UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Alan Kolk

Visualization of rooftop solar potential in smart cities

Bachelor's Thesis (9 ECTS)

Supervisor(s): Pelle Jakovits, PhD

Tartu 2023

Visualization of rooftop solar potential in smart cities

Abstract:

Visualizing 3D building rooftops with solar potential data on a map-based application gives a new look into the possible gains of using rooftop solar panels in a smart city. For local governments, cities, and individuals, visualizing the solar potential of buildings allows for a more informed evaluation of what the maximum gain from using solar panels on a roof would be. In this work, a solution for visualizing 3D city buildings in the 3DCityDB web map client was analysed and developed as a continuation of a previous thesis project. Scripts were created, that directly colour KML 3D building rooftops, based on the rooftops solar potential. The finished solution is made using open-source software and is capable of visualizing Estonian CityGML data attributed with solar potential data. Issues of the previous continuation project were fixed, and points of future developments are discussed. The finished work also serves as a basis for future 3D visualizations using CityGML data and the web application can be further improved to support more types of semantical 3D visualizations.

Keywords:

Solar Potential, Smart City, Visualization

CERCS:

P170 Computer science, numerical analysis, systems, control

Katuste päikeseenergia potentsiaalide visualiseerimine targas linnas

Lühikokkuvõte:

3D-hoonete katuste visualiseerimine päikeseenergia potentsiaali andmetega kaardipõhises rakenduses annab uue pilgu katusel asuvate päikesepaneelide kasutamise võimalikele eelistele targas linnas. Kohalike omavalitsuste, linnade ja üksikisikute jaoks võimaldab hoonete päikeseenergia potentsiaali visualiseerimine paremini hinnata, milline oleks maksimaalne kasu kui kasutada päikesepaneele linna majade katusel. Käesolevas töös analüüsiti ja töötati välja lahendus linna 3D-hoonete visualiseerimiseks, kasutades 3DCityDB veebi kaardirakendust, edasi arendades olemasolevat Magistritöö projekti. Töö käigus valimisid skriptid, mis värvivad päikesepotentsiaali põhjal KML andmefailides asuvaid 3D-hoonete katuseid. Valminud lahendus kasutab avatud lähtekoodiga tarkvaraga ning on võimeline visualiseerima Eesti CityGML andmeid, millele on omistatud päikesepotentsiaali andmed. Varasemalt olemasoleva projekti vead parandati ning toodi välja uue lahenduse võimalikud edasiarendused. Valminud töö on aluseks ka tulevastele CityGML andmeid kasutavatele 3D-visualisatsioonidele ning valminud veebirakendust saab edasiarendusena veelgi täiustada, et võimaldada muid semantilisi 3D-visualisatsioone.

Võtmesõnad:

Päikeseenergia Potentsiaal, Tark Linn, Visualiseerimine

CERCS:

P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Table of Contents

1 Introduction

Growing cities have introduced an increased need for alternative energy sources. To solve this increased demand for energy, it is necessary to research what solutions are available for city-wide energy infrastructures. Contrary to non-renewable energy resources, renewable energy resources provide such alternative solutions, which do not increase the damage caused to the environment by harmful emissions [1]. Therefore, to start using new city-wide energy solutions, it is imperative to first research and then plan the general use of renewable sources of energy, to get the most benefit from them and to guarantee optimal growth possibilities for future city developments. With the modern and more affordable solar panels, it is possible to redefine the use of a city building rooftop. Implementing solar panels on rooftops provides energy from a relatively unused area and large area, which can be harnessed as a potential energy source for smart cities.

Visualizing 3D rooftop solar potential data on a map-based application gives a new look into the planning of rooftop solar panels. When coordinates and geometrical building data is visualized with its solar potential data, it is easier to spot patterns in the different city areas. Geographical solar data, which in this work comes as a large text-based file, can be better understood when visualized on a map along with relevant metadata. The visual representation with colour-coding allows the user to grasp relevant information more easily and is visually more appealing.

For local governments and cities visualizing the solar potential of buildings allows the parties to better evaluate what the maximum gain from constructing solar panels on building rooftops would be. Such a visualization could also for example allow them to analyse what type of roofs, the shape and facing direction, would potentially generate the most power if solar panels were to be installed.

The aim of this work is to create an improved 3D visualization in the form of a web application, which shows solar potential data of rooftops in a smart city. This work is a continuation of a master's thesis done in 2022 in The University of Tartu that created a prototype [2] of a solar potential estimation pipeline and an initial visualization that finds the potential energy generated by buildings in a city, based on the location and shape of the building. The data is displayed on a 3D map where the buildings are interactable, whereby clicking on any specific building you can get the solar potential data attributed to the building.

The previous prototype was a requested work from the city of Tartu in collaboration with the University of Tartu. This work is also part of the collaboration between the city and the university, to further develop the 3D solar potential web application visualization.

An objective of this work is to cut the 3D building rooftop into parts, so instead of a single interactable building, we would have segments of the rooftop that all have their own solar potential data attributed to them. The reason for having differently coloured parts of the roof, is that in the previous prototype it was hard to get a good overview of how much one side of the roof contributed to the entire solar potential of the building. The previous implementation also coloured the entire building in one colour, which meant that walls were the same colour as the roof, whereas only the roof was planned to have solar panels on it. Having the roof and walls be the same colour in terms of solar potential generation, makes the visualization more confusing and does not allow the user to understand, which sides of the roof are the most beneficial to install solar panels on.

To colour different parts of the roof with different colours, it is most likely necessary to replace either the used Cesium JS [3] 3D map software with another open-sourced map software or use a different 3D object file type to implement segmented rooftops, which could then be coloured with different colours based on the solar potential that each segment gets. This visualization improvement of rooftop segments would allow any user of the web application to get more detailed information about the best location, where to put the solar panels on the rooftop in terms of solar potential. Other improvements that this work aims to achieve compared to the existing prototype come from reduced demand on hardware and bugfixes of the existing web application, which are analysed in more detail in section 3.2 and further compared with the results of this work in section 5.

The structure of this thesis is such that in section 3, related works are compared to the aims of this work and the existing prototype is analysed. In section 4, Estonian 3D data handling using 3DCityDB and creating visualization exports with the data are analysed and discussed. In section 4.3, gathering and using attribute data is explained. In section 5, visualization implementation is discussed, and examples of Tartu city visualization are shown. In section 6, the validation of the performance and usability of the work is done along with an overviewing analysis. In section 7, conclusions and issues are summarized.

2 Analysis of existing solutions

This section gives a brief overview of the studies and projects concerning city rooftop solar potential.

2.1 Related works of visualizing solar potential on buildings

In the following sub-sections, related works that have implemented solar potential visualization on building rooftops are analysed and discussed. The works are compared to what the aims of this thesis project are based on the positives and negatives of each related work.

2.1.1 Project Sunroof

Project Sunroof [4] is a solar potential web visualization that is similar to what this work aims to achieve. The project contains a 2D map, where buildings are coloured from violet to yellow based on the amount of sunlight on a rooftop. For each building on the map, it gives a brief analysis on the number of hours of usable sunlight per year and the area on a roof that is available for solar panels. It also calculates an estimation of the net savings over 20 years of using solar panels on a selected roof. Example of the application can be seen in figure 1.

Figure 1. Project Sunroof visualization of solar irradiance on a building.

What is missing in terms of this work is that clicking on a building requires opening a new webpage where the information is then displayed. The extra manual steps needed to get information on a building make the application slow to use. The main problem with Project Sunroof is that it currently does not have the data for Estonia, so a solution specifically for Estonia needs to be created. This work also aims to create a 3D visualization compared to Project Sunroof's 2D visualization.

2.1.2 Visualization of semantic 3D city models

In a journal article by Zhihang Yao *et al.* [5] a 3D visualization model was created using 3DCityDB [6]. The visualization used solar irradiation data, considering the shadows created by nearby buildings, the elevation of the buildings, and how much sunlight they get based on the location and weather data, to create textured 3D building models. The textures on the buildings are coloured green to red based on the amount of solar irradiance that the building faces get. An example of the texture-based visualization can be seen in figure 2.

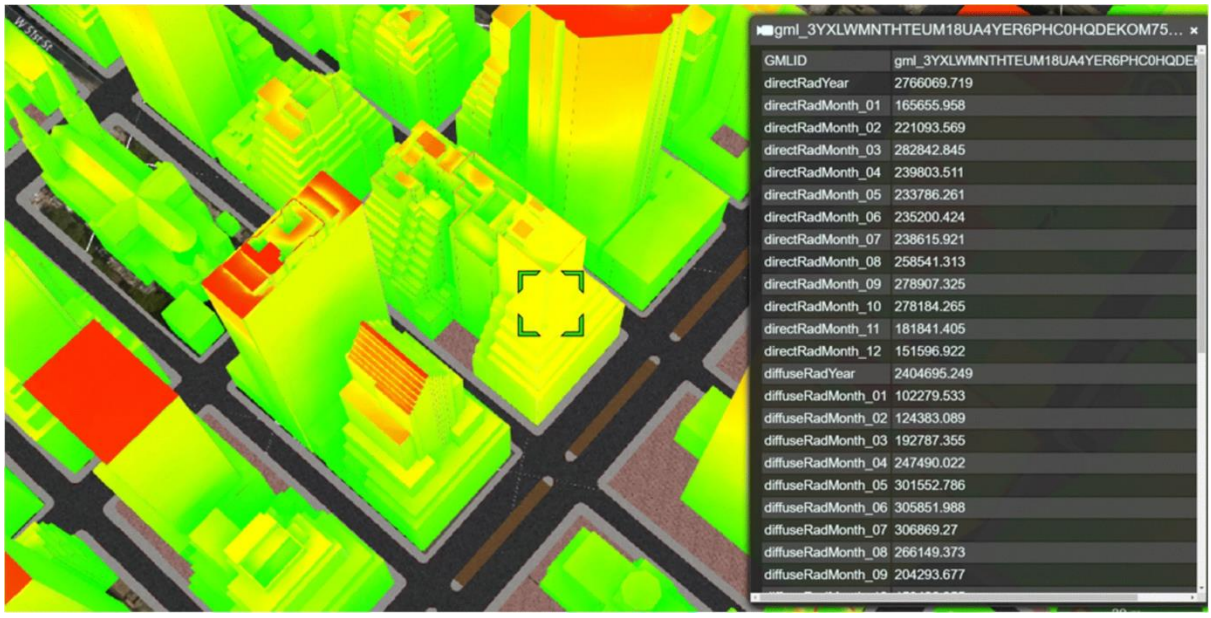

Figure 2. Visualization of solar irradiance heatmap on buildings [5, Fig. 18]

Creating a heatmap for the solar irradiance data allows for the visualization to be very precise. In the context of this work, it is however not possible to create such a precise heatmap for the data if the work depends on the current implementation of the pipeline. Because the pipeline from the prototype provides data for the main cardinal directions and for flat surfaces, it would be necessary to recreate the pipeline into a variant that is capable of outputting such data, which is needed for a solar potential heatmap. Since the aim of this work is not to remake the pipeline, then the colouring was done based on the four cardinal directions and "flat-surface" area provided in the current pipeline output data.

2.1.3 PV*SOL online

PV*SOL online [7] is a free web tool for the calculation of a solar panel systems output. It calculates the solar irradiation of a selected building on a world map and finds the best configuration and output for solar panels based on the type, amount, and the geometrical data, like the roof shape and various details. In terms of visualization, it offers a basic 2D map with no solar potential displayed on the map, which is a main component for this work. Because of PV*SOL online's basic 2D visualization and very manual use, it is considerably limited in the visualization, since the aim of this work is a 3D visualization, and when wanting to look at other buildings data, one would have to manually enter all the parameters again, which makes the tool slow to use.

2.2 Existing prototype

The objective of this work is to continue and improve a master's thesis project (Referred to as "prototype" throughout this thesis) by B. Romashchenko "Mapping Solar Potential of Tartu" [2], which was done at the University of Tartu in 2022. As a result of the thesis project, a solar potential data pipeline and a prototype web application were created for processing and visualizing 3D city spatial data to find the solar potential of city building rooftops. The source code is publicly available in a GitHub repository [8].

The prototype uses a custom-made pipeline for attributing the 3D spatial data with the solar potential of the buildings. To use the pipeline, CityGML data of a city is needed. The building geometry is analysed in the pipeline and solar potential data is then queried from PVGIS API [9], based on the roof geometry. The solar potential data is then added to the building in the CityGML data file and JSON files are created, which contain summarized data of the solar potential in the city and solar potential data for each roof polygon that was analysed by the pipeline. The solar potential data along with the CityGML 3D geometry is converted into Cesium 3D Tiles, which is then displayed in a web application, which visualizes the buildings of on a 3D map in a web browser. The web application uses Cesium JS for the 3D map and visualizing the Cesium 3D Tiles. The buildings themselves are colour-coded based on the amount of solar potential that they generate on average per year. An example image of the prototype can be seen in figure 3.

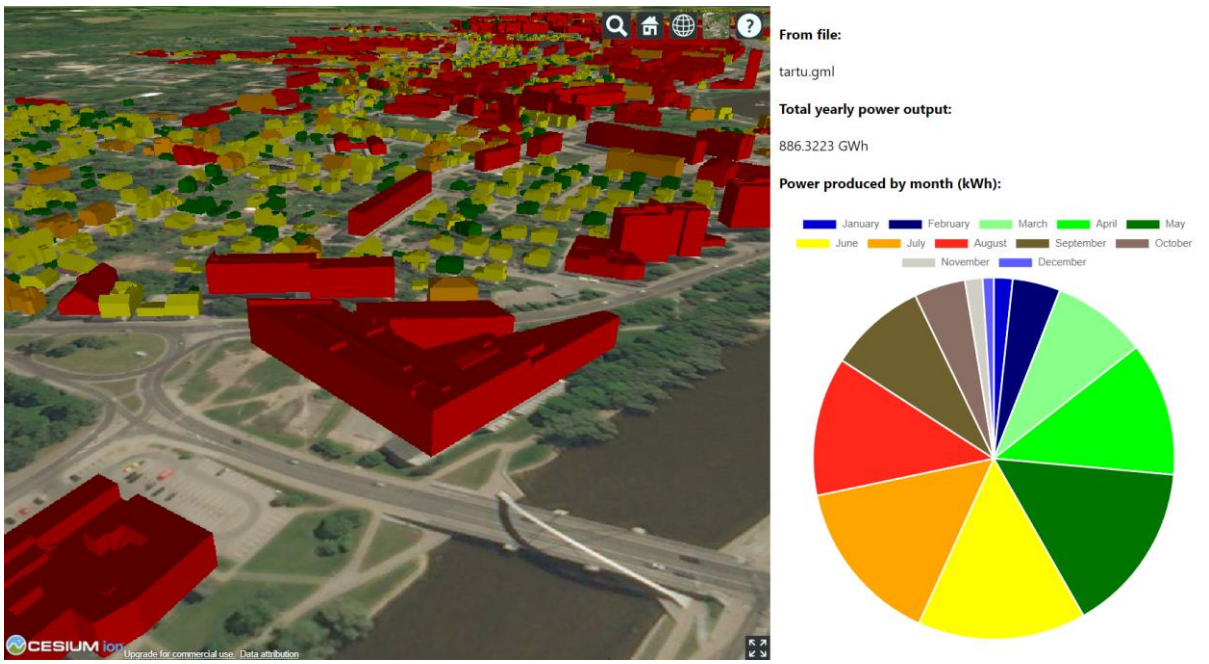

Figure 3. Screenshot taken from the existing prototype [2].

The requester of the existing prototype, the city of Tartu, wanted to take the prototype into public use, but there were problems with the web application, so further development was requested. So, the objectives of this work were to analyse the issues with the prototype and to design and develop an improved visualization solution that could fix usability problems with the prototype and reduce the demand on hardware resources along with an improved visualization.

In Romashchenko's thesis [2], several future improvements are proposed. One improvement mentioned was to implement the 3D objects in a way, where the building rooftops can be segmented and coloured differently to visualize the solar potential of a roof more accurately. This improvement is also one of the main objectives of this work. Currently in the prototype the buildings are all coloured with a single colour, which represents the solar potential of the rooftop. By having different parts of the roof coloured differently, it is possible to more accurately determine where the best place to install solar panels on the roof is and understand what areas of the roof the data processing pipeline has taken into consideration when calculating the solar potential of a building. Other improvements that could be done to the web application were various bugfixes and reducing the amount of hardware resources the application required. For example, the current web application requires the users system to have about 4 GB of RAM, a sufficient processor and network speed to download and smoothly display the 3D spatial data.

To make improvements possible, the 3D map software Cesium JS, that the prototype used for visualization, was decided to be replaced with another web map client, to fix the following issues. The prototype web application contained viewing problems, namely buildings disappeared when looking at a certain angle or moving in too close with the camera. The same thing happened if the camera was moved to the edge of the rendered city. Another problem with the implementation of the prototype was the use of resources when using the application. Since the application loads all the 3D data into memory during loading, then the application is slow to start. While testing the application, it took 9 seconds to load in the city of Tartu and depending on the hardware and network speed of the user, this time can be even longer. This was also a main request of the city of Tartu, as the slow loading time was something that they brought out in their feedback of the prototype application.

Colouring limitations of the rooftops came from the Cesium 3D Tiles datatype used in the prototype for the 3D objects. The Cesium 3D Tiles take a lot of resources to display, since they are all loaded into memory when using the application. The datatype was also the reason why the author of the prototype was not able to implement splitting up the rooftop surface colour into different coloured parts in the 3D web map.

3 Importing and converting 3D data

The prototype pipeline was implemented with the use of CityGML (City Geography Markup Language) for the 3D data objects. The data for this work is publicly provided by the Estonian Land Board [10]. 3D spatial data exists for the entire Estonia, but for development and testing only the Tartu city data was used. Since the pipeline was implemented for processing specifically CityGML files, and the solar potential data, that this work used for the visualization, came from the prototype's pipeline, then it was decided that the input data, used for representing the 3D objects in this work, would also be CityGML files. The only problem with CityGML was that there was no way to directly visualize it in a web application [11], so the file had to be converted to another 3D object file type.

3.1 Tools for data conversion and visualization

The Cesium JS web viewer, which was used to visualize the Cesium 3D Tiles in the prototype, was decided to be replaced because of its issues with performance and object viewing. A new solution had to be found for visualizing the 3D data in a web browser.

3.1.1 ArcGIS Online

ArcGIS Online [12] is a web-based mapping software that can be used to build interactive web maps. The software has many features and is capable of creating all kinds of semantic visualizations. Data can be imported into the software using various kinds of file types like GeoJSON, KML, and other common geospatial files. Another useful feature of ArcGIS is that it can be used for data analysis, which is important for solar potential data where finding the maximum benefits of the rooftop potentials is a key point. What makes ArcGIS unsuitable for this work is that the software costs and that is against the objective of developing the work using open-source software.

3.1.2 3DCityDB

3DCityDB [6] is an open-source 3D city database for storing, managing, and representing virtual 3D city models using a standard spatial relational database. The database can store CityGML and CityJSON data with its respective level of detail and can even store appearance data, like textures and colours. An Importer/Exporter [13] tool that comes with the 3D City Database can convert the imported data into different 3D visualization file formats, which can be used to convert the CityGML data that the work uses to another 3D file that can be visualized in web applications. It also contains a tiling feature for large datasets to make them more efficient to display and thus requiring less hardware power, which is also an objective of this work, as performance was an issue. It works best with the CityGML standard, so this aligns with the previously mentioned objective of continuing the use of CityGML in this work.

Because of the good compatibility with CityGML and the selection of tools that exist for converting the CityGML files to a different 3D visualization format, 3DCityDB was chosen for creating a new visualization in this work.

3.2 Estonian CityGML data import into 3DCityDB

Importing the CityGML data into 3D City Database required setting up a database with extensions. This was needed to use the Importer/Exporter tool, that 3DCityDB provides. To do a visualization export using the tool, it is required to import the data into a 3D City Database, as directly converting CityGML into a visualizable 3D format using the tool is not possible. This meant that it is also worth researching in the future if direct conversion is possible using another tool, for example FME “Convert CityGML to KML” [14], that is not usable in the context of this work as it costs and the aim was for this project to be open-source. This is something that could potentially reduce the steps that are needed to create a visualization for a new city, so a direct conversion solution could be considered for future developments.

The supported databases for the Importer/Exporter are PostgreSQL, Oracle, or PolarDB database. For the project PostgreSQL database was used with the administration platform pgAdmin [15] as the Author had the most experience using this database system out of the supported ones. The importing of the data into the database was a straight-forward process following the documentation of 3DCityDB [16] and the importing of the Estonian 3D data worked without needing any modifications. The tool correctly identified the coordinate system and the types of 3D objects in the dataset as buildings.

3.3 Available visualizable 3D data formats

3DCityDB allows exporting the data in the database as three different 3D object types. First is KML (Keyhole Markup Language) geometry file format, which is based on the XML (Extensible Markup Language) standard. This is usable in Cesium JS, which the prototype uses, and in a web map client [17] that comes with 3DCityDB. Secondly it is possible to export as COLLADA file format, which is also an XML-based file type for 3D assets. Finally, there is glTF, which is specifically for 3D scenes and models. The biggest difference between KML, COLLADA and glTF is that the latter two allow the use of textures in digital globe browsers. An example of a KML file can be seen in figure 4.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Written by 3D City Database Importer/Exporter, version "4.3.0" -->
<!-- Chair of Geoinformatics, Technical University of Munich -->
<kml xmlns:xal="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns="http://www.opengis.net/kml/2.2" xmlns:atom="http://www.w3.org/2005/Atom"
  <Document>
    <name>Tartu_Tile_5_10_geometry</name>
    <open>false</open>
    <Placemark>
      <name>Tile border</name>
      <LineString>
        <tessellate>true</tessellate>
        <coordinates>26.504345235583752,58.334129053654664 26.50652976329173,58.334129053654664 26.50652976329173,58.33530934148203 26.50
      </LineString>
    </Placemark>
    <Placemark id="KMLGeom_etak_6956437_hooned_lod2_RoofSurface">
      <name>etak_6956437_hooned_lod2_RoofSurface</name>
      <styleUrl>#RoofSurfaceNormal</styleUrl>
      <MultiGeometry>
        <Polygon>
          <altitudeMode>absolute</altitudeMode>
          <outerBoundaryIs>
            <LinearRing>
              <coordinates>26.5044668,58.3343237,2.54 26.5044668,58.3343237,2.54 26.5044146,58.3343419,2.0 26.5043343,58.3342785,2.45 26.
            </LinearRing>
          </outerBoundaryIs>
        </Polygon>
      </MultiGeometry>
    </Placemark>
```

Figure 4. Example of a KML file from the Estonian CityGML data.

By experimenting with the different file formats that could be exported from the Importer/Exporter tool, KML and glTF worked with the Estonian CityGML data without needing any modifications, when importing them into the 3DCityDB web map client. When importing the visualization as COLLADA into the web map client, the buildings were invisible. But through analysis into the options of colouring the COLLADA and glTF objects, it was determined that to achieve a visualization, where the rooftop parts are coloured differently, COLLADA and glTF were unsuitable for this work, as they could only be coloured using textures, but a way to colour individual polygons was needed. So, it was not necessary to fix the issue that was present when trying to use COLLADA as the 3D file type, as textures could not be created with the current pipeline implementation.

One change that needed to be done was an affine transformation on the CityGML coordinates data. That is because the Estonian data coordinates are in the format {latitude, longitude, height} but to visualize the data in the 3DCityDB web map client, the data coordinates need to be in the format {longitude, latitude, height}. For example, in a standard XYZ 3-dimensional grid, one would need to switch the places of X and Y in the data structure for the map client to correctly position the buildings. This change was very easy to do in 3DCityDB Import/Export tool, as there is an automated changing process for it in the preferences of the tool that does the affine transformation automatically when importing to the 3D spatial database.

Out of the KML and glTF file formats, KML was the only format that could be used for further development. glTF converted the buildings into single 3D objects just like the Cesium 3D tiles used in the prototype, so creating segmented roofs that could have different colours would not have been achievable without the use of textures. KML was the only export file format that kept the styling data of the polygons and geometry in the building data file itself.

The limitation that comes with KML is that the polygon styling is done in the file itself, which means that to colour the polygons individually, the modifications also need to be done in the file directly. Referencing the colours for the polygons from a different source is a topic that would need to be analysed further in the future. This meant, that to colour the polygons of a building based on the solar potential, scripts for directly changing the KML files were needed.

3.4 Linking the attribute data to the buildings

The CityGML files usually contain various attribute data about the objects in the data files. These attributes can be for example the address, building type, construction date, or any other custom value. To display these attributes with 3DCityDB, you must export them into tabular data like CSV or Microsoft Excel. The Importer/Exporter comes with a helpful plugin called Spreadsheet Generator Plugin, which takes a template file to filter and export user specified attribute values into a tabular data file. The exported tabular file can be uploaded to Google Sheets and then be used to show building data in 3DCityDB web map client [17]. The created tabular file can also be supplemented with any additional information that we want to display when selecting a building in the 3DCityDB web map client. An example of the tabular file containing attribute data can be seen in figure 5.

GMLID	ADDRESS_STREET	ADDRESS_HO	ADDRESS_CITY	BUILDING_MEASURED_HEIGHT	EXTERNAL_REFERENCE_NAME	SURFACE_GEOMETRY	LOD2_MUUTN	ETAK_MUUTI	TYYP	TYYP_TEKST	ALS_AASTA
etak_727906_hooned_lod2	Sepikoja tÃ¤nav	7	Tartu linn	7.79	104040745, 727906, EE00772848	12	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_730967_hooned_lod2	Kungla tÃ¤nav	6	Tartu linn	8.32	104043422, 730967, EE00775324	20	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_729069_hooned_lod2	Ã–¶biku tÃ¤nav	20	Tartu linn	4.31	729069, ME03027847	8	06/03/2021	24/06/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_710902_hooned_lod2	Peetri tÃ¤nav	53	Tartu linn	7.07	104014518, 710902, EE00748624	30	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_718550_hooned_lod2	Kopli tÃ¤nav	13	Tartu linn	8.37	104042819, 718550, EE00774751	36	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_720610_hooned_lod2	Pikk tÃ¤nav	100	Tartu linn	20.84	104017951, 720610, EE00751819	56	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_730820_hooned_lod2				2.92	730820	7	06/03/2021	12/09/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_7541478_hooned_lod2	Sepa tÃ¤nav	19	Tartu linn	18.01	120869462, 7541478, EE03714443	51	06/03/2021	07/01/2022	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_717302_hooned_lod2	Kroonuaia tÃ¤nav	21	Tartu linn	3.28	717302, ME03025111	7	06/03/2021	19/06/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_710384_hooned_lod2	Ujula tÃ¤nav	76	Tartu linn	9.41	104042047, 710384, EE00774037	26	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_716457_hooned_lod2	Sakala tÃ¤nav	2	Tartu linn	8.44	104041014, 716457, EE00773092	27	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_713730_hooned_lod2	Lembitu tÃ¤nav	1	Tartu linn	10.73	104038540, 713730, EE00770877	27	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_717514_hooned_lod2	Kullerkupu tÃ¤nav	9	Tartu linn	8.95	104016543, 717514, EE00750493	21	06/03/2021	12/07/2021	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_644404_hooned_lod2				3.36	644404	7	06/03/2021	10/05/2022	20	KÃµrval- vÃµi tootmishoone	2020
etak_714523_hooned_lod2	PÃµhja puiestee	33	Tartu linn	3.59	714523, ME03035915	11	06/03/2021	19/06/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_714571_hooned_lod2	Friedrich Reinhold	60	Tartu linn	12.07	104040757, 714571, ME01668753	106	06/03/2021	07/12/2021	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_710836_hooned_lod2	TÃ¤he tÃ¤nav	50a/3	Tartu linn	3.18	710836, ME03043421	7	06/03/2021	19/06/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_716993_hooned_lod2				3.72	716993	7	06/03/2021	12/09/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_7212168_hooned_lod2				3.05	7212168	7	06/03/2021	18/09/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_719786_hooned_lod2	Rebase tÃ¤nav	16-Jan	Tartu linn	8.14	104017176, 719786, ME00751086	13	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_736779_hooned_lod2	Hipodroomi tÃ¤nav	3c	Tartu linn	5.57	120282644, 736779, EE00994444	11	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_644825_hooned_lod2	Filosoofi tÃ¤nav	12	Tartu linn	7.06	104013581, 644825, EE00747777	18	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_6502671_hooned_lod2				2.87	6502671	9	06/03/2021	10/09/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_6966028_hooned_lod2	Raba tee	21	Tartu linn	4.13	6966028, ME03513851	8	06/03/2021	24/06/2019	20	KÃµrval- vÃµi tootmishoone	2019
etak_6555437_hooned_lod2	Anne tÃ¤nav	70h/11	Tartu linn	2.89	6555437, ME03271878	7	06/03/2021	19/06/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_712477_hooned_lod2	Riia tÃ¤nav	145	Tartu linn	8.68	104029856, 712477, EE00762994	34	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_706367_hooned_lod2			Tartu linn	6.62	104032029, 706367, ME00765026	9	06/03/2021	24/06/2019	20	KÃµrval- vÃµi tootmishoone	2019
etak_715644_hooned_lod2	Piiri tÃ¤nav	6	Tartu linn	3.27	715644, ME03027738	8	06/03/2021	24/06/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_6555913_hooned_lod2	Akadeemia tÃ¤nav	1b/3	Tartu linn	3.56	6555913, ME03272251	7	06/03/2021	17/05/2020	20	KÃµrval- vÃµi tootmishoone	2020
etak_728190_hooned_lod2	RÃ¤Ã¤gu tÃ¤nav	13	Tartu linn	2.77	728190, ME03035787	9	06/03/2021	19/06/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_718430_hooned_lod2	Jaama tÃ¤nav	153	Tartu linn	8.48	104040139, 718430, EE00772293	22	06/03/2021	24/06/2019	10	Elu- vÃµi Ã¼hiskondlik hoone	2020
etak_6499069_hooned_lod2				3.1	6499069	7	06/03/2021	13/09/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_7106760_hooned_lod2				3.01	7106760	8	06/03/2021	13/09/2019	20	KÃµrval- vÃµi tootmishoone	2020
etak_702929_hooned_lod2	Kandi tee	12	Tartu linn	9.33	104049156, 702929, EE00780868	15	06/03/2021	07/04/2020	10	Elu- vÃµi Ã¼hiskondlik hoone	2020

Figure 5. Attribute data of buildings exported using Spreadsheet Generator Plugin.

4 Visualizing solar potential data

3D City Database comes with a web map client [17] developed by the creators of the database using Cesium JS. It can visualize the data that the Exporter generates from the CityGML data in a web client, which makes it a suitable tool for visualizing the exported data from the Importer/Exporter. It also allows basic interactions with the 3D objects, like highlighting and hiding selected objects. This was chosen to replace the Cesium JS web application that the prototype used, as the one made by 3DCityDB worked well with the 3D visualization data and contained more features that were already implemented.

A visualization export of the Tartu city data with default styling options selected in the Importer/Exporter tool along with linked thematic data can be seen in figure 6.

Figure 6. 3DCityDB KML geometry visualization of buildings in Tartu with linked attribute data.

The individual polygons, of the KML geometry objects can be manually coloured if the KML files are modified directly. When a visualization export is exported into KML geometry objects, then each building's roof surface and wall surface are in separate groups that group all the polygons which make up the surface.

4.1 Colouring roof polygons

Since it was decided that the KML files would be modified directly, a script was made for colouring all the building roof surfaces in the KML dataset. The colouring of the polygons was based on the yearly kW/h solar potential data that was generated for each polygon using the pipeline. This required linking different data files to get all the needed data for the colouring and identifying the buildings and the specific polygons of them. To colour the buildings, it was first needed to get the solar potential data for them. The pipeline from the prototype generates a JSON data file that contains solar potential data for each roof polygon that was deemed suitable for solar panels in the pipeline implementation. For Tartu city the output data is already generated and available in a public GitHub repository [18], made by Romashchenko. The Tartu city data file "**city-attributes.json"** was 52.8 MB in size and contained 79443 roof surface polygons.

In the roof data file "**city-attributes.json**" [18] each polygon of a building contained different attributes like, "area", "azimuth", "tilt", "orientation", coordinates of the polygon, and potential yearly and monthly kW/h data. A part of the JSON file can be seen in figure 7.


```
{"7106760": {"roofs": [{"id": 1, "area": 26.198232537613297, "azimuth": -96.83710029141791, "tilt": 4.921, "orientation":
"none", "points_epsg_3301": [[6470477.35, 659463.92, 2.4499999999999957], [6470483.97, 659463.34, 3.009999999999998],
[6470483.62, 659459.43, 2.9399999999999977], [6470478.32, 659459.89, 2.489999999999995], [6470477.0, 659460.01,
2.3799999999999955], [6470477.35, 659463.92, 2.4499999999999957]], "yearly_kwh": 5242.83, "monthly_average_kwh": 436.9,
"monthly_kwh": [120.03, 267.37, 488.21, 624.31, 754.57, 721.51, 722.32, 634.87, 477.82, 265.07, 97.54, 69.2], "total_loss":
-10.33}], "total_roof_area": 26.2, "lat": 58.34570776117792, "lon": 26.723821908190466}, "728449": {"roofs": [{"id": 2,
"area": 34.22197663033071, "azimuth": 94.49312776951285, "tilt": 7.901, "orientation": "none", "points_epsg_3301":
[[6471469.5, 659649.43, 2.8699999999999974], [6471473.52, 659649.08, 2.3100000000000023], [6471472.79, 659640.71,
2.3200000000000003], [6471468.77, 659641.06, 2.8699999999999974], [6471469.5, 659649.43, 2.8699999999999974]], "yearly_kwh":
6839.15, "monthly_average_kwh": 569.93, "monthly_kwh": [142.16, 345.88, 624.75, 821.54, 994.65, 945.24, 950.62, 829.34,
621.95, 350.34, 127.04, 85.65], "total_loss": -10.41}], "total_roof_area": 34.22, "lat": 58.35453891595716, "lon":
26.727599614908748}, "737756": {"roofs": [{"id": 3, "area": 23.30935124222668, "azimuth": -43.279135877201554, "tilt":
31.058, "orientation": "south", "points_epsg_3301": [[6471977.95, 663310.42, 2.6000000000000014], [6471982.38, 663314.66,
2.5700000000000003], [6471984.23, 663312.73, 4.18], [6471978.09, 663306.67, 4.18], [6471977.95, 663310.42,
2.6000000000000014]], "yearly_kwh": 4339.29, "monthly_average_kwh": 361.61, "monthly_kwh": [69, 181.99, 372.15, 518.12,
672.03, 656.47, 648.22, 543.84, 377.81, 195.39, 63.29, 40.99], "total_loss": -10.68}, {"id": 4, "area": 19.18320705653449,
"azimuth": 135.37570464518558, "tilt": 31.538, "orientation": "north", "points_epsg_3301": [[6471981.33, 663306.8,
2.8400000000000034], [6471978.09, 663306.67, 4.18], [6471984.23, 663312.73, 4.18], [6471985.8, 663311.09,
2.8599999999999994], [6471981.33, 663306.8, 2.8400000000000034]], "yearly_kwh": 2319.89, "monthly_average_kwh": 193.32,
```


Figure 7. "city-attributes.json" file containing data about roof surface polygons

The script for colouring the KML files also depended on a metadata JSON file that the visualization export generates if an option ("Record metadata about exported features in JSON file") in the preferences of the tool is ticked. From the metadata JSON file, the script gets information about which tiles to process. Because a lot of tiles in the generated tile-grid do not have any buildings on them, the read-write operations that the script must do can be reduced by skipping empty tiles, therefore making the script process time shorter.

A script named "**color_building_roofs.py**" was made using Python. The script can be found in the project repository, which can be found in Appendix I. Since a large amount of KML files needed to be processed, a KML parser was necessary. A Python package pyKML [19] was used to parse the KML documents. It uses lxml, an XML toolkit for Python, that vastly simplifies parsing XML files, which KML files are also based on. Python and pyKML were selected because of the easy handling of reading and writing files and existence of documented and easy to use XML and KML parsers. The script was run in Visual Studio Code using a desktop machine with a Ryzen 7 5800X processor, 16 GB of RAM and an SSD, the processing and overwriting the KML files of the whole Tartu city, which consists of around 20000 buildings, took about 1 minute.

The script algorithm can be described with the following steps:

1. look through all the tiles in the dataset;
2. process all the buildings in a single tile;
3. get the roof solar data from the city-attributes.json file;
4. for each polygon of the roof;
    a. match the polygon coordinates with the coordinates from city-attributes.json roof data;
        i. city-attributes.json coordinates are transformed from EPSG:3301 reference system to WGS:84, which the visualization export uses.
    b. if the coordinates match, colour the roof based on the potential;
    c. create a new placemark that contains a single processed polygon;
5. append the new polygon placemark to the KML
6. remove old roof surface placemark;
7. overwrite unmodified KML file;

Using the script, it is possible to colour the polygons of a roof with any given parameter, for example orientation, yearly kW/h, or monthly kW/h data. A test colouring of the entire Tartu dataset can be seen in figure 8. Each cardinal direction is represented with a different colour and the polygons that were ignored by the pipeline are coloured black.

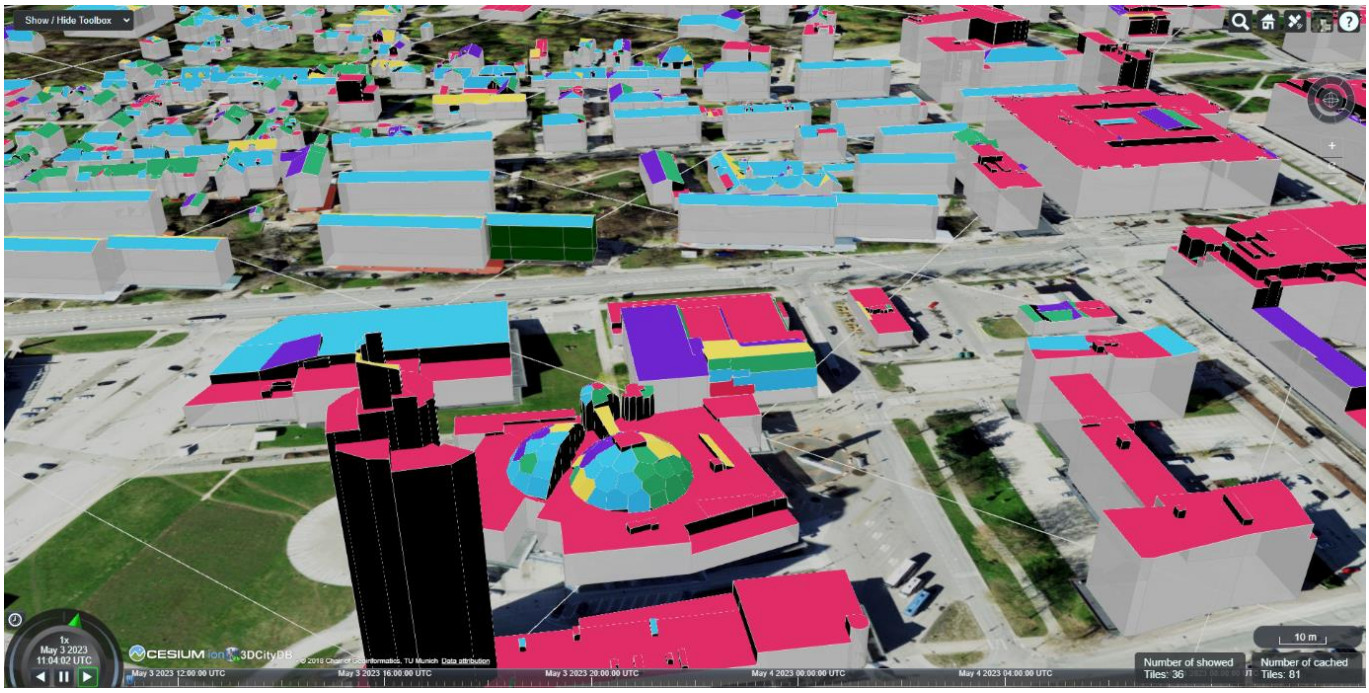

Figure 8. Tartu CityGML rooftop polygons coloured based on the orientation from the pipeline data.

4.2 Visualization based on yearly potential output

To create a distinguishable visualization of the solar potential data of rooftops a colour gradient was created. The colours of the gradient went from green to orange to purple, where green indicated little amounts of solar potential of a surface and purple indicated a large amount of solar potential.

A script "**retrieve_solar_potential_data.py**" was created that contains a function "**describe_yearly_kwh()**". The function uses a Python library called Pandas to easily describe the yearly kW/h data of the roof polygons. All the yearly potential data is gathered into a Pandas dataframe and then the dataframe is described with many different metrics that can be seen in table 1. The values in table 1 all represent yearly kW/h values of roof surface polygons and there are 12 percentiles specified, to find an overall distribution of the values.

Table 1. Analysis of Tartu roof surface polygons yearly kW/h values.

Metrics	Values (yearly_kwh)
count	79443.00000
mean	11268.71109
std	35483.85916
min	92.70000
1%	415.10410
5%	678.53300
10%	997.11000
25%	2107.99000
35%	3044.10900
50%	4746.19000
65%	7105.89600
75%	9770.26000
85%	14482.24700
95%	37195.70100
99%	124549.53720
99.9%	365933.59664
max	4671885.84000

12 percentiles were selected because this made the gradient colours to be easily distinguishable from each other, as the ranges for colours were picked based on the percentiles, along with the min and max. It was decided that too many colours in the gradient would make it hard to distinguish between the colours if looking at an area that was tightly packed with buildings, the polygon colours would blend with each other if the colours had a smoother transition in the gradient, making roof surfaces harder to distinguish between based on the solar potential.

The result of colouring the roofs based on the potentials can be seen in figure 9, where polygons with very high levels of solar potential are visible in purple and polygons with low levels of solar potential are in green.

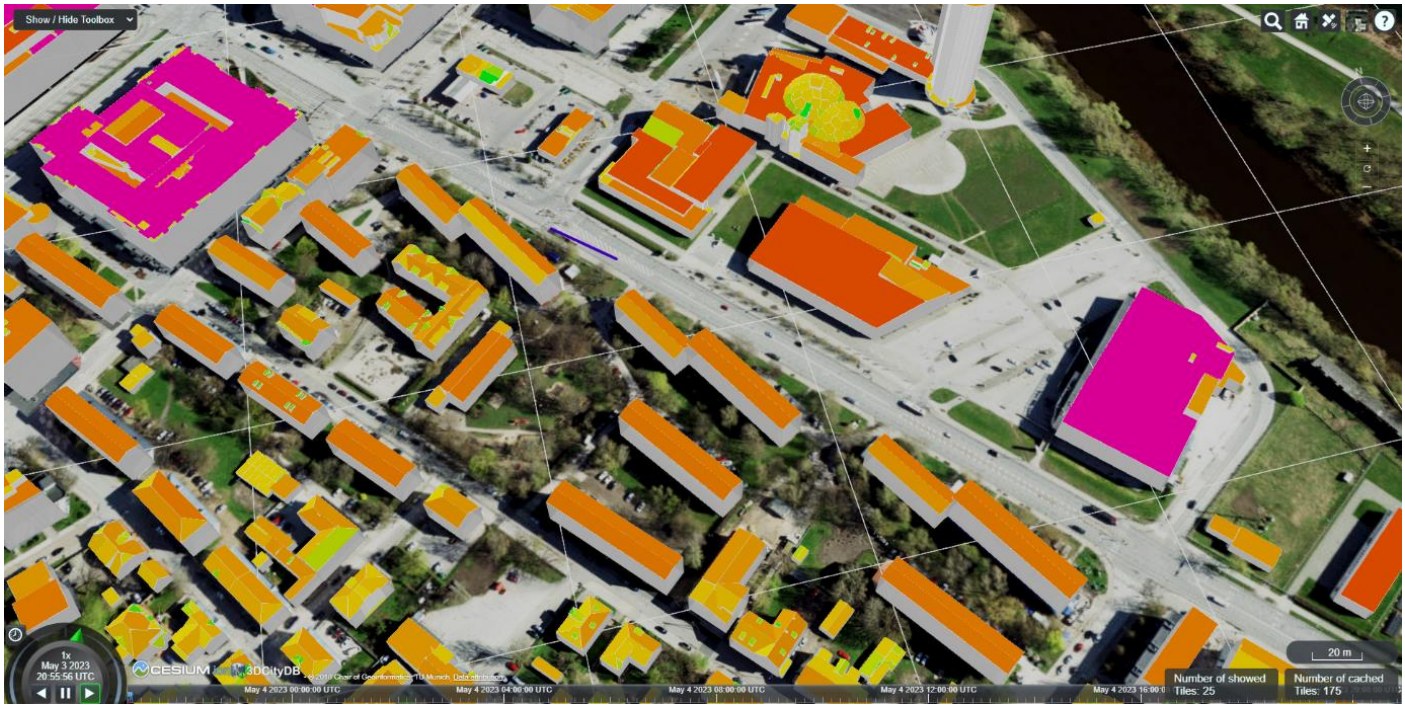

Figure 9. Visualization based on the solar potential of the roof surfaces.

The script also contains a function "**generate_csv()**" that adds the solar potential data into the tabular data file that was mentioned in section 4.3 of this work. This allows the web map client to display data about the building, when the user clicks on a specific building including all the potential power values to better understand the visualization and the indicated solar potential values. An example can be seen in figure 10.

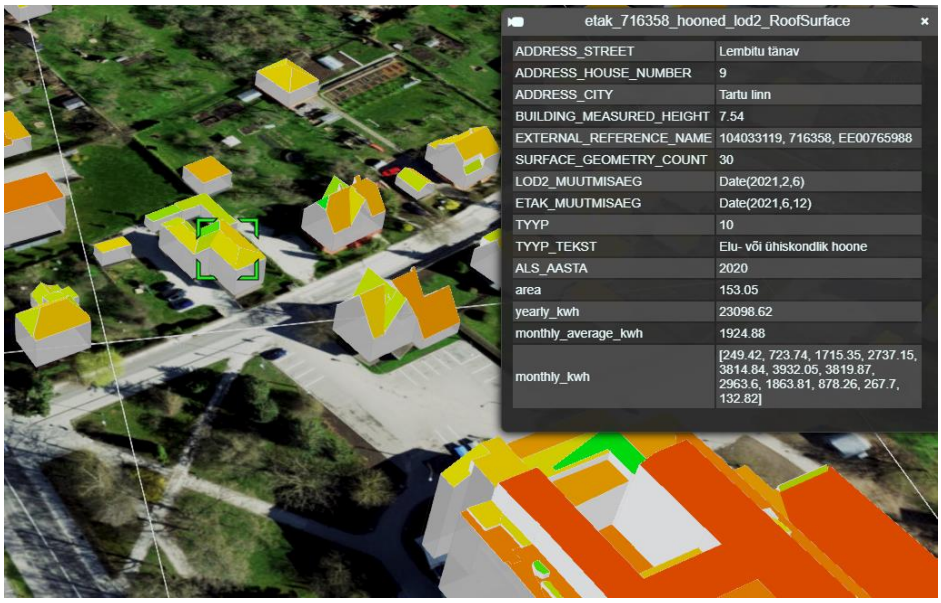

Figure 10. Attribute data of a building.

5 Validation and analysis

The aim to create an improved 3D visualization compared to the existing prototype [2], was successful. Compared to the prototype's visualization seen in section 3.2 Figure 3, where each building was a single 3D object with no visual distinction between the walls and roof, the new solution using 3DCityDB offers a much more visually informative solution.

It was understood that the issue with the previous prototype that did not allow colouring the roof parts differently lied with the data type used. By using 3DCityDB's Importer/Exporter tool to create a KML visualization export, it was possible to colour individual polygons that the CityGML geometry defines without needing to rely on textures. The KML visualization export allowed a solution, in the form of scripts, to be made, that made the colouring of individual polygons possible. This was something that the previous data type Cesium 3D Tiles was not capable of allowing and is a solution that was not observed in the related works, as the solar potential visualizations relied mainly on generated textures.

The prototype also had bugs that hindered the usage of the software. Under some viewing angles and when moving to the edges of the city that was rendered, the 3D objects would disappear. Clicking buildings would sometimes not work and the user had to rotate the camera to be able to click on the building they wanted to view data about the building. These issues were fixed with the usage of the 3DCityDB web map client. Usage of the new web map client introduced new features, like highlighting building parts that the user is hovering over and allowing for hiding and selecting multiple buildings in the web map.

Performance between the new solution using 3DCityDB web map client and the prototype made with Cesium JS varies greatly. In local tests, the prototype was more responsive than the web map client. This is because the prototype loads all the data into memory, and this allows it to show different parts of the rendered city very quickly. The downside of the prototype reading the data into memory is that the initial loading time is long. In local tests, the application loaded for 9 seconds before the city of Tartu was shown. The long loading time is heavily dependent on the hardware of the computer that uses the application and since it must download large data files (about 128 MB) over the internet, if one were to use it from a server and not have the data locally, then it also depends on network speed.

Compared to the prototype the new application loads in very quickly since it utilises a tile-based system, but it is not as responsive as the prototype was, because it must load parts in as the user is moving around with the camera. But once the new web map client has loaded tiles in, it is as responsive as the prototype. What the tile-based system allows is that the application is configurable to accommodate weaker hardware by reducing the number of tiles that are concurrently shown and vice-versa more powerful hardware can utilise a faster loading of tiles and have more tiles shown at once. This makes the new solution more available for users since one does not have to rely on a powerful system to run the application. When testing the web application with a different number of tiles shown concurrently, then systems with 4 GB of RAM should be able to handle the application without mayor loss in performance.

The web map client that was used for this work is also built to allow the use of various project setup structures in the operation of the application. The data is easily separatable from the project files and can therefore be hosted on a file server and referenced using an URL. Thematic data, which contains the attribute info, is currently hosted on Google Drive as a Google Sheets file but can be moved to a database to implement a REST API structure for the project.

The steps that a user has to take to generate a new final visualization are briefly described in this paragraph. First the previous prototype pipeline has to be run with 3D data from the Estonian Land Board to get the solar potential data. Then the Estonian Land Board 3D data needs to be imported into the 3D City Database. Afterwards, a visualization export into KML geometry must be done and the scripts must be used on the generated export. The data can then be imported into the web application after going through the aforementioned steps that need to be done in a configured environment for generating a visualization. Finally, the main **server.js** file in the project, found in the project repository in appendix I, needs to be run using Node.js [20] and then users of the application can look at the final visualization in a web browser.

6 Conclusions

The aim to create an improved 3D visualization as stated in the introduction, was achieved. The new solution fixed the viewing issue, where buildings would disappear when looking under a certain angle or moving to the edges of the city. Clicking on buildings also worked without depending on the viewing angle, which was an issue with the previous prototype. Performance issues were solved with the use of a tile-based system that 3DCityDB offered.

The objective of colouring the parts of a rooftop differently based on the amount of solar potential that a particular side generates was also made possible with the use of 3DCityDB, used as a core basis for the work. The usage of a different 3D object file type called KML, was the main key in colouring the rooftops with the data obtained by the prototype's pipeline to achieve the desired visualization.

Referencing the colours of the rooftop surfaces from a different source was not achievable in this work, therefore scripts were made to directly edit the KML files generated by the 3DCityDB Importer/Exporter tool visualization export. The main colouring script "**color_building_roofs.py**" processed each tile that contained a building and coloured the rooftop polygons based on the solar potential data, which originated from the pipeline's output. The script "**retrieve_solar_potential_data.py**" was used to do data analysis on the yearly kW/h data of the rooftop polygons to define a colour gradient for visualization and all the solar potential data was added to the thematic data tabular file with the help of the script.

The direct modification of the KML files using the scripts made the visualization based on solar potential data possible using 3DCityDB web map client and a satisfactory result was achieved.

6.1 Issues with the developed solution

The web application is very configurable to accommodate hardware with varying power for the application to use. But finding the right configuration is a manual process that requires testing to find out how many tiles can be shown to the user at once depending on the hardware power that is available.

6.2 Possible future improvements

There are different aspects of the work that can be improved upon. For the visualization, an entirely new approach could be taken, where instead of using KML geometry for colouring the rooftop polygons, textures or a completely solution could be implemented to further improve the accuracy of the visualization.

Generating a KML export requires setting up a 3D City Database instance, which is a task only needed before the visualization export is generated. If the CityGML to KML conversion could be done directly without the use of 3D City Database, it would reduce the number of steps that are needed in creating a visualization.

The web application design could be improved to better connect with the semantical data that is being visualized. For the project as a whole, a REST API system could be set up where the files are stored in a file server, the web application is hosted in a different machine, and the data generation could be done in a third machine.

The KML building colouring is currently done in file, but the colours could potentially be referenced from a different source, this could allow for the colours to be adjusted without the need of running the script on the whole visualization dataset.

References

[1] A. Abu-Rayash and I. Dincer, “Development and analysis of an integrated solar energy system for smart cities,” *Sustainable Energy Technologies and Assessments,* vol. 46, aug 2021. DOI: 10.1016/j.seta.2021.101170

[2] B. Romashchenko, *Mapping Solar Potential of Tartu,* Tartu: Institute of Computer Science, 2022. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=75190&year=2022.

[3] Cesium, “Cesium JS,” [Online]. Available: https://cesium.com/platform/cesiumjs/. [Accessed 26 February 2023].

[4] Google, “Project Sunroof,” [Online]. Available: https://sunroof.withgoogle.com/. [Accessed 12 March 2023].

[5] Z. Yao, C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubauer, T. Adolphi and T. H. Kolbe, “3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML,” *Open Geospatial Data, Software and Standards,* vol. 3, no. 1, p. 5, 2018. DOI: 10.1186/s40965-018-0046-7

[6] 3DCityDB, “The CityGML Database 3D City DB,” [Online]. Available: https://www.3dcitydb.org/3dcitydb/. [Accessed 26 February 2023].

[7] V. S. GmbH, “PV*SOL online,” [Online]. Available: https://pvsol-online.valentin-software.com/#/. [Accessed 12 March 2023].

[8] B. Romashchenko, “GitHub - Mapping Solar Potential of Tartu,” [Online]. Available: https://github.com/boroma4/Mapping-Solar-Potential-of-Tartu. [Accessed 15 February 2023].

[9] P. G. I. System, “PVGIS web tool,” [Online]. Available: https://re.jrc.ec.europa.eu/pvg_tools/en/. [Accessed 7 May 2023].

[10] E. L. Board, “3D spatial data of Estonia,” [Online]. Available: https://geoportaal.maaamet.ee/eng/Download-3D-data-p837.html. [Accessed 27 February 2023].

[11] J. T. Santhanavanich, “Web-based 3D Data Visualization with CityGML City Models,” [Online]. Available: https://towardsdatascience.com/web-based-3d-data-visualization-with-ciytgml-city-models-f0796b37e9f5. [Accessed 8 May 2023].

[12] Esri, “ArcGIS Online,” [Online]. Available: https://www.esri.com/en-us/arcgis/products/arcgis-online/overview. [Accessed 8 May 2023].

[13] 3DCityDB, “Importer/Exporter,” [Online]. Available: https://www.3dcitydb.org/3dcitydb/d3dimpexp/. [Accessed 27 February 2023].

[14] S. Software, “Convert CityGML to KML,” [Online]. Available: https://engage.safe.com/convert/citygml/kml/. [Accessed 7 May 2023].

[15] pgAdmin, “pgAdmin PostgreSQL Tools,” [Online]. Available: https://www.pgadmin.org/. [Accessed 27 February 2023].

[16] 3DCityDB, “3DCityDB software documentation,” [Online]. Available: https://www.3dcitydb.org/3dcitydb/documentation/. [Accessed 2023 February 26].

[17] 3DCityDB, “3dweblient,” [Online]. Available: https://www.3dcitydb.org/3dcitydb/3dwebclient/. [Accessed 26 February 2023].

[18] B. Romashchenko, “Solar-potential-output-example,” [Online]. Available: https://github.com/boroma4/solar-potential-output-example. [Accessed 22 March 2023].

[19] T. Erickson, “Python - pyKML,” [Online]. Available: https://pypi.org/project/pykml/. [Accessed 20 April 2023].

[20] Node.js, “Node.js,” [Online]. Available: https://nodejs.org/en. [Accessed 5 May 2023].

Appendix

I. Code

The project is publicly available as a GitHub repository along with a tutorial for setting it up for being locally run at:

https://github.com/alankolk/3dcitydb-web-map-Solar-Visualization.

The generated data for Tartu city can be found in the following GitHub repository:

https://github.com/alankolk/Tartu_solar_visualization_data.

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Alan Kolk,

1. grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, my thesis

Visualization of rooftop solar potential in smart cities,

supervised by Pelle Jakovits, PhD.

2. I grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in points 1 and 2.

4. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Alan Kolk

09/05/2023