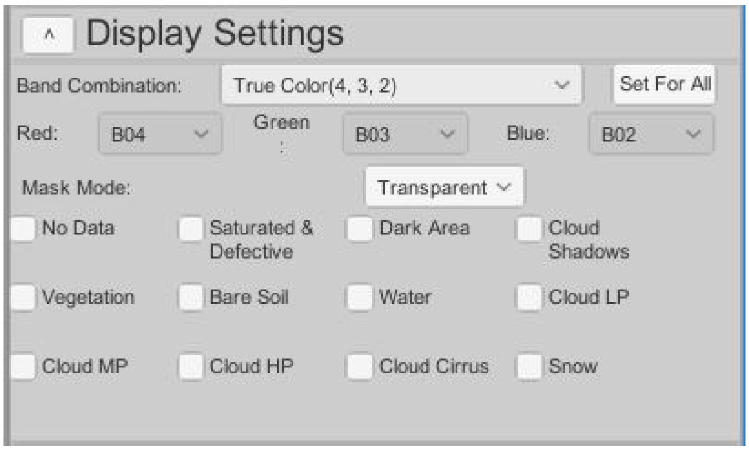
Display Settings Widget

Being one of the most important widgets, the *Display Settings* widget (see Figure 5.14) is responsible for providing an interface to the user that allows setting up the actual visualization of the dataset. Those settings are:

- Selecting a band combination for the rendered RGB-image by choosing one of the following presets:

– True Color	(B04, B03, B02)
– False Color IR	(B08, B04, B03)
– False Color Urban	(B12, B11, B04)
– Agriculture	(B11, B08, B03)
– Atmospheric Penetration	(B12, B11, B8A)
– Healthy Vegetation	(B08, B11, B02)
– Land/Water	(B08, B11, B04)
– Custom	

- Masking out arbitrary pixel classes according to scene classification mask
- Setting a masking mode for masked pixel classes (Transparent or a solid color)


Display Settings

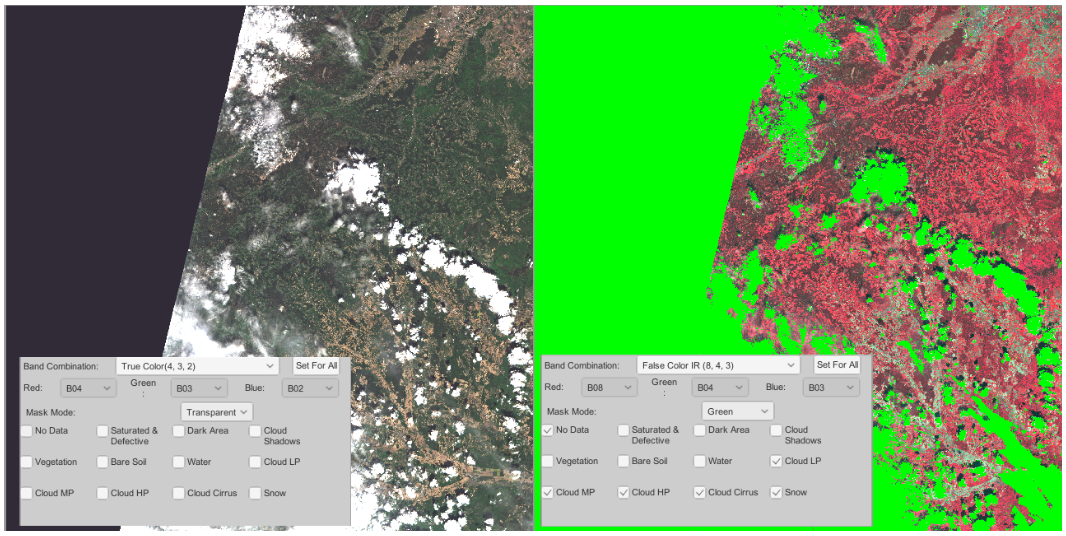
Band Combination: True Color(4, 3, 2)

Red: B04 Green: B03 Blue: B02

Mask Mode: Transparent

Figure 5.14.: The *Display Settings* widget in its default state.

When a new scene is loaded (or the selected scene changes), the *Display Settings* widgets controller retrieves the `SceneLoadArgs` associated with the currently active scene. This class holds all information necessary for the visualization shader, such as the current band combination, masked classes and the masking mode. The user interface is updated to match the actual visualization parameters that may have been set previously or have been loaded by default. Upon changing one of the parameters (see Figure 5.15), the `DisplaySettingsController` retrieves a reference to the currently active scene from the `LoadedScenesController` and constructs a new `SceneLoadArgs` instance. By passing this instance to the `S2Scene`, the scene is either reloaded or redrawn. Recalling the loading process, reloading the scene is necessary only if the band combination has been changed. Due to the architecture of the `S2Scene`, the actual redrawing/reloading is only done for tiles that are actually visible at the time of change. For tiles that are not currently visible, the changes are stored and only applied once the tile becomes visible to the camera. For convenience, this widget also features a button to apply the currently set display parameters to all loaded scenes. The class diagram in Figure 5.16 shows how this widget interacts with other components in the scene.


Band Combination: True Color(4, 3, 2) | Set For All
Red: B04 | Green: B03 | Blue: B02
Mask Mode: Transparent
☐ No Data ☐ Saturated & Defective ☐ Dark Area ☐ Cloud Shadows
☐ Vegetation ☐ Bare Soil ☐ Water ☐ Cloud LP
☐ Cloud MP ☐ Cloud HP ☐ Cloud Cirrus ☐ Snow

Band Combination: False Color IR (8, 4, 3) | Set For All
Red: B08 | Green: B04 | Blue: B03
Mask Mode: Green
✓ No Data ☐ Saturated & Defective ☐ Dark Area ☐ Cloud Shadows
☐ Vegetation ☐ Bare Soil ☐ Water ✓ Cloud LP
✓ Cloud MP ✓ Cloud HP ✓ Cloud Cirrus ✓ Snow

Figure 5.15.: Left: Original scene without masking in True Color band combination. Right: The same scene in False Color IR and masked cloud, snow and no-data pixels. Masking color has been set to solid green.

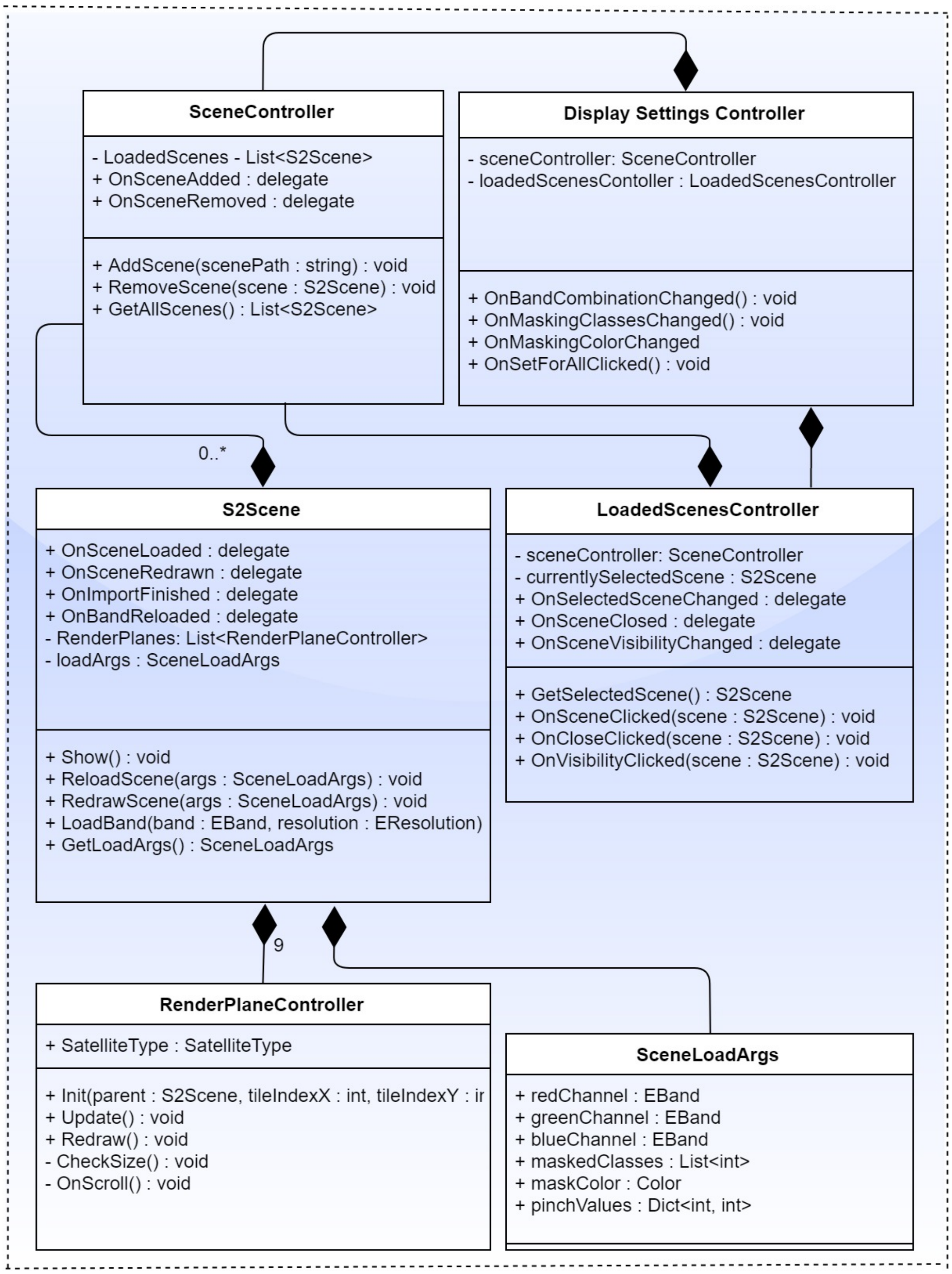

SceneController
- LoadedScenes - List<S2Scene>
+ OnSceneAdded : delegate
+ OnSceneRemoved : delegate
+ AddScene(scenePath : string) : void
+ RemoveScene(scene : S2Scene) : void
+ GetAllScenes() : List<S2Scene>
Display Settings Controller
- sceneController: SceneController
- loadedScenesContoller : LoadedScenesController
+ OnBandCombinationChanged() : void
+ OnMaskingClassesChanged() : void
+ OnMaskingColorChanged
+ OnSetForAllClicked() : void
0..*
S2Scene
+ OnSceneLoaded : delegate
+ OnSceneRedrawn : delegate
+ OnImportFinished : delegate
+ OnBandReloaded : delegate
- RenderPlanes: List<RenderPlaneController>
- loadArgs : SceneLoadArgs
+ Show() : void
+ ReloadScene(args : SceneLoadArgs) : void
+ RedrawScene(args : SceneLoadArgs) : void
+ LoadBand(band : EBand, resolution : EResolution)
+ GetLoadArgs() : SceneLoadArgs
LoadedScenesController
- sceneController: SceneController
- currentlySelectedScene : S2Scene
+ OnSelectedSceneChanged : delegate
+ OnSceneClosed : delegate
+ OnSceneVisibilityChanged : delegate
+ GetSelectedScene() : S2Scene
+ OnSceneClicked(scene : S2Scene) : void
+ OnCloseClicked(scene : S2Scene) : void
+ OnVisibilityClicked(scene : S2Scene) : void
9
RenderPlaneController
+ SatelliteType : SatelliteType
+ Init(parent : S2Scene, tileIndexX : int, tileIndexY : ir
+ Update() : void
+ Redraw() : void
- CheckSize() : void
- OnScroll() : void
SceneLoadArgs
+ redChannel : EBand
+ greenChannel : EBand
+ blueChannel : EBand
+ maskedClasses : List<int>
+ maskColor : Color
+ pinchValues : Dict<int, int>

Figure 5.16.: Class diagram for all involved classes surrounding the *Display Settings* widget.

Value Pincher Widget

In addition to changing the display settings, the user has the possibility to affect the visualization by editing the pinching values. Recalling the Design chapter, the shader maps the 12-bit value range to an 8-bit range by remapping the reflectance values according to their mean and standard deviation. The *Value Pincher* widget (see Figure 5.17) allows the user to render pixels outside the specified reflectance values invisible. As a visual aid, the mean and standard deviation are displayed for each band individually.

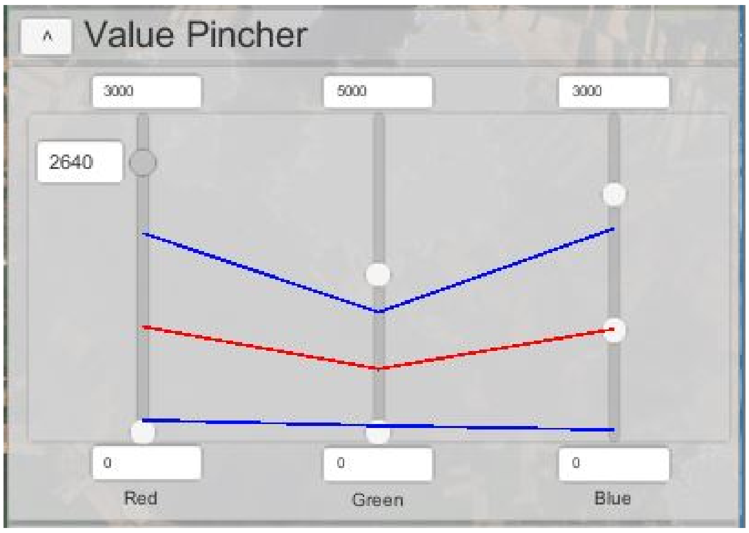


Figure 5.17.: The *Value Pincher* widget.

The widget provides three double-ranged sliders, one for each currently loaded band. the maximum and minimum of each individual slider are set by default to the highest and lowest occurring reflectance value in the band, respectively. The red line denotes the mean value for the band, the upper and lower blue lines denote the standard deviation. Moving the sliders modifies the shader threshold values for masking out pixels. The maximum and minimum can be changed at the users discretion to allow for a finer, more granular control over the ranges. When clicking upon a slider handle, an input widgets appears. It reveals the current reflectance threshold for the band while acting as an input widget at the same time, allowing the user to enter an exact reflectance threshold value. This tool can be helpful in finding areas with specific reflectance parameters that could have been miss-classified by the scene classification algorithm, providing a method for manually masking out pixels. Figure 5.18 shows an example of the effect on a scene with applied pinching values (from Figure 5.17).

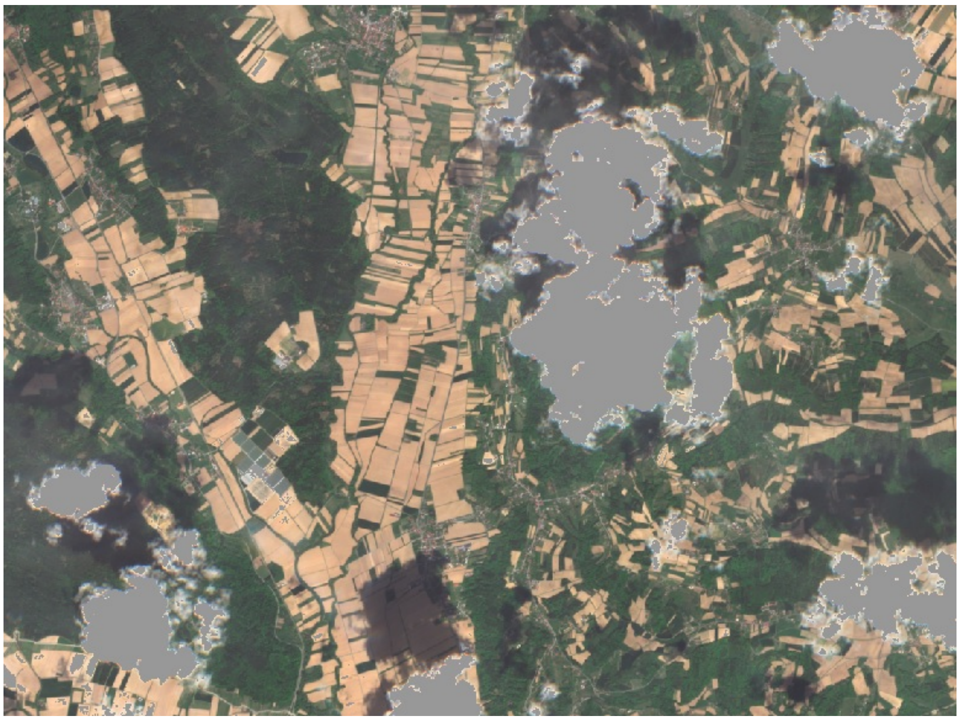


Figure 5.18.: Cloud masking with the help of the *Value Pincher* widget. The cloud shadows are still visible.

Upon changing the pinching values, the modified `DRSliderController` notifies the `ValuePincherController` of its changed value, which in turn gets a reference to the currently active `S2Scene` from the `LoadedScenesController`. The `SceneLoadArgs` of the scene are then modified to match the currently set pinching values. The scene is then redrawn by issuing a call to all `RenderPlaneController` instances associated with the scene. When another scene becomes active, the `ValuePincherController` is notified via callbacks and updates its pinching values by retrieving `SceneLoadArgs` currently set in the scene.

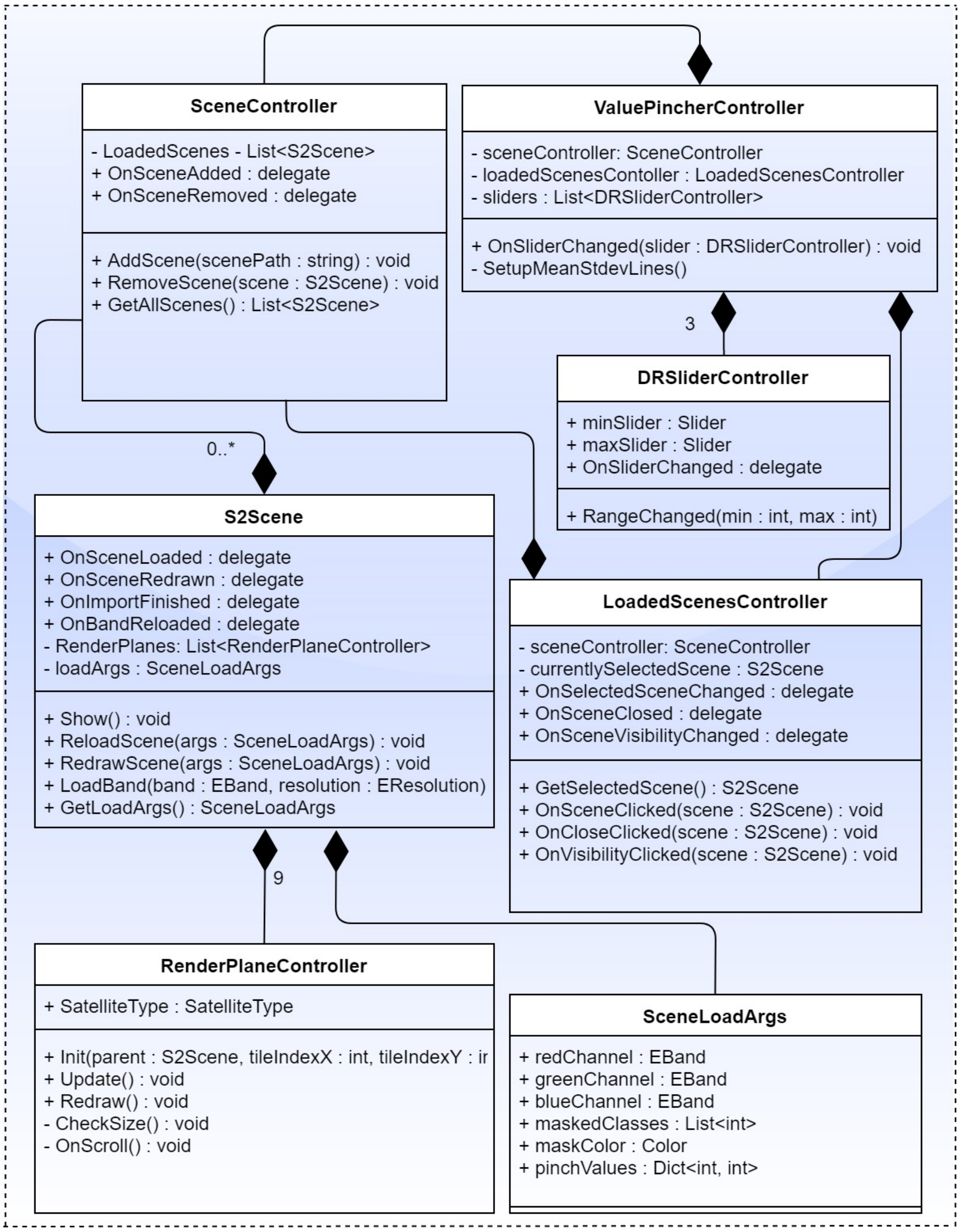
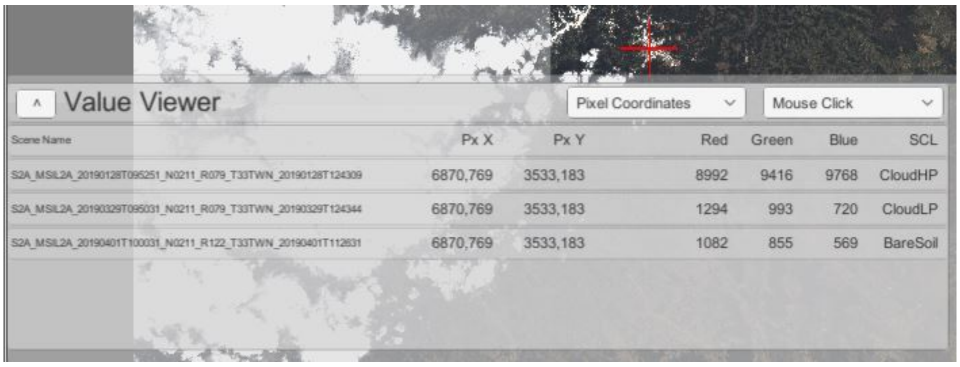

SceneController
- LoadedScenes - List<S2Scene>
+ OnSceneAdded : delegate
+ OnSceneRemoved : delegate
+ AddScene(scenePath : string) : void
+ RemoveScene(scene : S2Scene) : void
+ GetAllScenes() : List<S2Scene>
ValuePincherController
- sceneController: SceneController
- loadedScenesContoller : LoadedScenesController
- sliders : List<DRSliderController>
+ OnSliderChanged(slider : DRSliderController) : void
- SetupMeanStdevLines()
3
DRSliderController
+ minSlider : Slider
+ maxSlider : Slider
+ OnSliderChanged : delegate
+ RangeChanged(min : int, max : int)
0..*
S2Scene
+ OnSceneLoaded : delegate
+ OnSceneRedrawn : delegate
+ OnImportFinished : delegate
+ OnBandReloaded : delegate
- RenderPlanes: List<RenderPlaneController>
- loadArgs : SceneLoadArgs
+ Show() : void
+ ReloadScene(args : SceneLoadArgs) : void
+ RedrawScene(args : SceneLoadArgs) : void
+ LoadBand(band : EBand, resolution : EResolution)
+ GetLoadArgs() : SceneLoadArgs
LoadedScenesController
- sceneController: SceneController
- currentlySelectedScene : S2Scene
+ OnSelectedSceneChanged : delegate
+ OnSceneClosed : delegate
+ OnSceneVisibilityChanged : delegate
+ GetSelectedScene() : S2Scene
+ OnSceneClicked(scene : S2Scene) : void
+ OnCloseClicked(scene : S2Scene) : void
+ OnVisibilityClicked(scene : S2Scene) : void
9
RenderPlaneController
+ SatelliteType : SatelliteType
+ Init(parent : S2Scene, tileIndexX : int, tileIndexY : ir
+ Update() : void
+ Redraw() : void
- CheckSize() : void
- OnScroll() : void
SceneLoadArgs
+ redChannel : EBand
+ greenChannel : EBand
+ blueChannel : EBand
+ maskedClasses : List<int>
+ maskColor : Color
+ pinchValues : Dict<int, int>

Figure 5.19.: Class diagram showing the interaction of the *Value Pincher* widget with other components in the scene.

Value Viewer Widget

The *Value Viewing* widget (see Figure 5.20) is responsible for indicating pixel values at a given location in the scene. The widget reports the pixel values for all currently loaded bands. In addition, if the point of interest is covered by multiple scenes, values for all scenes are retrieved. There are two modes of operation:

- Mouse Hover: The widget retrieves values for the pixel which is currently hovered over by the mouse pointer.
- Mouse Click: Values are only retrieved if the user clicks on a pixel. The position is marked with a cross indicator.



Scene Name	Px X	Px Y	Red	Green	Blue	SCL
S2A_MSIL2A_20190128T095251_N0211_R079_T33TWN_20190128T124309	6870,769	3533,183	8992	9416	9768	CloudHP
S2A_MSIL2A_20190329T095031_N0211_R079_T33TWN_20190329T124344	6870,769	3533,183	1294	993	720	CloudLP
S2A_MSIL2A_20190401T100031_N0211_R122_T33TWN_20190401T112631	6870,769	3533,183	1082	855	569	BareSoil

Figure 5.20.: The *Value Viewer* widget, displaying the pixel values of three loaded datasets.

Depending on the selected operation mode, either at a click or at every mouse move event, a ray is cast at the screen-space coordinates of the mouse pointer. For all objects that have been hit (which are `S2Scene` instances), the currently loaded bands are accessed. The actual reflectance values are retrieved from the physical files on the disk, since the corresponding image data may not be loaded by the application. This is the case if a scene is occluded by another scene and has therefore not been updated to reflect the correct display settings. The widget is not affected by masked classes or pinching values, displaying the correct values for each band regardless of settings made in the *Display Settings* and *Value Pincher* widget. In addition, the class of the selected pixel is indicated. The class diagram in Figure 5.21 shows the linkage of the `ValueViewerController` to other scene components.

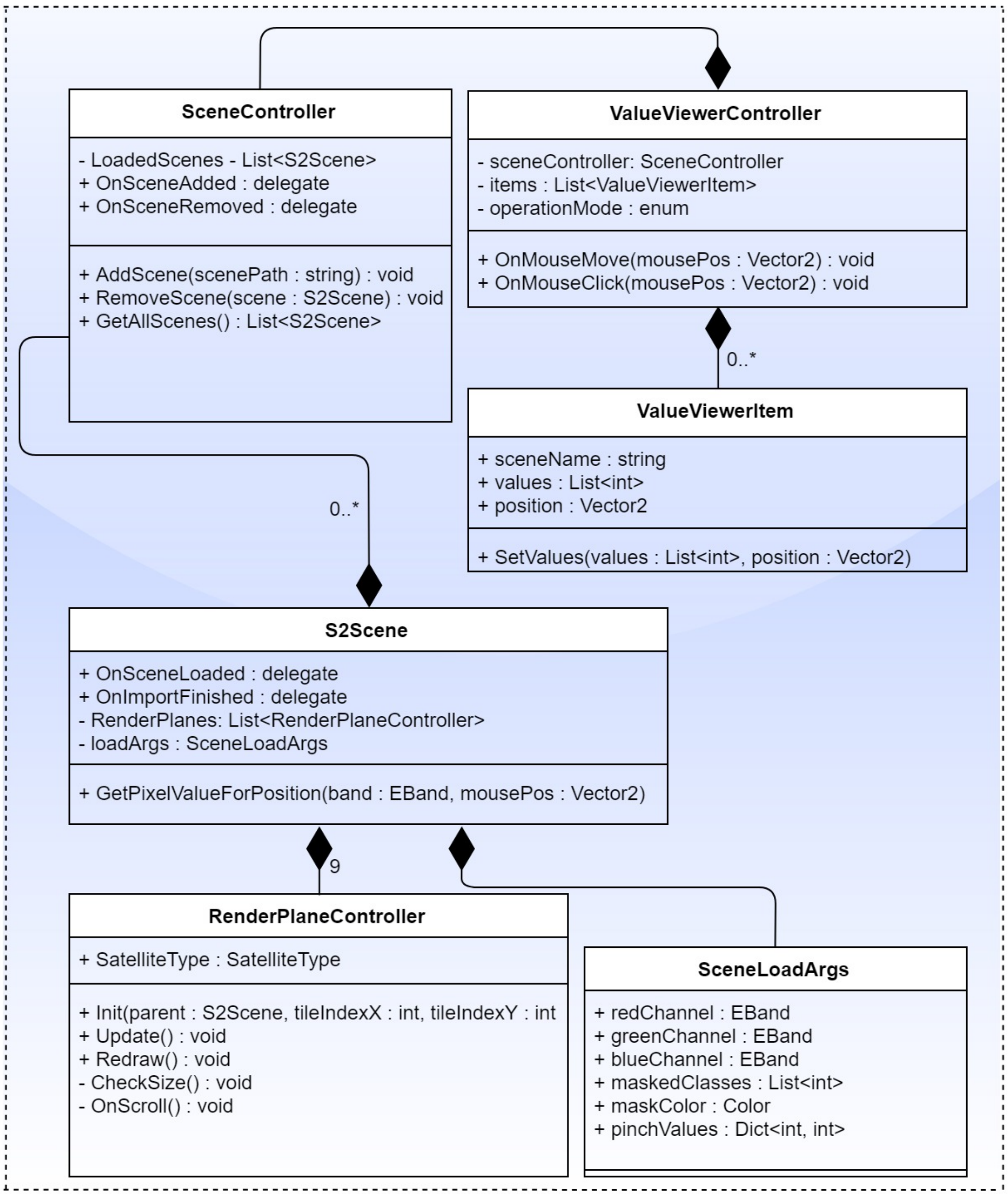

SceneController
- LoadedScenes - List<S2Scene>
+ OnSceneAdded : delegate
+ OnSceneRemoved : delegate
+ AddScene(scenePath : string) : void
+ RemoveScene(scene : S2Scene) : void
+ GetAllScenes() : List<S2Scene>
ValueViewerController
- sceneController: SceneController
- items : List<ValueViewerItem>
- operationMode : enum
+ OnMouseMove(mousePos : Vector2) : void
+ OnMouseClick(mousePos : Vector2) : void
0..*
ValueViewerItem
+ sceneName : string
+ values : List<int>
+ position : Vector2
+ SetValues(values : List<int>, position : Vector2)
0..*
S2Scene
+ OnSceneLoaded : delegate
+ OnImportFinished : delegate
- RenderPlanes: List<RenderPlaneController>
- loadArgs : SceneLoadArgs
+ GetPixelValueForPosition(band : EBand, mousePos : Vector2)
9
RenderPlaneController
+ SatelliteType : SatelliteType
+ Init(parent : S2Scene, tileIndexX : int, tileIndexY : int
+ Update() : void
+ Redraw() : void
- CheckSize() : void
- OnScroll() : void
SceneLoadArgs
+ redChannel : EBand
+ greenChannel : EBand
+ blueChannel : EBand
+ maskedClasses : List<int>
+ maskColor : Color
+ pinchValues : Dict<int, int>

Figure 5.21.: Class diagram showing the interaction of the *Value Viewer* widget with other components in the scene.

Temporal Analysis Widget

The *Temporal Analysis* widget (see Figure 5.22) allows the user to compare parameters of scenes with differing acquisition dates. The displayed values are the respective counts of all pixel classes in the scene, normalized to account for no-data or cloud pixels. The classes are displayed in a graph, with the Y axis showing percentage values and the X axis the acquisition dates of the individual scenes. This is useful when comparing i.e. the amount of vegetated area in a scene over a period of several years. The observations in the graph are ordered by date, regardless of the sequence of loading by the user and positioned relative to their temporal distance to one another. Classes that are of no interest to the user can be hidden by un-checking their respective checkboxes in the widget.

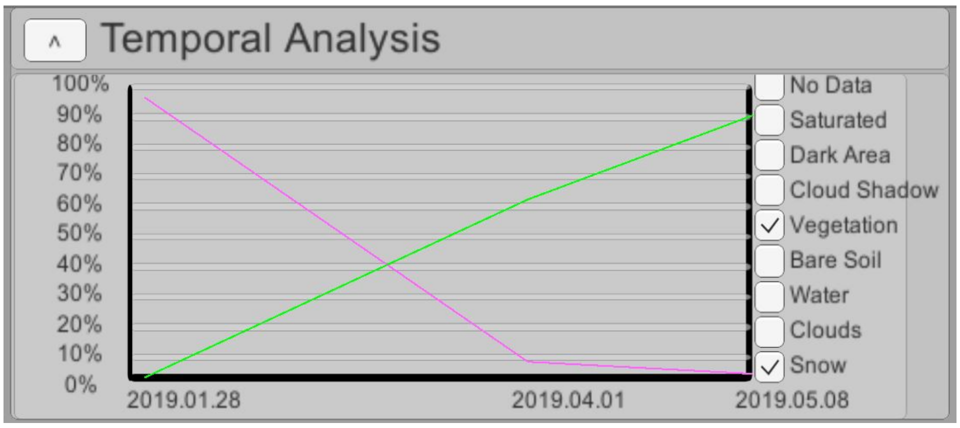


Figure 5.22.: Example of the *Temporal Analysis* widget with three loaded scenes. The widget shows the diminishing snow (purple) and the increase in vegetation (green) from the end of January until 8th of May.

When a scene is loaded or selected, the widget checks if the tile ID is matching, since displaying temporal relationship between observations only makes sense for matching geographical areas. Then, the scene classification statistics are retrieved from the (ordered) `S2Scene` instances (see corresponding class diagram in Figure 5.23). The graph is then redrawn to show statistics for all scenes currently loaded, whose tile ID matches with the scene that is currently selected.

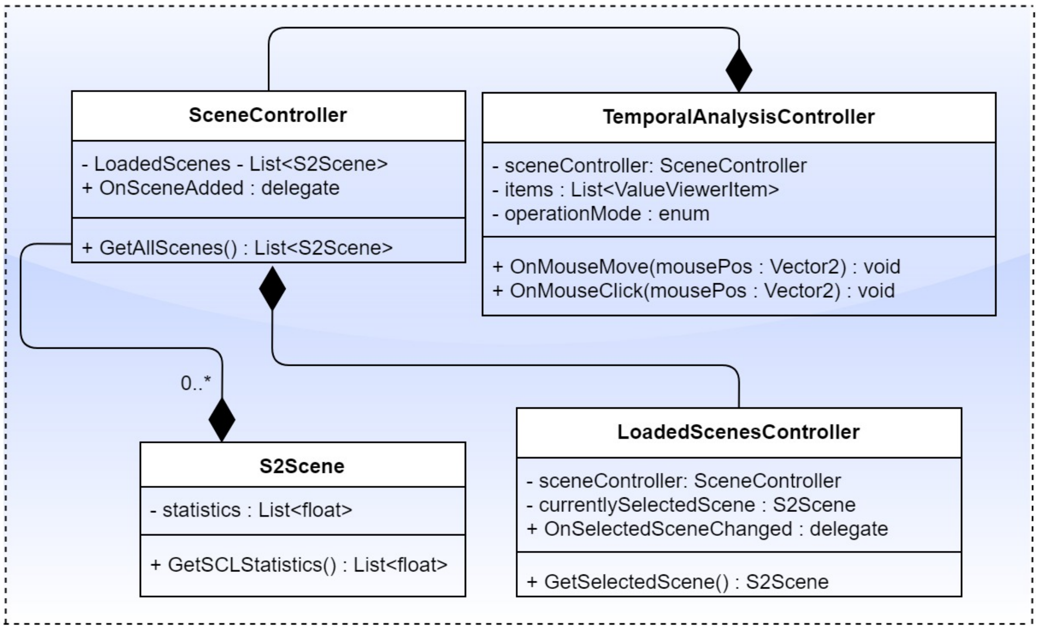

SceneController
- LoadedScenes - List<S2Scene>
+ OnSceneAdded : delegate
+ GetAllScenes() : List<S2Scene>
TemporalAnalysisController
- sceneController: SceneController
- items : List<ValueViewerItem>
- operationMode : enum
+ OnMouseMove(mousePos : Vector2) : void
+ OnMouseClick(mousePos : Vector2) : void
0..*
S2Scene
- statistics : List<float>
+ GetSCLStatistics() : List<float>
LoadedScenesController
- sceneController: SceneController
- currentlySelectedScene : S2Scene
+ OnSelectedSceneChanged : delegate
+ GetSelectedScene() : S2Scene

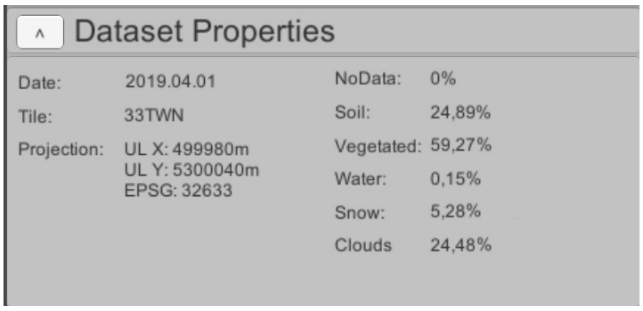
Figure 5.23.: Class diagram showing the interaction of the *Temporal Viewer* widget with other components in the scene.

Dataset Info Widget

This widget is used to display high-level metadata of the currently selected dataset, namely the following properties:

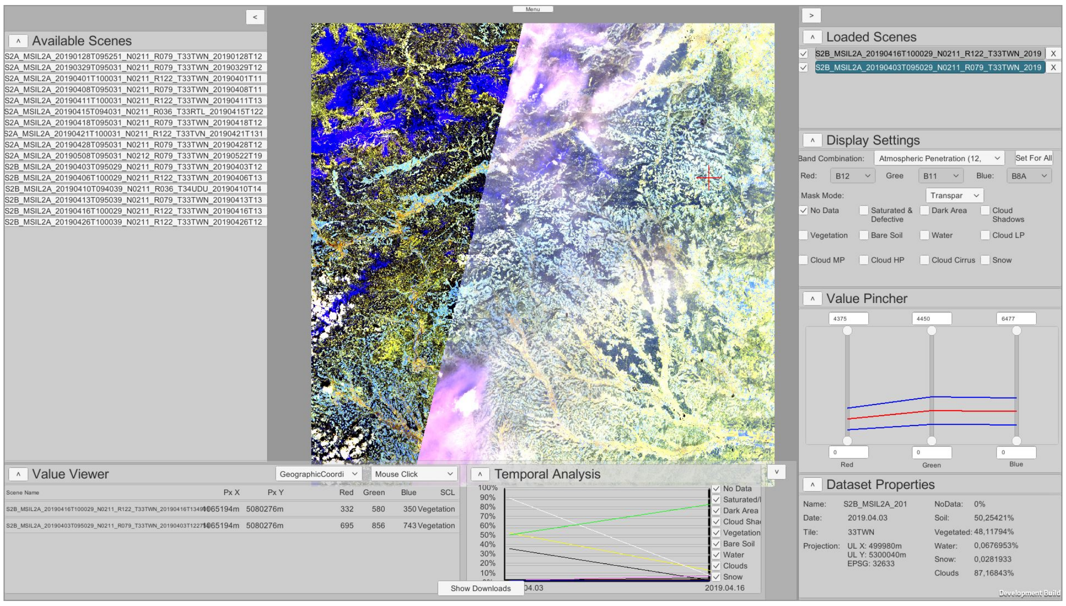
- Acquisition Date
- Tile ID
- Projection Information (coordinates of the upper-left most pixel, EPSG code)
- Normalized percentages of all pixel classes.

This section closes with a screenshot of the *Dataset Info* widget in Figure 5.24 followed by an image of the entire *Scene Explorer* visible in Figure 5.25. The next and final section in this chapter will focus on the implementation and creation of the Earth used in the main menu.



Date:	2019.04.01	NoData:	0%
Tile:	33TWN	Soil:	24,89%
Projection:	UL X: 499980m UL Y: 5300040m EPSG: 32633	Vegetated:	59,27%
		Water:	0,15%
		Snow:	5,28%
		Clouds	24,48%

Figure 5.24.: The *Dataset Info* widget is displaying information about the currently selected scene.



Available Scenes

S2A_MSIL2A_20190128T095251_N0211_R079_T33TWN_20190128T12
S2A_MSIL2A_20190329T095031_N0211_R079_T33TWN_20190329T12
S2A_MSIL2A_20190401T100031_N0211_R122_T33TWN_20190401T11
S2A_MSIL2A_20190408T095031_N0211_R079_T33TWN_20190408T11
S2A_MSIL2A_20190411T100031_N0211_R122_T33TWN_20190411T13
S2A_MSIL2A_20190415T094031_N0211_R036_T33RTL_20190415T122
S2A_MSIL2A_20190418T095031_N0211_R079_T33TWN_20190418T12
S2A_MSIL2A_20190421T100031_N0211_R122_T33TVN_20190421T131
S2A_MSIL2A_20190428T095031_N0211_R079_T33TWN_20190428T12
S2A_MSIL2A_20190508T095031_N0212_R079_T33TWN_20190522T19
S2B_MSIL2A_20190403T095029_N0211_R079_T33TWN_20190403T12
S2B_MSIL2A_20190406T100029_N0211_R122_T33TWN_20190406T13
S2B_MSIL2A_20190410T094039_N0211_R036_T34UDU_20190410T14
S2B_MSIL2A_20190413T095039_N0211_R079_T33TWN_20190413T13
S2B_MSIL2A_20190416T100029_N0211_R122_T33TWN_20190416T13
S2B_MSIL2A_20190426T100039_N0211_R122_T33TWN_20190426T12

Value Viewer — GeographicCoordi — Mouse Click

Scene Name	Px X	Px Y	Red	Green	Blue	SCL
S2B_MSIL2A_20190416T100029_N0211_R122_T33TWN_20190416T1349	1065194m	5080276m	332	580	350	Vegetation
S2B_MSIL2A_20190403T095029_N0211_R079_T33TWN_20190403T1227	1065194m	5080276m	695	856	743	Vegetation

Loaded Scenes

✓ S2B_MSIL2A_20190416T100029_N0211_R122_T33TWN_2019 X
✓ S2B_MSIL2A_20190403T095029_N0211_R079_T33TWN_2019 X

Display Settings

Band Combination: Atmospheric Penetration (12, — Set For All
Red: B12 Gree B11 Blue: B8A
Mask Mode: Transpar
✓ No Data ☐ Saturated & Defective ☐ Dark Area ☐ Cloud Shadows
☐ Vegetation ☐ Bare Soil ☐ Water ☐ Cloud LP
☐ Cloud MP ☐ Cloud HP ☐ Cloud Cirrus ☐ Snow

Value Pincher

Red	Green	Blue
4375	4450	6477
0	0	0

Dataset Properties

Name:	S2B_MSIL2A_201	NoData:	0%
Date:	2019.04.03	Soil:	50,25421%
Tile:	33TWN	Vegetated:	48,11794%
Projection:	UL X: 499980m UL Y: 5300040m EPSG: 32633	Water:	0,0676953%
		Snow:	0,0281933
		Clouds	87,16843%

Figure 5.25.: The *Scene Explorer* with a loaded scene and all widgets.

5.3. Main Menu

The main menu is the starting point when launching the application. It features a rendering of Earth using the Blue Marble dataset and allows the user to select whether she or he wants to access the Copernicus server for data retrieval or inspect already downloaded scenes. The generation of the earth sphere is explained in this section.

5.3.1. UV-Mapping

UV-Mapping is the process of mapping a 2D texture (i.e. an image) onto an arbitrary 3D surface.

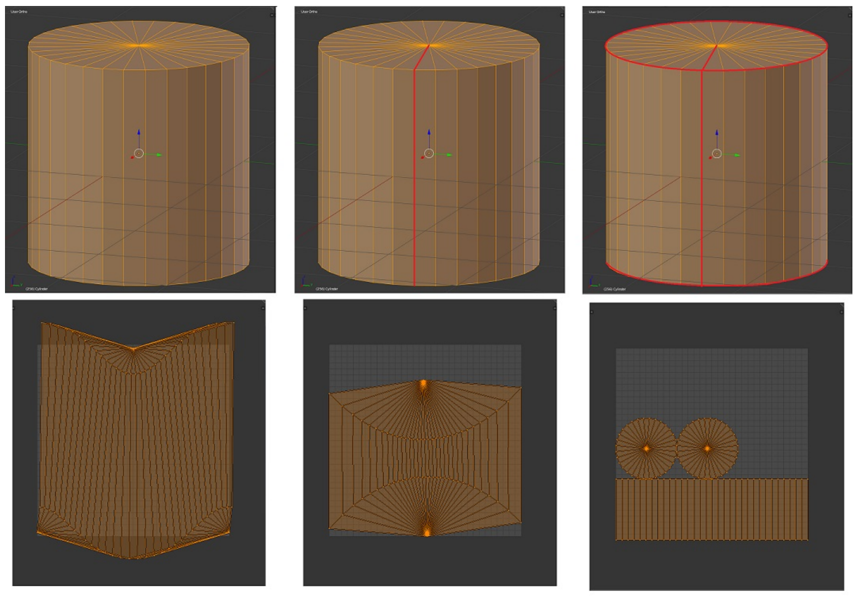


Figure 5.26.: UV Unwrapping. The upper row shows the cylinder mesh in three stages. Left: Original mesh with no seams (cutlines). Middle: One cut along the cylinder. Right: Fully seamed cylinder. The lower row shows the unwrapped mesh according to the mesh above. Only the unwrap on the right shows no distortion.

The used transformation matrix is called UV-Map, where *U* and *V* index the image. As described by Mullen (2011), the process of UV-mapping involves unwrapping the mesh, creating the texture and applying the texture to the mesh. The unwrapping step is crucial. The goal is to project (flatten) the 3D object with minimal distortion. Depending on the object, it is essential to define so-called seams which can be seen as cut-lines to unwrap the mesh. An example where this is necessary can be seen in Figure 5.26.

5.3.2. Low-Resolution Earth

The aforementioned Blue Marble dataset provides resolutions of up to 500m. In comparison to the later used Sentinel-2 datasets, which, as already explained, provide a spatial resolution of up to 10m, the generation of the globe described in this section will henceforth be denoted as "Low-resolution Earth" (LRE).

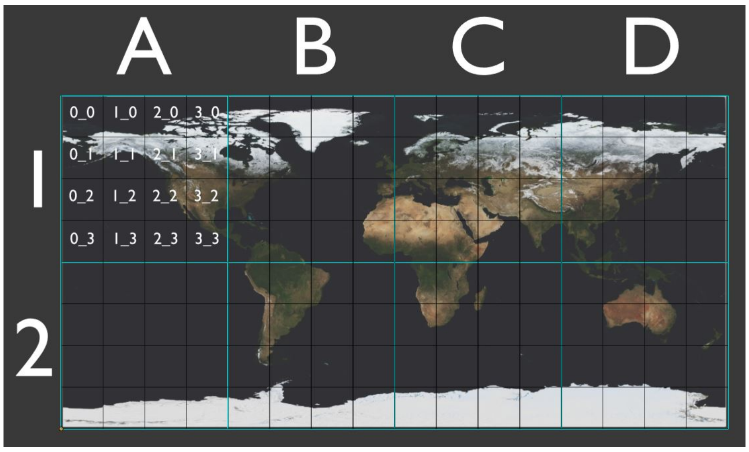


Figure 5.27.: The applied tiling grid. For visibility, the sub-tiling is only shown for the first octant.

The files are divided into eight sub-images, where each has a pixel resolution of 21600x21600. To overcome various single texture size-limits within Blender Unity3D and most modern GPUs, the images have been further divided into smaller chunks with a pixel resolution of 5400x5400. This results in a total of 128 files. The tiling grid is shown in Figure 5.27. The upper-left most image is denoted `A1_0_0`, while i.e. the bottom-right image in octant `C2` is denoted `C2_3_3`. In order to assemble a Sphere with the explained tiling, 128 square planes are created, each with the corresponding texture tile applied as UV-Color texture. Algorithm 1 shows the generation algorithm. The Blue Marble dataset also provides topographic maps (which can be used as normal map) in a resolution of 1km (10800x10800). For specularity information (amount of shininess), NASA provides a land mask.[4]

[4]Land Mask: https://eosweb.larc.nasa.gov/project/gewex-rfa/documents/land-mask

Algorithm 1: Generation of a flat earth.

```
Input: $hemi$    // Either 1 for northern or 2 for southern hemisphere
Function CreateEarthHemisphere($hemi$):
    $mesh = $ Blender.CreateMesh($hemi$)    // Create mesh with name $hemi$
    $tiles = [A, B, C, D]$
    $position = 0$
    $tilePos = -1$
    for $subTileX \leftarrow 0$ to $16$ do
        for $subTileY \leftarrow 0$ to $4$ do
            if $position \bmod numTilesX = 0$ then
                $tilePos \leftarrow tilePos + 1$
            end
            $currentTile = tiles[tilePos] + hemi + tileX \bmod 4 + tileY$

                /* Create the plane defined by the for vertex corners */
            $vUL$ = Vertex( $tileX + 0.5, -tileY + 0.5, 1$)
            $vLL$ = Vertex( $tileX - 0.5, -tileY - 0.5, 1$)
            $vLR$ = Vertex( $tileX + 0.5, -tileY - 0.5, 1$)
            $vUR$ = Vertex( $tileX - 0.5, -tileY + 0.5, 1$)

            $mesh$.AddFace( $vUL, vLL, vLR, vUR$)

                /* Create a new material and assign textures */
            $material$ = $mesh$.AddMaterial( $currentTile$)
            $material$.AddColorTexture( $COL + currentTile$)
            $material$.AddNormalTexture( $NRM + currentTile$)
            $material$.AddSpecTexture( $SPEC + currentTile$)

            $position \leftarrow position + 1$
        end
    end
    return $mesh$

$northHemisphere$ = CreateEarthHemisphere(1)
$southHemisphere$ = CreateEarthHemisphere(2)
$Earth$ = Blender.Join($northHemisphere, southHemisphere$)
```

The constructed plane has 128 faces and maps an equal number of textures on each face according to the described tiling. To generate a sphere out of this plane, it is first subdivisioned using a simple face-split (see Figure 5.28) to increase the number of vertices, which allows for a more realistic-looking sphere. For the following description of the warping process, the un-subdivisioned mesh is shown solely for the sake of visibility.

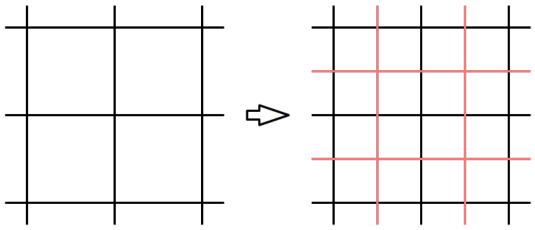


Figure 5.28.: Simple face-split. Left: Original quad faces. Left: inserted edges (red) create four new faces for each original face.

The subdivisioned plane is warped twice around different axes to receive a sphere. Assuming the plane is planar to the X, Y axes, its short side is warped around the X axis by $1\pi rad$ (see Figure 5.29). The warp transformation warps vertices around a pivot point with a given angle (Blender, 2019).

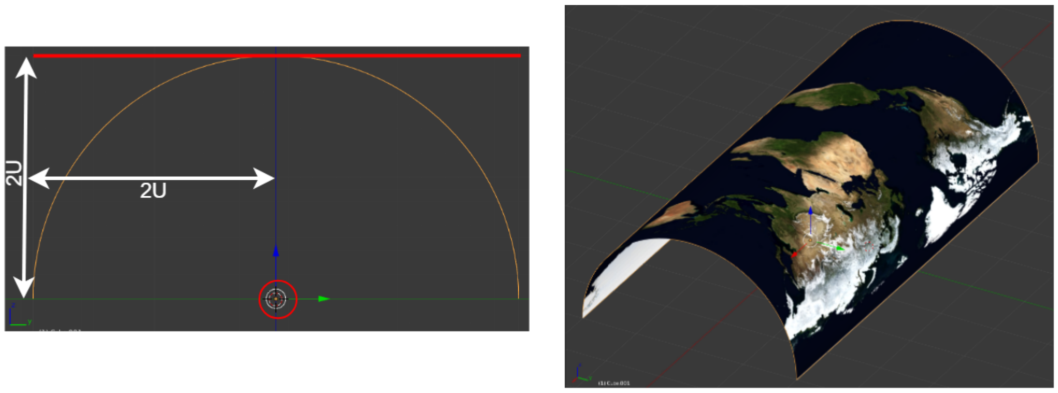


Figure 5.29.: Warping the plane. Left: Original plane shown in red, $2U$ above the coordinate center, plane is $4U$ wide. The plane is warped around the (circled in red) pivot point by $1\pi rad$ The warped plane is outlined in orange. Left: 3D view of the warping result.

The result is a half-open cylinder with no end-caps. To create a sphere, this mesh has to be warped again, this time around the Y axis, as shown in Figure 5.30.

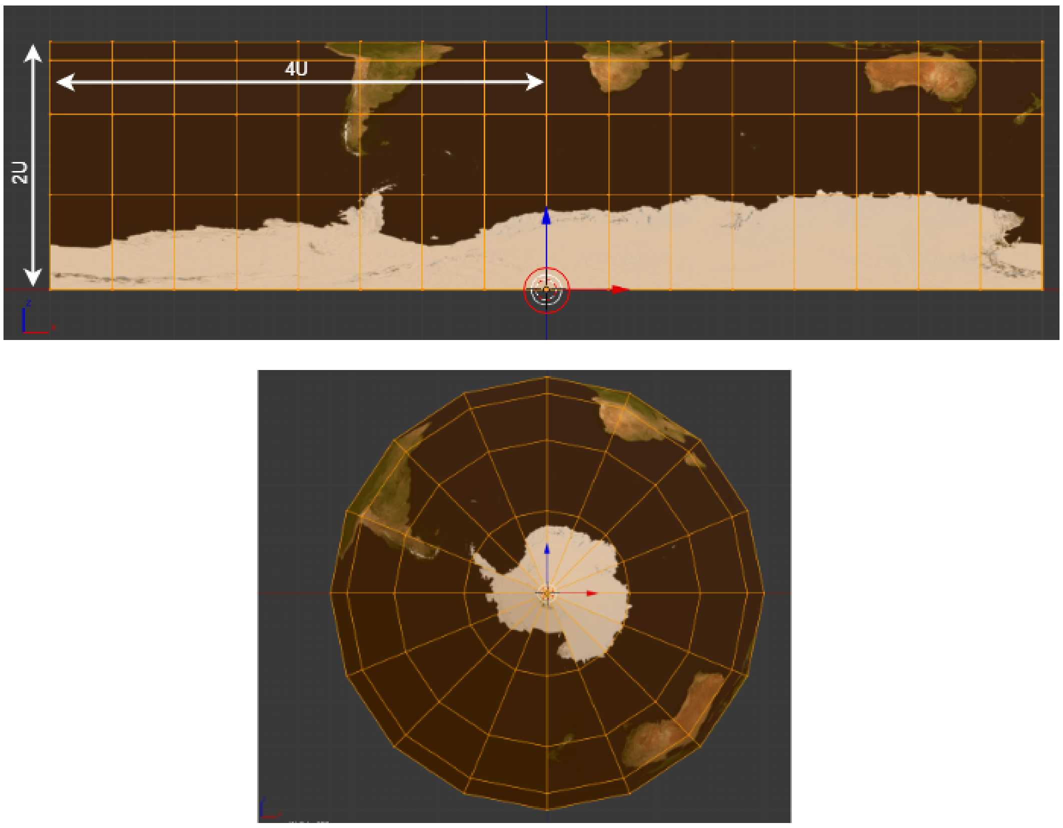


Figure 5.30.: The second warp. Upper image: The warping result of the previous step shown in X, Z view (red circle: pivot point). Lower image: resulting sphere after applying a warping angle of $2\pi rad$.

The same exact process as described above has been applied for a second sphere which maps a cloud texture (also provided by NASA[5]), although in a smaller resolution. The raw-cloud plane is constructed using 32 planes (compared to the 128 of the surface LRE) and scaled up ($x1.02$) to create an appealing cloud layer. An image of the LRE imported and in Unity is shown in Figure 5.31.

[5]Blue Marble Clouds: https://visibleearth.nasa.gov/view.php?id=57747

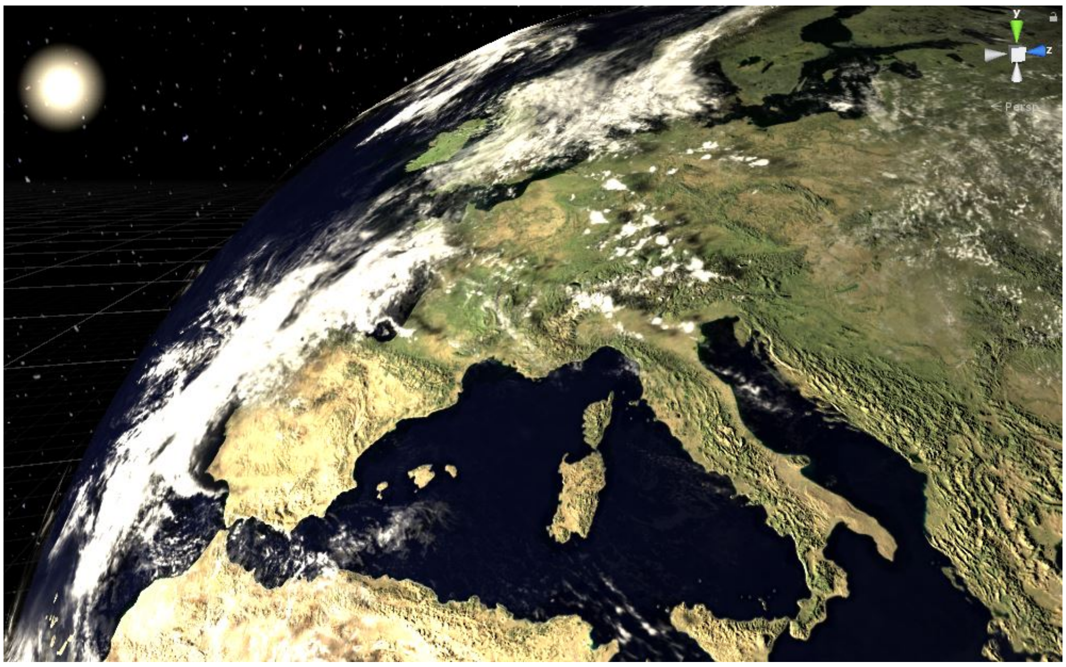


Figure 5.31.: The finished LRE sphere in Unity (view over Europe).

5.3.3. LRE Shading

In addition to the aforementioned color texture, a night map[6] is used to simulate a day-night transition according to the suns incidence angle. For this, a custom surface shader was implemented. Surface shaders are available in Unity as a combination of vertex and fragment shader that take advantage of ShaderLab extensions to automate code that is commonly used (Doppioslash, 2017, chap. 1, p. 8). As input properties, day and night textures, normal maps and ocean masks (for specularity), as well as the UV layout are provided. The actual blending between day and night texture is handled by a custom lighting function, which is adapted from the standard

[6]NASA Earth's City Lights: https://visibleearth.nasa.gov/view.php?id=55167

lighting shader (see Algorithm 2).

Algorithm 2: Blending Day and Night textured according to amount of light.

```
Input: so                                   // Output from the surface shading.
       viewDir                              // View direction in world space.
       gi                                   // Global Illumination information.
Function EarthLighting(so, viewDir, gi):
    // Calculate the Albedo Color from DayColor and Specular Texture.
    $so.albedo = $ Unity.CalcAlbedo($so.dayColor, so.specular$)

    $brightness = $ max($0$, dot($gi.light.dir, so.normal$))
    $ncAmount = $ lerp($so.nightColor$, $0$, clamp($brightness$, $-1.0$, $1.0$))
    $outputColor = $ Unity.BRDFPBS($so.albedo, so.specular, gi$)
    $outputColor.rgb += ncAmount$
    return $outputColor$
```

This allows for a realistic blend between day and night texture, according to the amount of light present at the current point on the sphere. The lesser the amount of light received, the higher the additive amount of the night texture added to the final output (see Figure 5.32).

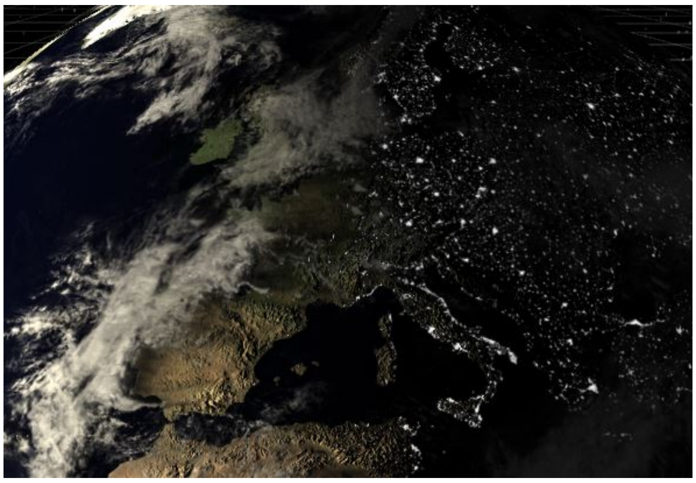


Figure 5.32.: Night cycle blending with custom shader (view over Europe).

5.4. Summary

This chapter described the implementation of the *Sentinel-2 Explorer* framework. Following the design description of the previous chapter, the framework is split into three parts: The *Earth Explorer* for downloading data, the *Scene Explorer* for analyzing downloaded datasets and a main menu to connect all components. We explained, how the *Earth Explorer* makes use of a Plate Carrée Map to ease querying specific locations on the planet. In addition, a detailed description of all implemented tools for analyzing individual scenes has been given. The last section in this chapter explained how the globe visualization in the main menu was created by warping a rectangular polygon to a sphere. The next chapter will deal with the evaluation of the implemented framework.

6. Evaluation

The success of a framework such as the *Sentinel-2 Explorer* largely depends on user acceptance. In order to identify strong suits and weaknesses of the system, an evaluation has to be performed. This chapter explains in detail how this study was designed and carried out.

6.1. Methodology

Two groups participated in a study to evaluate the *Sentinel-2 Explorer*. One group consisted of people that are well-versed in general computer usage but do not have trained skills in using remote sensing tools. The other group was composed of remote-sensing specialists on levels of high proficiency. They are accommodated to use various existing tools in order to fulfill their tasks. This two-group approach is commonly used to identify usability problems and distinguish between problems that are caused by the general scope of remote sensing and problems that are indeed rooted in the design of the framework itself, indicating further room for improvement.
Before starting work on the actual evaluation tasks, a pre-questionnaire was given to each of the users. After completing the tasks, users where asked to fill out a post-questionnaire capturing their emotions and opinions. During the actual evaluation, the participants were under the supervision of an expert. This expert also provided a quick introduction to the system to explain basic features and functionality, and (if necessary) introduced the participants to the topic of remote sensing.
The evaluation focuses on these aspects:

- The usefulness and effectiveness of the implemented tools.
- The usability and user-friendliness of the provided tools.

- The overall system performance regarding loading times and how this affects the users' perception of the system.

The questionnaires were designed to capture all relevant information about the system and the user. In the pre-questionnaire, the users were asked about demographic information, such as their age, profession and general knowledge about computer usage and their proficiency in remote sensing. During the post-questionnaire, users were asked whether they were satisfied with the framework, what they would improve and which features they found especially helpful. The Computer Emotion Scale (CES) and System Usability Scale (SUS) were used to measure user emotion and usability rating, respectively.

6.1.1. Computer Emotion Scale

Developed by Kay and Loverock (2008), the purpose of this evaluation tool is to measure emotions when learning the usage of new software. By answering twelve questions about how the user feels when using the software, four emotional responses are evaluated: anger, anxiety, happiness and sadness.

6.1.2. System Usability Scale

The SUS is used to measure the usability, efficiency and effectiveness of a software system. Users are given the opportunity to answer ten questions after working with the software. The result of the evaluation tool is a score between 0 and 100 (Brooke et al., 1996).

6.2. Process

Both groups were given the same tasks, consisting of various typical activities when working with satellite data. The tasks can be split up into two groups: retrieval of data (evaluation of the *Earth Explorer*) and operations that have to be performed on the retrieved data in the *Scene Explorer*.

6.2.1. Earth Explorer

In this first part, the users were asked to make themselves comfortable with the *Earth Explorer*. They could explore and make themselves familiar with all available features while a supervisor explained the theoretical background about the Sentinel-2 satellites, the tiling grids and which query parameters are available. The first assignment was to retrieve all tiles covering various countries, making the user familiar with the interface. The subsequent task asked users to query all datasets that matched a given tile ID, date range and cloud coverage. Following that, the users were asked to exclude various datasets manually from the download, according to given parameters (such as no-data percentage).

6.2.2. Scene Explorer

The second part of the evaluation focused on analyzing retrieved data. First, the users were asked to load a scene with a specific tile ID and date. They then were tasked to investigate the coverage of various pixel classes in the scene, before proceeding to switch between band combinations in order to distinguish different land coverage types. The next task required the users to make use of value pinching, scilicet, manually hiding clouds and snow in the scene. The next step was to load a second scene (with the same tile ID but a different acquisition date) and mask all clouds. This time, the users were free to choose which method they wanted to use. The final assignment was to find a pixel that was classified as vegetation in both scenes, retrieve the reflectance values, and answer whether the overall percentage of vegetation increased or decreased between the two observations.

6.3. Participants

All participants were aged between 23 and 35, with a $M = 28, SD = 3.21$ years. All were proficient in using computers, being required to use them on a daily base, either by their field of study or their line of work. As already mentioned, the users were divided into two groups: remote sensing experts and non-experts.

6.3.1. Experts

Six experts were selected to evaluate the implemented framework. All of them are proficient in retrieving and analyzing satellite data, with almost all of them having worked with Sentinel-2 data before. Regarding their skill in remote sensing, on a scale of one to five, with five meaning "very experienced" and one meaning "no experience at all", the expert group has organized around $M = 4.33$, which was to be expected.

6.3.2. Non-Experts

This group also consisted of six participants who rated their knowledge of remote sensing at an average of 1.33, which clearly indicates a large knowledge gap between the groups. It is worth noting that only one participant did not know that light is a form of electromagnetic radiation, and that its color is defined by the wavelength of the radiation. This ensures that most participants, albeit not being experts, had a basic understanding of the reason satellite images consist of reflectance values and what meaning they convey. Most participants in this group were students and/or software developers with the one exception of a chemist.

6.4. Results

The results consist of a series of open-ended questions, answered in the form of textual input, regarding general preferences and dissatisfactory outcomes in the evaluated framework. The users were also asked about recommendations and ideas for improvements. The questionnaire was split into two parts, one for the *Earth Explorer* and one for the *Scene Explorer*, each adapted to account for the different features presented in each component.

6.4.1. Earth Explorer Results

All participants responded positively to the question "How was it?", with no clear differences between experts and non-experts. Most felt intrigued by the *Earth Explorer*. Similarly, all users rated the loading speeds as fast and the user interface as responsive and intuitive when asked about what they liked most about the component. In addition, some users mentioned the integration of a map of Earth with the capability to browse for specific tiles visually as a good feature. Regarding what users didn't like, the non-expert group criticized missing help-indicators such as tool-tips or small text elements.
The expert group criticized that panning with the middle mouse button a rather intuitive experience, indicating a clear preference to the panning modes they are used to. When asked which features they would like to see in the *Earth Explorer*, the non-expert group wished for a feature that allows to search directly for tiles overlapping a specific country. Almost all experts expressed their desire for a tool that allows drawing an area of interest, that is, a polygon for which encompassed tiles are appended to the query. One expert requested a way to sort results according to date or cloud coverage. Users were asked to answer, on a Likert scale (Likert, 1932), between one ("I strongly agree") and five ("I strongly disagree"), how they think about certain aspects of the *Earth Explorer*. Figure 6.1 shows the evaluation results of these questions. In addition, Table 6.1 contains mean and SD values for each question.

Question	Overall		Experts		Non-Experts	
	AVG	SD	AVG	SD	AVG	SD
The map was helpful for locating tiles.	4.67	0.62	4.50	0.76	4.83	0.37
The Earth Explorer felt fast and responsive.	4.67	0.62	4.50	0.76	4.83	0.37
Querying datasets with the Earth Explorer was easy.	4.58	0.64	4.50	0.76	4.67	0.47
Downloading datasets with the Earth Explorer was easy.	4.67	0.47	4.67	0.47	4.67	0.47

Table 6.1.: Average and SD for each question in the *Earth Explorer* post-questionnaire.

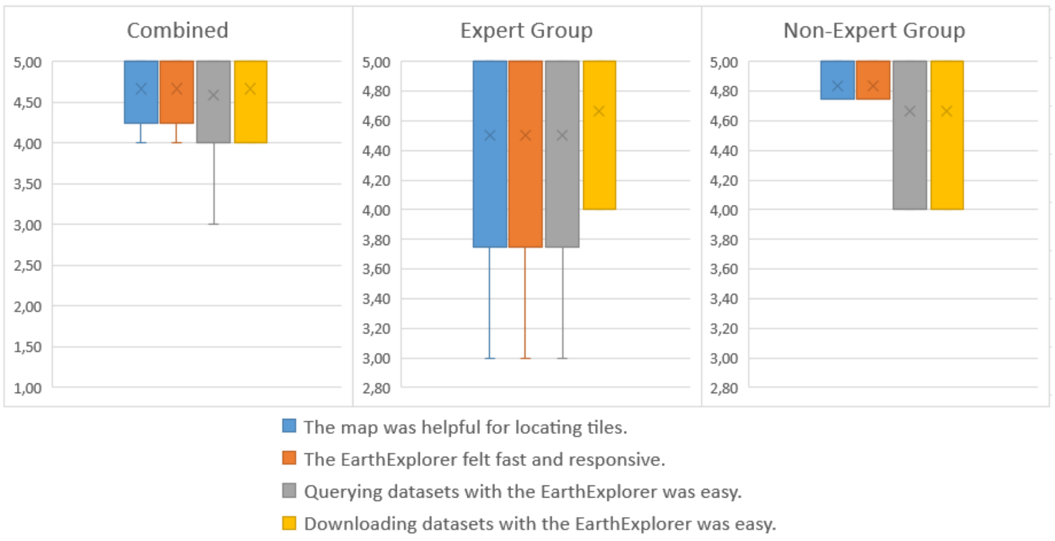


Figure 6.1.: Box plot of the evaluation questions for the *Earth Explorer*.

6.4.2. Scene Explorer Results

All participants felt that the *Scene Explorer* was intuitive to use and rated the overall experience as “interesting”. Most users liked the possibility to directly mask specific pixel classes and that the system feels manageable despite the plenitude of features. One expert noted that the *Scene Explorer* allows for rapid inspection of Sentinel-2 scenes with various visualization presets. One participant in the non-expert group did not understand the purpose of the *Value Pincher* widget, indicating that the widget's user interface needs improvement. Regarding new features, users suggested the inclusion of map providers such as Google Earth or OpenStreetMap. Experts suggested that adding vegetation indices would greatly increase the functionality of the tool, allowing for in-depth analysis of datasets. Another expert wished for a quicklook preview of available scenes, similar to the preview in the *Earth Explorer*. Two experts and two non-experts had no suggestions for future features. Regarding the *Value Pincher*, users felt that there is some wasted space which could be used for additional information. Experts thought that adding "Reset" and "Invert" functionality to the *Value Pincher* would increase usability, further supporting the earlier indication. Possible use-cases for the *Scene Explorer* as identified by the users include the creation of zoning maps, STEM education, land use analysis, and agricultural harvest prediction. It is worth noting that all experts would use the *Scene Explorer* in the future. Figure 6.2 shows the evaluation results of the questions that were given to the participants regarding their opinion toward the *Scene Explorer* and it's associated components. In addition, Table 6.2 shows all mean and SD values for the participant rating.

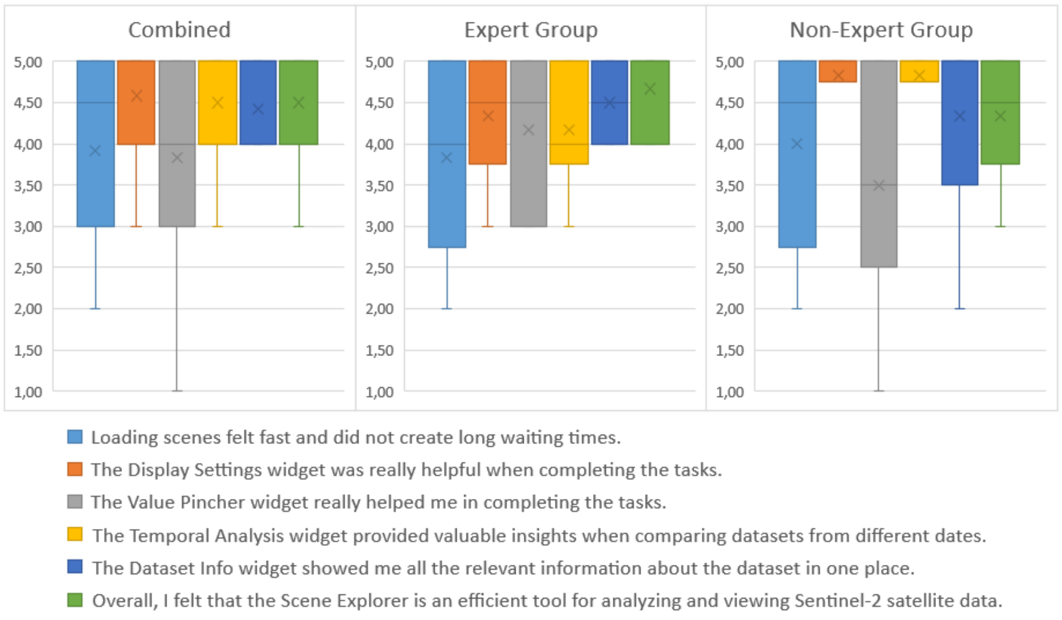


Figure 6.2.: Box plot of the evaluation questions for the *Scene Explorer*.

Question	Overall		Experts		Non-Experts	
	AVG	SD	AVG	SD	AVG	SD
Loading scenes felt fast and did not create long waiting times.	3.92	1.19	3.83	1.21	4.00	1.15
The Display Settings widget was really helpful when completing the tasks.	4.58	0.64	4.33	0.75	4.83	0.37
The Value Pincher widget really helped me in completing the tasks.	3.83	1.21	4.17	0.90	3.50	1.38
The Temporal Analysis widget provided valuable insights when comparing datasets from different dates.	4.50	0.65	4.17	0.69	4.83	0.37
The Dataset Info widget showed me all the relevant information about the dataset in one place.	4.42	0.86	4.5	0.5	4.33	1.11
Overall, I felt that the Scene Explorer is an efficient tool for analyzing and viewing Sentinel-2 satellite data.	4.50	0.65	4.67	0.47	4.33	0.75

Table 6.2.: Average and SD for each question for in the *Scene Explorer* post-questionnaire.

6.4.3. Usability Results

As already mentioned, the SUS by Brooke et al. (1996) is used to measure the overall usability of the *Sentinel-2 Explorer* framework.
For the *Earth Explorer*, ten out of twelve participants rated the usability according to the SUS with 70 or more. Overall, this component achieved a score of 81.88, with a standard deviation of 11.18. The expert group score with $M = 80.83, SD = 11.33$ is slightly lower than the overall score. The non-expert group scores the component at a mean of 82.91 ($SD = 29.23$).
Regarding the *Scene Explorer*, eleven out of twelve people scored the usability with 70 or higher. The overall mean is 81.04, with a standard deviation of 11.43. This time, the expert group exhibits a higher mean at 85.41 ($SD = 9.24$), while the non-experts score the system at a mean of 77.08 ($SD = 30.54$).

6.4.4. Motivation Results

Participants were asked to answer a questionnaire regarding the emotions they have experienced while using the system. Using the Computer Emotion Scale evaluation developed by Kay and Loverock (2008), the four archetypal emotions happiness, sadness, anxiety and anger are captured. Figure 6.3 shows the CES score achieved by the system.

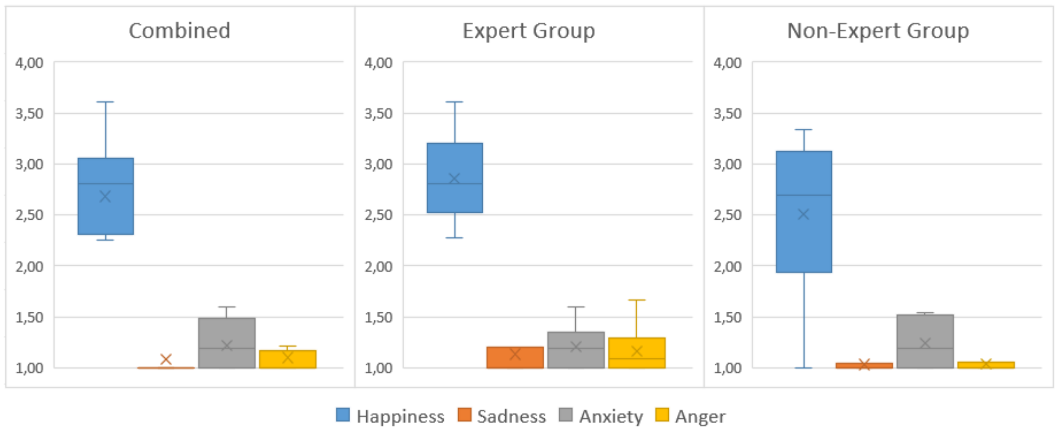


Figure 6.3.: Box plot of the overall CES score regarding the *Sentinel-2 Explorer* framework.

The CES results for the *Earth Explorer* and *Scene Explorer* are outlined in Table 6.3 and Table 6.4, respectively.

Emotion	**Overall**		**Experts**		**Non-Experts**	
	AVG	**SD**	**AVG**	**SD**	**AVG**	**SD**
Hapiness	2.73	0.73	2.86	0.43	2.6	0.97
Sadness	1.08	0.19	1.10	0.23	1.06	0.40
Anxiety	1.24	0.26	1.21	0.26	1.26	0.43
Anger	1.08	0.20	1.17	0.25	1.00	0.42

Table 6.3.: CES scoring for the *Earth Explorer*.

Emotion	Overall		Experts		Non-Experts	
	AVG	**SD**	**AVG**	**SD**	**AVG**	**SD**
Hapiness	2.63	0.65	2.86	0.43	2.41	0.75
Sadness	1.08	0.28	1.17	0.37	1.00	0.00
Anxiety	1.21	0.2	1.20	0.16	1.22	0.24
Anger	1.12	0.22	1.17	0.25	1.07	0.16

Table 6.4.: CES scoring for the *Scene Explorer*.

Overall, happiness, as identified through the emotions of satisfaction, excitement and curiosity, can be beheld as the dominant emotion while using this framework. Closer inspection of the three impacting emotions yields that users achieved a satisfaction rating of $M = 3.04, SD = 0.78$, marking it the dominant emotion before curiosity ($M = 2.79, SD = 0.96$) and excitement ($M = 2.33, SD = 0.85$), respectively.

6.5. Discussion

The evaluation of the *Sentinel-2 Explorer* framework showed that the system is very well accepted by the users. Both experts and non-experts alike, felt very happy using the system, which is a prerequisite to introducing such a system. It is worth mentioning that with an average score of 1.00, anger is the lowest rated emotion in the evaluation. All participants completed the given tasks within 25 minutes, although no time constraints were given during the evaluation process. Both user groups stated that the implemented tools and features are considered helpful for accomplishing a variety of analyses.
Regarding the *Earth Explorer*, all users felt very confident querying and downloading datasets. They stated, that retrieving tile IDs by clicking on a point on the screen was heeded intuitive. However, suggestions by the participants should be considered as they can be valuable to enhance both functionality and effectiveness of the *Earth Explorer*. The experts all coincided that they would like to use the *Scene Explorer* in the future. Especially the *Display Settings* widget was very well received by all participants, although there are recommendations and suggestions for improvement. There is still

room for improvement, especially regarding the *Value Pincher* widget, as it's purpose was not clear to every user. The experts rated the usability of the *Scene Explorer* significantly higher than participants in the non-expert group, indicating that the user interface shows a clear bias towards remote sensing experts. Loading times were perceived as fast, but users pointed out that loading times of even a few hundred milliseconds would impact their concentration on the tasks negatively.
In conclusion, recalling the three evaluation tasks from the beginning of this chapter, the evaluation showed that the presented framework is indeed useful for retrieving and analyzing Sentinel-2 satellite data. The tools are effective for their intended purpose and achieved consistently high usability ratings. Unity3D proved to be an effective base framework for handling large-scale image data, as users rated loading times and speed as sufficiently high.

7. Lessons Learned

In this chapter, we take a look at insights gained during development and evaluation of this framework. This wisdom should be taken into consideration for future work, relating this chapter closely to the next where we will focus on possible improvements and future work.

7.1. Development

Handling satellite image data is not a trivial task. Considering large quantities of data, the many dimensions (spectral bands) and thereby different visualization modes, it becomes clear that the foundation of such a framework has to be capable of many different things before a solid system can be implemented. Unity3D demonstrated to be a very valuable base, being capable to offload costly calculations onto the GPU, reducing loading times significantly. Although using such an engine has many advantages, designing an efficient user interface was a cumbersome task. Unity3D's graphical user interface system is powerful and fast but designing complex widgets that interact closely with data in a manner that represents a professional imaging application, proves to be a challenging task and incommodious at times. Many widgets (such as a double-ranged slider) or graphs have to be implemented from scratch, taking valuable development time away from more beneficial features, which will be debated in the next chapter. However, Unity3D turned out to be a good choice for implementing this framework.
The chosen design strategy, outlined in Chapter 4, proved to be very effective. The decision of splitting the two main tasks, downloading and visualizing into separate scenes, namely the *Earth Explorer* and *Scene Explorer* allowed for a very clean and streamlined user experience. All requirements have

been implemented and some, that is, loading times, have been surpassed. In addition, the use of custom shader programming for visualizing the satellitedata offers a wide variety of possibilities, some of which will be discussed in Chapter 8.

7.2. Evaluation

The use of the standardized SUS and CES questionnaires proved to be very valuable as it makes future comparisons easier. Each of the questionnaires reveal different aspects about the system; The SUS focuses on the usability of the system while the CES specifically aims at capturing the users' emotions during the use of the software. Together, they reveal a much more complete image, representing real-world applications more closely. These two evaluation methods, together with the use of open-ended questions, allow for a thorough evaluation, as open-ended questions allow the participant to add his or her own suggestions, improvements, preferences and dissatisfactions to the evaluation. They can add their own thoughts, greatly enhancing the value of their feedback. The users' proved to be quite willingly to add their own suggestions. Future evaluations should make use of this fact by allowing for more open-ended answers, whilst asking more specific questions.

8. Future Work

When designing this framework, the original goal was to provide an efficient tool for handling and visualizing Sentinel-2 data. This goal has been met and according to the evaluation in Chapter 6, users were overall very confident and happy using this system. However, there are things that could be improved further. The *Earth Explorer* already offers a large feature set to query and acquire datasets, but some useful features could be considered in the future:

- Region Of Interest: At the moment, without advanced knowledge of the OpenSearch API, it is not possible to query more than one tile at a time. As mentioned by many users during the evaluation, a feature that allows drawing a polygon or select an entire country would prove beneficial.
- Pausing/Resuming Downloads: It is not possible to pause downloads and resume them later on.
- Handling of very large query responses: Although stable and fast, the framework suffers performance issues when dealing with more than a few thousand responses, the user interface becomes slow. Pooling responses could provide a solution to this problem.
- Querying of local datasets: At the moment, the *Earth Explorer* does not provide the ability to query only already downloaded datasets.

The *Scene Explorer* provides a good base for future enhancements and new features. Although loading times and efficient use of system resources is already ensured by the loading system outlined in Section 5.2.1, there is headroom for further performance increases:

- More intermediate resolutions: Currently, only three stages of granularity are possible: 10m, 20m and 60m resolution. With a more sophis-

ticated tiling algorithm, resampled intermediate resolutions could be realized, further increasing loading speeds.
- Shader optimizations: Currently, the shader calculates all parameters (spread, masking values, mean, SD, ect.) with each pass, even if only one display parameter has changed. Although the performance impact is negligible in 60m and 20m resolution modes, running a shader pass on a full 10m scene takes approximately 200 milliseconds on a 2016 mid-range GPU (AMD RX480).

Apart from performance optimizations, a variety of new features could be implemented into the *Scene Explorer*:

- On-the-fly re-projections: Currently, only scenes in the same projection system can be visualized simultaneously. On-the-fly re-projections would allow to load adjacent tiles that are projected in different UTM zones with minimal impact on loading times.
- Vegetation Indices: With the use of additional shader programs, indices could be calculated in near-real time speed for visualized datasets.
- Data Generation: Scenes that have been altered by using the *Value Pincher* or masking modes could be stored on disc.
- Mosaicing: This would enable users to generate a temporal composite of multiple scenes by using different algorithms, allowing to generate cloud-free mosaics over arbitrary date-ranges.

As pointed out by users during the evaluation, the *Value Pincher* should be updated to include a "Reset" and "Invert" button. The "Reset" button should revert all settings to their respective defaults, while "Invert" would toggle the visibility of all pixels in the scene, according to their current state with respect to the thresholds set in the *Value Pincher*.

9. Summary & Outlook

Remote sensing satellites have proven to be valuable assets in the fields of climate research, urban planning, agricultural monitoring and many more fields. Efficient handling of this data is crucial, as it allows scientists and experts to spend more time on actually analyzing this data. Although there are many different geoinformation systems that allow for handling large quantities of data, there has been no specific framework for the handling of Sentinel-2 image data.

In this thesis, we presented the *Sentinel-2 Explorer* framework, a tool specifically designed for the handling of images acquired by this satellite. We started by providing an introduction to the field of remote sensing, the associated instruments and the uses that these satellites provide. We then looked at different visualization techniques that allow a meaningful representation of this data and established the basic requirements for our framework. In the Dataset chapter, we explained the differences between Level-1C and Level-2A products and how the latter can be obtained from the former. We established that the use of Level-2A data is beneficial as it provides Bottom-of-Atmosphere corrected data with classifications provided in the form of a separate 20m band. We then proceeded to describe the specific data-format and which bands are found in a dataset. The next chapter, Design, started by defining the target user group and the associated core functionality, that is, the querying and downloading of data and the visual inspection of retrieved datasets. In succession, functional and non-functional requirements were defined to form a contract which the implementation has to fulfill. By providing a detailed description of how the framework will operated, we laid the groundwork for the actual implementation. Unity3D was chosen as a foundation as it provides the ability to use a computer's GPU and is capable of handling large amounts of data. The use of shader programs allow for complex massive parallel computation which proved to be very efficient during evaluation.

The implementation chapter described in detail how the established design was realized. We explained which features have been implemented and how each component interacts with the system. We further described in detail, how the complex loading of large datasets operates, and introduced a tiling system that only loads portions of the image that are actually visible to the user. In addition, datasets are loaded in the lowest sufficient resolution to improve efficiency.
In addition to the outlined improvements that were requested by the users, we provided a list of enhancements and new features in the previous chapter. The use of shaders could further be tapped by utilizing them for the calculation of vegetation indices. A possible use case could be the generation of mosaics, by creating a temporal composite of multiple datasets.

The evaluation of the framework showed that the presented system is indeed capable of handling large quantities of image data and can serve as a valuable asset when working with Sentinel-2 data. Overall, users where very satisfied with the system. Usability ratings were high, with an average rating of 81.46 on the SUS scale. We showed that Unity3D is a good foundation for implementing such frameworks, by providing many features out of the box, such as access to the GPU hardware via shader programs.

List of Figures

Appendix

Appendix A.

DVD Contents

The content of the provided DVD is as follows:

- Practical Part
  - The *Sentinel-2 Explorer* framework.
  - Collection of sample datasets (including the datasets used during the evaluation)
  - Configuration files and external libraries
- Theoretical Part
  - PDF version of this thesis
  - Evaluation questionnaires
  - Summary of the evaluation results

Appendix B.

Installation Guide

The directory containing the *Sentinel-2 Explorer* should be copied to a local drive on the PC. We strongly advise **against** using the framework directly on the DVD as loading speeds will suffer dramatically. The entire folder `Sentinel-2 Explorer` should be stored on a disk. In addition, the folder `ExampleData` should be copied as well.

A configuration file has to be edited in order to use the framework:

```
[local_path]/Sentinel2Explorer_Data/StreamingAssets/config.xml
```

The `DataDirectory` tag should point to a directory where imported scenes are stored. This should initially be an empty directory. The `DownloadDirectory` tag should point to a directory where downloaded scenes are stored, i.e. the copied `ExampleData` directory.

To use the *Earth Explorer*, a free registration on the Copernicus Server[1] is necessary. The credentials have to be provided to the framework via the `SciHubUsername` and `SciHubPassword` tag, respectively.

[1]https://scihub.copernicus.eu/dhus/#/self-registration

Bibliography

Aschbacher, Josef and Maria Pilar Milagro-Pérez (2012). "The European Earth monitoring (GMES) programme: Status and perspectives." In: *Remote Sensing of Environment* 120, pp. 3–8. DOI: 10.1016/j.rse.2011.08.028. URL: https://doi.org/10.1016/j.rse.2011.08.028 (cit. on pp. 27, 28).

Blau, Patrick (2016). *WorldView-4 Satellite*. URL: http://spaceflight101.com/worldview-4/ (cit. on p. 7).

Blender (2019). *Warp*. URL: https://docs.blender.org/manual/en/latest/modeling/meshes/editing/transform/warp.html (cit. on p. 89).

Brooke, John et al. (1996). "SUS-A quick and dirty usability scale." In: *Usability evaluation in industry* 189.194, pp. 4–7 (cit. on pp. 95, 101).

Colwell, Robert N (1985). *Manual of remote sensing*. American Society for Photogrammetry and Remote Sensing, Falls Church, VA (cit. on p. 7).

Craft, Brock and Paul Cairns (2005). "Beyond guidelines: what can we learn from the visual information seeking mantra?" In: *Ninth International Conference on Information Visualisation (IV'05)*. IEEE, pp. 110–118 (cit. on p. 17).

Dobashi, Yoshinori, Tsuyoshi Yamamoto, and Tomoyuki Nishita (2009). "Interactive and Realistic Visualization System for Earth-Scale Clouds." In: (cit. on pp. 21, 22).

Dodge, M., M. McDerby, and M. Turner (2008). *Geographic Visualization: Concepts, Tools and Applications*. Wiley. ISBN: 0470515112 (cit. on p. 23).

Doppioslash, Claudia (2017). *Physically Based Shader Development for Unity 2017: Develop Custom Lighting Systems*. Apress. ISBN: 1484233085. URL: https://www.amazon.com/Physically-Based-Shader-Development-Unity/dp/1484233085?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1484233085 (cit. on p. 91).

ESA (2016). *Sentinel-2 Naming Convention*. URL: https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi/naming-convention (cit. on p. 41).

ESA (2018a). *Copernicus Open Access Hub*. URL: https://scihub.copernicus.eu/userguide/WebHome (cit. on p. 24).

ESA (2018b). *Discover our satellites*. URL: https://www.copernicus.eu/en/about-copernicus/infrastructure/discover-our-satellites (cit. on p. 28).

ESA (2018c). *Level-1B Processing Overview*. URL: https://sentinels.copernicus.eu/web/sentinel/technical-guides/sentinel-2-msi/level-1b-processing (cit. on p. 35).

ESA (2018d). *Level-1C Algorithm*. URL: https://sentinels.copernicus.eu/web/sentinel/technical-guides/sentinel-2-msi/level-1c/algorithm (cit. on p. 35).

ESA (2018e). *Level-2A Algorithm*. URL: https://sentinels.copernicus.eu/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm (cit. on pp. 36–40).

ESA (2018f). *Overview*. URL: https://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Overview4 (cit. on p. 7).

ESA (2018g). *Sentinel-2 Data Format*. URL: https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi/data-formats (cit. on pp. 41, 42).

ESA (2019). *Sen2Cor*. URL: http://step.esa.int/main/third-party-plugins-2/sen2cor/ (cit. on p. 40).

Esa (2018a). *Facts and figures*. URL: https://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Sentinel-2/Facts_and_figures (cit. on pp. 1, 30).

Esa (2018b). *Facts and figures*. URL: http://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Sentinel-1/Facts_and_figures (cit. on p. 28).

Esa (2018c). *Facts and figures*. URL: http://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Sentinel-1/Facts_and_figures (cit. on p. 30).

Esa (2018d). *Facts and figures*. URL: https://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Sentinel-5P/Facts_and_figures (cit. on p. 30).

esri (2019). *ArcMap*. URL: http://desktop.arcgis.com/en/arcmap/latest/get-started/introduction/choosing-the-3d-display-environment.htm (cit. on p. 23).

Fletcher, Karen (2012a). *Sentinel-1 : ESA's radar observatory mission for GMES operational services*. Noordwijk, The Netherlands: ESA Communications. ISBN: 978-92-9221-418-0 (cit. on p. 28).

Fletcher, Karen (2012b). *Sentinel-2 : ESA's optical high-resolution mission for GMES operational services*. Noordwijk: ESA Communications. ISBN: 978-92-9221-419-7 (cit. on pp. 29, 30, 35).

Fletcher, Karen (2012c). *Sentinel-3 : ESA's global land and ocean mission for GMES operational services*. Noordwijk: ESA Communications. ISBN: 978-92-9221-420-3 (cit. on p. 30).

Fletcher, Karen (2012d). *Sentinel-5 Precursor: ESA's Atmospheric Chemistry and Pollution-Monitoring Mission*. Noordwijk: ESA Communications. ISBN: 978-92-9221-430-2 (cit. on p. 30).

Foresman, T. W. (2004). "DIGITAL EARTH VISUALIZATION AND WEB-INTERFACE CAPABILITIES UTILIZING 3-D GEOBROWSER TECHNOLOGY." In: (cit. on p. 22).

Freitag, Lori A and Raymond M Loy (1999). "Adaptive, multiresolution visualization of large data sets using a distributed memory octree." In: *SC'99: Proceedings of the 1999 ACM/IEEE Conference on Supercomputing*. IEEE, pp. 60–60 (cit. on pp. 11, 53).

GDAL/OGR contributors (2019). *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation. URL: https://gdal.org (cit. on p. 68).

Geology, Maribeth H. Price Assistant Professor of (2015). *Mastering ArcGIS (WCB Geography)*. McGraw-Hill Education. ISBN: 9780078095146. URL: https://www.amazon.com/Mastering-Geography-Maribeth-Assistant-Professor/dp/007809514X?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=007809514X (cit. on p. 23).

Google (2018). *How images are collected*. URL: https://support.google.com/earth/answer/6327779?hl=en (cit. on p. 1).

Graham, Steve (1999). *Remote Sensing*. URL: https://earthobservatory.nasa.gov/features/RemoteSensing/remote_08.php (cit. on p. 8).

Haber, RB and DA McNabb (1990). "Visualization idioms: A conceptual model for visualization systems." In: *Visualization in Scientific Computing*, pp. 74–93 (cit. on p. 10).

Hagolle, Olivier et al. (2015). "A Multi-Temporal and Multi-Spectral Method to Estimate Aerosol Optical Thickness over Land, for the Atmospheric Correction of FormoSat-2, LandSat, VENS and Sentinel-2 Images." In: *Remote Sensing* 7.3, pp. 2668–2691. ISSN: 2072-4292. DOI: 10.3390/rs70302668. URL: http://www.mdpi.com/2072-4292/7/3/2668 (cit. on p. 21).

Hall, Dorothy K., George A. Riggs, and Vincent V. Salomonson (1995). "Development of methods for mapping global snow cover using moderate resolution imaging spectroradiometer data." In: *Remote Sensing of Environment* 54.2, pp. 127–140. ISSN: 0034-4257. DOI: https://doi.org/10.1016/0034-4257(95)00137-P. URL: http://www.sciencedirect.com/science/article/pii/003442579500137P (cit. on p. 33).

Heckbert, Paul and Michael Garland (1994). "Multiresolution modeling for fast rendering." In: *Graphics Interface*. Canadian Information Processing Society, pp. 43–43 (cit. on pp. 11, 53).

Inselberg, Alfred (2008). "Parallel coordinates: visualization, exploration and classification of high-dimensional data." In: *Handbook of Data Visualization*. Springer, pp. 643–680 (cit. on p. 14).

Inselberg, Alfred (2009). "Parallel Coordinates: Interactive Visualisation for High Dimensions." In: *Trends in Interactive Visualization*. Springer, pp. 49–78 (cit. on pp. 15, 16).

Irons, James R (2018). *Landsat 1*. URL: https://landsat.gsfc.nasa.gov/landsat-1/ (cit. on p. 6).

Jackson, Ray D. and Alfredo R. Huete (1991). "Interpreting vegetation indices." In: *Preventive Veterinary Medicine* 11.3, pp. 185–200. ISSN: 0167-5877. DOI: https://doi.org/10.1016/S0167-5877(05)80004-2. URL: http://www.sciencedirect.com/science/article/pii/S0167587705800042 (cit. on p. 19).

Jensen, R. John (2013). *Remote Sensing of the Environment: Pearson New International Edition: An Earth Resource Perspective*. Pearson Education Limited. ISBN: 1292021705. URL: https://www.amazon.com/Remote-Sensing-Environment-International-Perspective/dp/1292021705?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=

xm2&camp=2025&creative=165953&creativeASIN=1292021705 (cit. on pp. 7, 31–34).

Kay, Robin H and Sharon Loverock (2008). "Assessing emotions related to learning new software: The computer emotion scale." In: *Computers in Human Behavior* 24.4, pp. 1605–1623 (cit. on pp. 95, 102).

Koelsch, Geroge (2016). *Requirements Writing for System Engineering -*. New York: Apress. ISBN: 978-1-484-22099-3 (cit. on p. 45).

Kohonen, Teuvo (1995). *Self-organizing maps*. Berlin New York: Springer. ISBN: 3540586008 (cit. on p. 39).

Liang, Xiaohui and Chunqiang Yuan (2016). "Derivation of 3D Cloud Animation from Geostationary Satellite Images." In: *Multimedia Tools Appl.* 75.14, pp. 8217–8237. ISSN: 1380-7501. DOI: 10.1007/s11042-015-2738-7. URL: http://dx.doi.org/10.1007/s11042-015-2738-7 (cit. on p. 22).

Likert, Rensis (1932). "A technique for the measurement of attitudes." In: *Archives of psychology* (cit. on p. 98).

Liou, Kuo-Nan (1986). "Influence of cirrus clouds on weather and climate processes: A global perspective." In: *Monthly Weather Review* 114.6, pp. 1167–1199 (cit. on p. 39).

Loranty, Michael M. et al. (2018). "Vegetation Indices Do Not Capture Forest Cover Variation in Upland Siberian Larch Forests." In: *Remote Sensing* 10.11. ISSN: 2072-4292. DOI: 10.3390/rs10111686. URL: http://www.mdpi.com/2072-4292/10/11/1686 (cit. on p. 20).

Louis, J. et al. (2016). "Sentinel-2 Sen2Cor: L2A Processor for Users." In: *Living Planet Symposium*. Vol. 740. ESA Special Publication, p. 91 (cit. on p. 20).

Mayer, B. and A. Kylling (2005). "Technical note: The libRadtran software package for radiative transfer calculations - description and examples of use." In: *Atmospheric Chemistry and Physics* 5.7, pp. 1855–1877. DOI: 10.5194/acp-5-1855-2005. URL: https://doi.org/10.5194/acp-5-1855-2005 (cit. on p. 36).

Mullen, Tony (2011). *Mastering blender*. John Wiley & Sons (cit. on p. 86).

Müller, R. Dietmar et al. (2016). "The GPlates Portal: Cloud-Based Interactive 3D Visualization of Global Geophysical and Geological Data in a Web Browser." In: *PLOS ONE* 11.3. Ed. by Juan A. Añel, e0150883. DOI: 10.1371/journal.pone.0150883. URL: https://doi.org/10.1371/journal.pone.0150883 (cit. on p. 23).

Muller-Wilm, U. et al. (2013). "Sentinel-2 Level 2A Prototype Processor: Architecture, Algorithms And First Results." In: *ESA Living Planet Symposium*. Vol. 722. ESA Special Publication, p. 98 (cit. on pp. 20, 21).

Murray, Scott (2017). *Interactive data visualization for the web: an introduction to designing with.* " O'Reilly Media, Inc." (cit. on p. 17).

Nishita, Tomoyuki et al. (1993). "Display of the Earth Taking into Account Atmospheric Scattering." In: *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '93. Anaheim, CA: ACM, pp. 175–182. ISBN: 0-89791-601-8. DOI: `10.1145/166117.166140`. URL: `http://doi.acm.org/10.1145/166117.166140` (cit. on p. 22).

Paetsch, Frauke, Armin Eberlein, and Frank Maurer (2003). "Requirements engineering and agile software development." In: *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* IEEE, pp. 308–313 (cit. on p. 45).

Reichhardt, Tony (2006). *First Photo From Space*. URL: `https://www.airspacemag.com/space/the-first-photo-from-space-13721411/` (cit. on p. 5).

Rencz, Andrew N. (1999). *Manual of Remote Sensing, Vol. 3: Remote Sensing for the Earth Sciences*. Wiley. ISBN: 9780471294054. URL: `https://www.amazon.com/Manual-Remote-Sensing-Vol-Sciences/dp/0471294055?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0471294055` (cit. on p. 7).

Rhodes, Philip J, R Daniel Bergeron, and Ted M Sparr (2003). "A data model for adaptive multi-resolution scientific data." In: *Data Visualization*. Springer, pp. 257–272 (cit. on p. 11).

Rouse Jr, J_W et al. (1974). "Monitoring vegetation systems in the Great Plains with ERTS." In: (cit. on p. 31).

Schowengerdt, Robert A. (2006). *Remote Sensing: Models and Methods for Image Processing*. Academic Press. ISBN: 0123694078. URL: `https://www.amazon.com/Remote-Sensing-Models-Methods-Processing/dp/0123694078?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0123694078` (cit. on pp. 8, 19).

Shneiderman, Ben (2003). "The eyes have it: A task by data type taxonomy for information visualizations." In: *The Craft of Information Visualization*. Elsevier, pp. 364–371 (cit. on p. 17).

Sinergise (2018a). *Sentinel Hub*. URL: `https://www.sentinel-hub.com/about` (cit. on pp. 18, 24, 25).

Sinergise (2018b). *Sentinel Hub*. URL: `https://www.sentinel-hub.com/develop/documentation/eo_products/Sentinel2EOproducts` (cit. on p. 24).

Sinergise (2018c). *Sentinel Hub*. URL: `https://www.sentinel-hub.com/explore/eobrowser` (cit. on p. 25).

Smith, Mike J. and Chris D. Clark (2005). "Methods for the visualization of digital elevation models for landform mapping." In: *Earth Surface Processes and Landforms* 30.7, pp. 885–900. DOI: `10.1002/esp.1210`. URL: `https://doi.org/10.1002/esp.1210` (cit. on p. 19).

Stöckli, N. et al. (2005). *The Blue Marble Next Generation - A true color earth dataset including seasonal dynamics from MODIS*. URL: `https://visibleearth.nasa.gov/` (cit. on p. 67).

Suhet (2015). *SENTINEL-2 User Handbook* (cit. on pp. 12, 34–36, 55).

Tatem, Andrew, Scott Goetz, and Simon Hay (2008). "Fifty Years of Earth-observation Satellites." In: *American Scientist* 96.5, p. 390. DOI: `10.1511/2008.74.390`. URL: `https://doi.org/10.1511/2008.74.390` (cit. on pp. 6, 7).

Unity (2019). *Unity*. URL: `https://unity3d.com/unity?_ga=2.202707793.1510080073.1554197475-302470389.1554197475` (cit. on p. 50).

Ward, Matthew O., Georges Grinstein, and Daniel Keim (2015). *Interactive Data Visualization: Foundations, Techniques, and Applications, Second Edition - 360 Degree Business*. 2nd. Natick, MA, USA: A. K. Peters, Ltd. ISBN: 1482257378, 9781482257373 (cit. on pp. 13, 18, 66, 67).

Ware, Colin (2004). *Information Visualization: Perception for Design (Interactive Technologies)*. Morgan Kaufmann. ISBN: 1558608192. URL: `https://www.amazon.com/Information-Visualization-Perception-Interactive-Technologies/dp/1558608192?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1558608192` (cit. on pp. 9, 17).

Weiskopf, Daniel (2019). *gpu-based$_{i}$nteractive$_{v}$isualization$_{t}$echniques*. Springer. ISBN: 3540332626. URL: `https://www.xarg.org/ref/a/B002DOQKJQ/` (cit. on pp. 9–12).

Wolff, David (2018). *OpenGL 4 Shading Language Cookbook - Build high-quality, real-time 3D graphics with OpenGL 4.6, GLSL 4.6 and C++17, 3rd Edition*. 3rd. Birmingham: Packt Publishing Ltd. ISBN: 978-1-789-34066-2 (cit. on p. 53).

Yengoh, Genesis T et al. (2015). *Use of the Normalized Difference Vegetation Index (NDVI) to Assess Land Degradation at Multiple Scales: Current Status, Future Trends, and Practical Considerations*. Springer (cit. on pp. 31–33).

Yi, Ji Soo, Youn ah Kang, and John Stasko (2007). "Toward a deeper understanding of the role of interaction in information visualization." In: *IEEE transactions on visualization and computer graphics* 13.6, pp. 1224–1231 (cit. on p. 17).