

Cartographic visualization in Unreal Engine

And procedural modelling

FSv ČVUT, K155

Geoharmonizer APP – Unreal - https://gitlab.com/geoharmonizer_inet/vr-ar-app-prototype

Cartographic visualization in Unreal Engine

This exercise follows the previous lecture, which showcased procedural modeling and procedural generation in GIS for the creation of cartographic visualization in GIS. The goal of this exercise is the recreation of said GIS visualization in UE using the same GIS data.

While using UE in combination with GIS is still quite uncommon, there have been major advances in creating realistic cartographic 3D visualizations. This exercise presents one of many ways that UE can be used to visualize a landscape using real 3D GIS data. This is important since the demand for more realistic visual visualizations is steadily increasing thanks to the advancements in computer hardware and graphical technologies, which are now more accessible and affordable.

This exercise focuses on two main topics. Firstly, the processing of raster data in UE, with a focus on the importing process, requirements for the imported rasters, and the possible pitfalls specific to UE regarding the use of rasters. Secondly, it focuses on procedural modeling and procedural generation of 3D content in UE. Both of those topics are further expanded to explain and showcase the use of plugins, assets (packs), and the basic functionality of UE to re-create the visualization of a landscape using real data.

Used Software: Unreal Engine, CityEngine, ArcGIS Pro

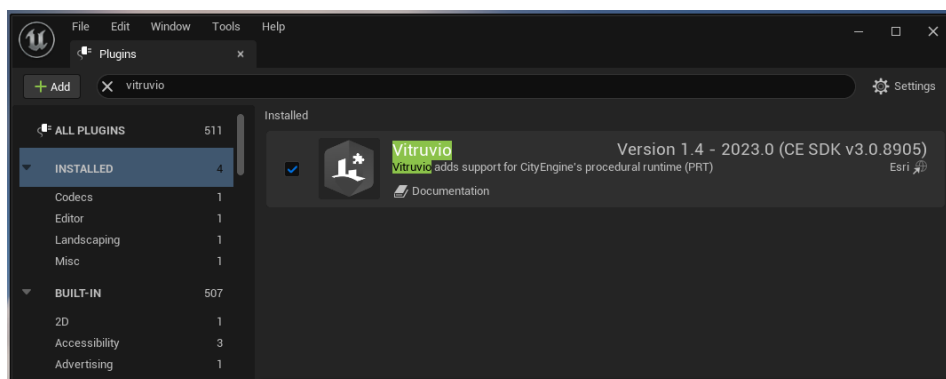
Used UE plugins: Datasmith Importer, Landscape Pro 2.0, Vitruvio, Water

Used data: DMT, land use data, RPK file(s)

Step 1—Preparation of UE project

The first step is a prelude to importing all the necessary data and then processing it further into the desired cartographic visualization. Finding the right solutions (plugins) for importing specific types of data and creating specific output, both of which are generally uncommon in UE, can be challenging. For this purpose, all the data and plugins were preselected to showcase UE's capabilities.

First, download and install the plugin [Vitruvio](#). Then, create a new empty UE project and activate the plugins Datasmith Importer, Water, and Vitruvio (Edit > Plugins).



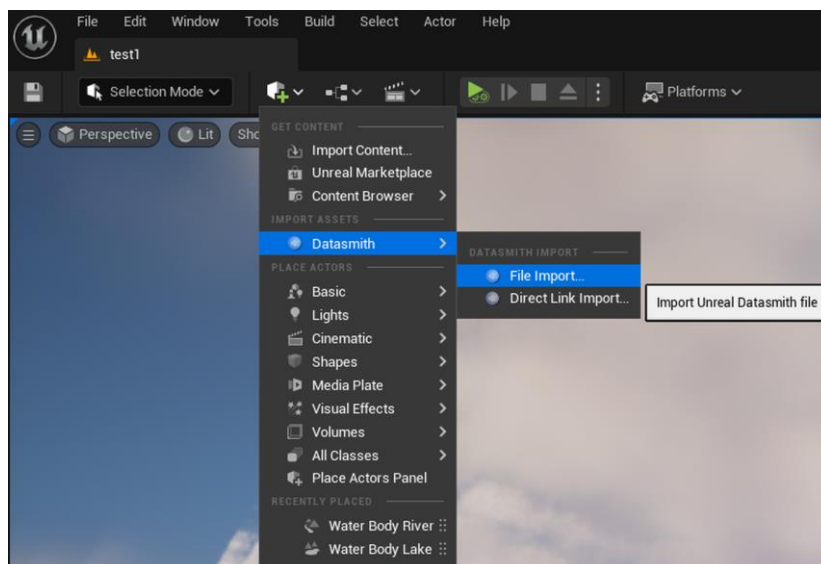
Lastly, log in to fab.com and add the library asset pack [Landscape Pro 2.0](#) to the project. Then, inside the Epic Games Launcher, add the asset pack to the project.

Using the abovementioned plugins, it will be possible to import GIS data using the Czech coordinate system S-JTSK (EPSG: 5514) and create a virtual landscape scene with all the necessary objects (buildings, water bodies, vegetation, etc.).

Step 2—Importing DMT

The standard way of importing DMT into UE is through *Landscape mode>Mange>New>Import from file*. However, this is problematic when trying to import DMT from GIS. In GIS, DMTs often use formats like geoTIFF, with many supplementary files containing all the information used by GIS that cannot be stored directly in the TIFF file. Furthermore, DMT often use different color depths based on the detail and precision of the raster. Those mentioned specifics (commonly used file formats, varying color depths, complementary files) are not supported in UE and cannot be imported the standard way. For this reason, the official dual plugin Datasmith Importer/Exporter is used in this exercise.

Datasmith is the official built-in extension of the UE. It imports data from other software, such as CAD or 3D modeling software. Datasmith has an exporter part, a plugin for external software to export files in the *.*udatasmith* file format. The exporter converts the exported content (models, textures, etc.) to formats compatible with those used by the UE. Those *.*udatasmith* files are then imported into the UE using the Datasmith Importer. While there is no exporter plugin for ArcGIS or any other GIS software, it is available in ESRI CityEngine as one of the default ways of exporting data. CityEngine is a 3D modeling software that supports the majority of GIS formats, and its main purpose is to procedurally model cities based on GIS data. For this reason, CityEngine is used as a middleman to import GIS data into the UE in this exercise.

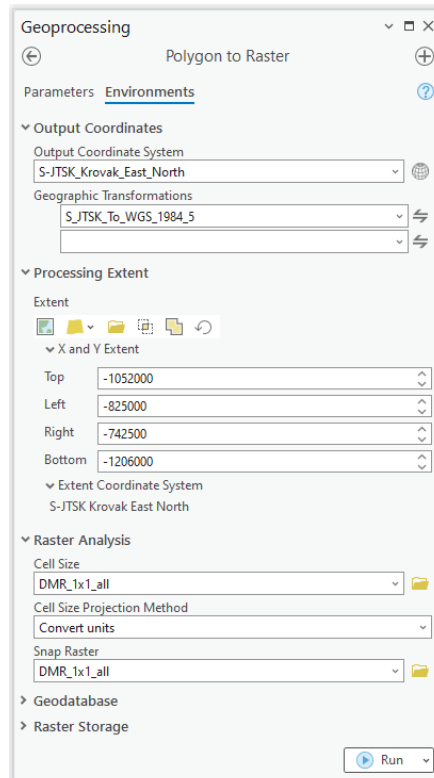


To do so, import all relevant data (in this case, the DMT) into the CE, select all the imported objects, and export them using the *Unreal Engine* option. The imported DMT will be automatically used to create a new landscape on top of which the whole visualization will be created.

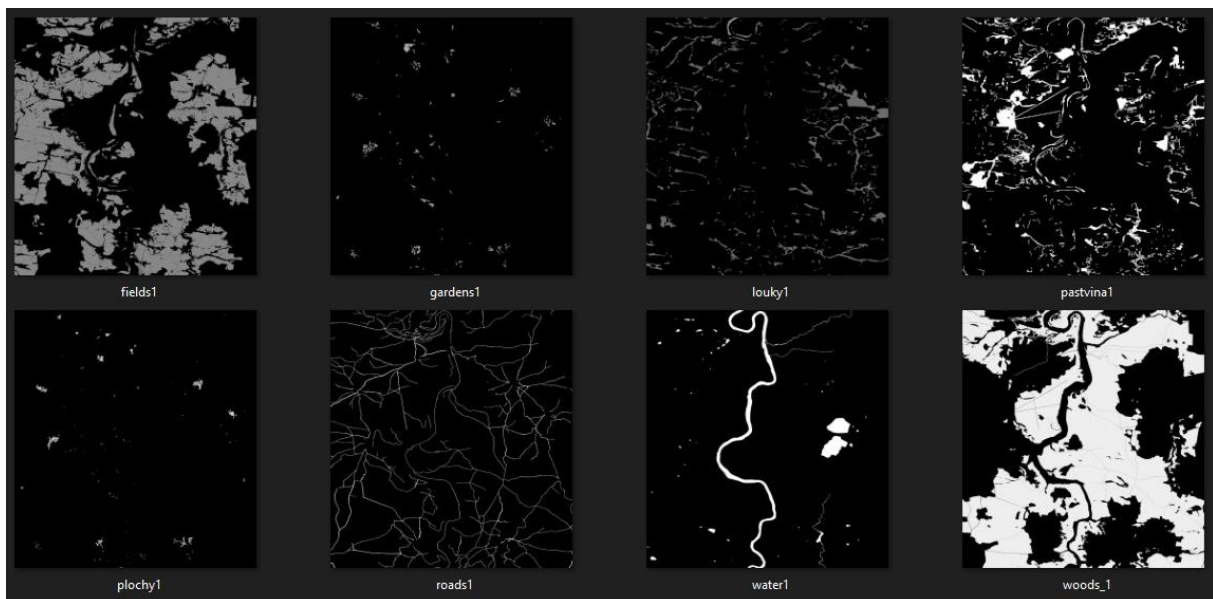
Step 3—Landscape Pro 2.0

In GIS, land use data is typically in vector form; however, to be usable in UE for visualization, it must be used as raster masks. This is because, in GIS, the vector polygons that contain symbology are projected onto the terrain, whereas UE works directly with the terrain. Raster masks tell UE where individual layers should be applied to the terrain. In a way, layers in UE can be seen as a symbology in GIS; however, they are way more complex due to the use of materials, which may contain way more functionality, such as the procedural generation of vegetation, which is later used in this exercise.

For this reason, individual vector layers need to be converted in ArcGIS Pro into rasters using the *Polygon to Raster* function. What needs to be highlighted is the use of *Snap Raster* option, which is mandatory because, without it, the output rasters would have different dimensions than the DTM, where the areas on the edges of the processing extent may be cut out (for example, because there may not be any forest in the border areas during the creation of land use raster of forests).

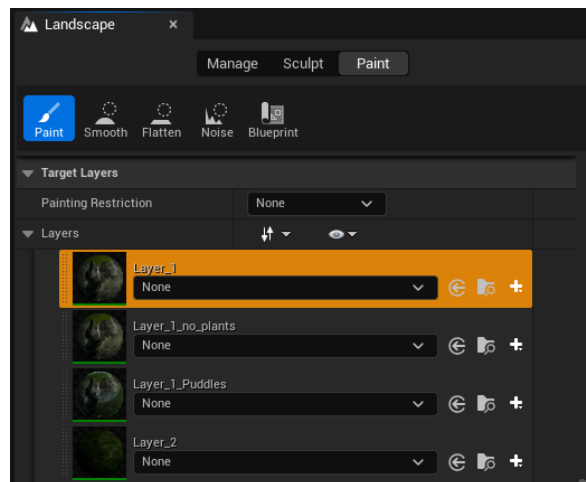


After that the raster can be exported for the use as Raster Mask. All rasters must cover the exact area as the DMT, have the same resolution, and be in a compatible file format (RAW or PNG).

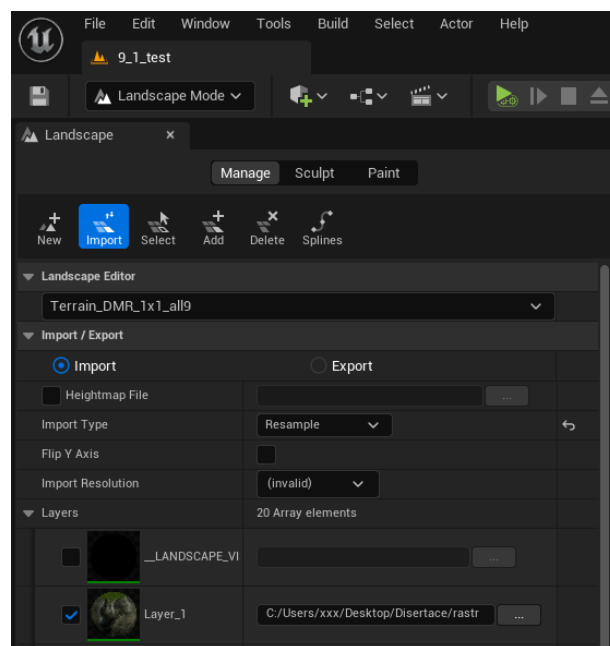


After the DMT is imported and raster masks are prepared, it is time to texture the terrain and fill the scene with content (3D models). While there are many ways to do that, this exercise uses a free asset pack containing all the required functionality for texturing and procedural foliage generation.

First, multiple layers must be created to apply multiple textures to the terrain. This has already been done within the landscape material contained in the asset pack Landscape Pro 2.0. To apply this landscape material, the imported DMT is selected, and the material file from the asset pack (Content>LandscapePro>Environment>Landscape) is dragged to the landscape material attribute of the DMT in its details tab. This will result in the DMT turning black. The next step is to create a Layer Info file for each layer of the DMT. This will automatically texture the black DMT. To create Layer Info, switch to Landscape Mode>Paint>Paint. All the terrain layers defined in the material layer should be visible in this tab. Next to each layer is a “+” icon, which allows the creation of a layer info file.

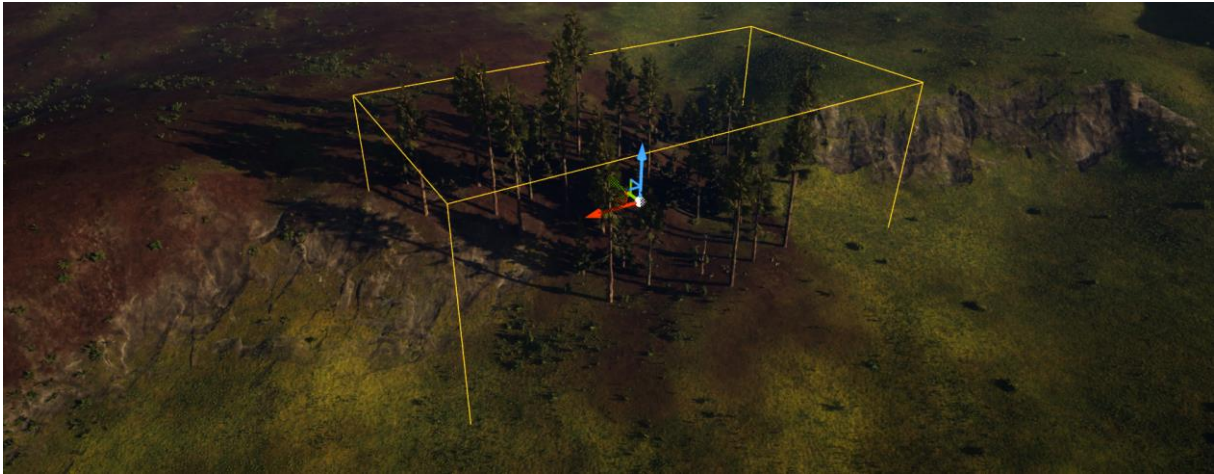


The last part of this step is to procedurally generate foliage. First, procedural foliage needs to be enabled in Edit>Editor Preferences>General>Experimental by toggling “Procedural Foliage” on. Then, each landscape layer needs to have its raster mask imported. What the raster mask does is tell the engine where individual textures (and appropriate foliage) should be applied. To import the raster mask to individual layers, in Landscape Mode>Manage>Import, the associated raster mask needs to be selected for each layer, and in the importing options, Import Type is set to “Resample.”



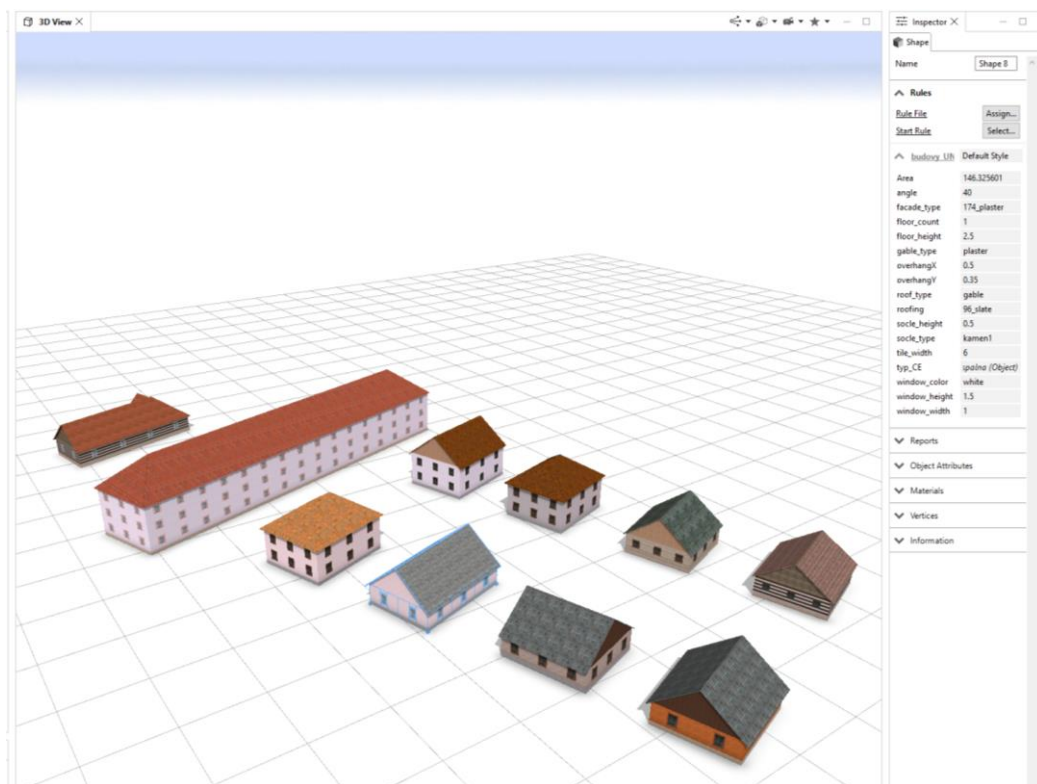
This will result in DMT being applied with textures according to the land use raster masks and automatically generated small (ground) foliage. To generate large foliage (trees), procedural spawners need to be placed into the scene. Those can be found in Content>LandscapePro>Procedural

Foliage>FoliageSpawner. After placing those spawners, their bounding box needs to be set (the area in which they will generate trees), and they need to be activated by clicking “Load Unloaded Areas” and “Resimulate” found in their details panel.



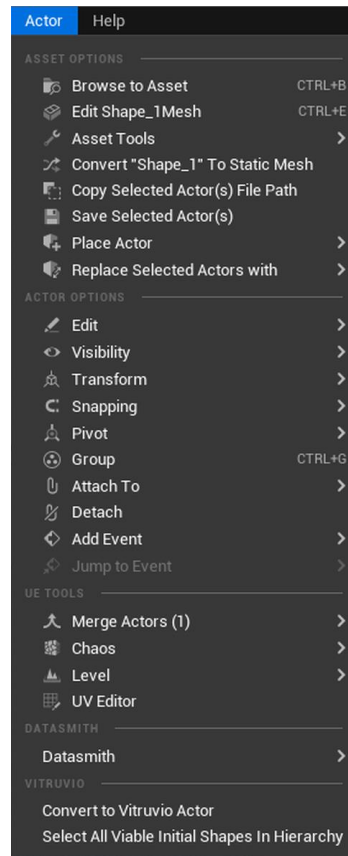
Step 4—Vitrivio

While the vegetation is created semi-automatically using advanced and complex materials, other 3D content has to be added or created manually. This is the case with 3D models of buildings, which can be imported directly or by using Datasmith. However, since CityEngine has already been used for importing GIS data, it is now also used for creating 3D models of buildings in this exercise.



For this, the plugin Vitrivio is used. Vitrivio is based on the CityEngine SDK, which provides functionality for the procedural modeling of buildings found in CityEngine. After the plugin is added to UE, new options for working with Actors will be added. This option allows for converting the Shape part

of an actor to VitruvioActor, which will allow the plugin to alter the shape according to the rule file from CityEngine and apply textures to it. As input, it uses the footprints of buildings and the RPK file generated in CityEngine. Building footprints is a vector dataset imported the same way as DMT (through CityEngine). The RPK file is a rule package file containing assets and rules for procedural building modeling. It is imported by dragging the RPK file into the Content Browser in UE.



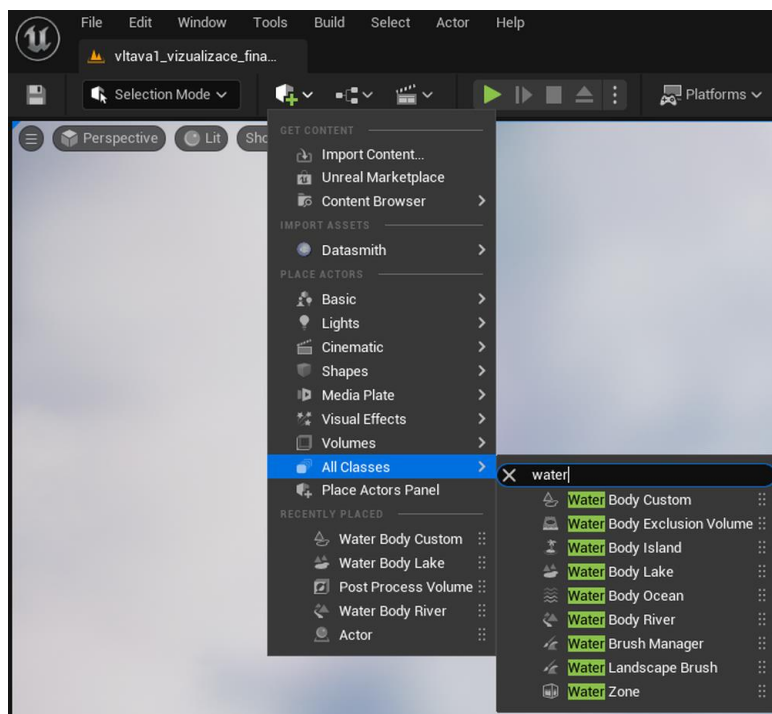
After the building footprints are imported, their shape components are selected using *Actor>Select all viable initial shapes in hierarchy*. Then *Actor>Convert to Vitruvio actor*, which will ask for RPK file. After selecting the RPK file, the 3D models of buildings are generated according to the selected RPK.



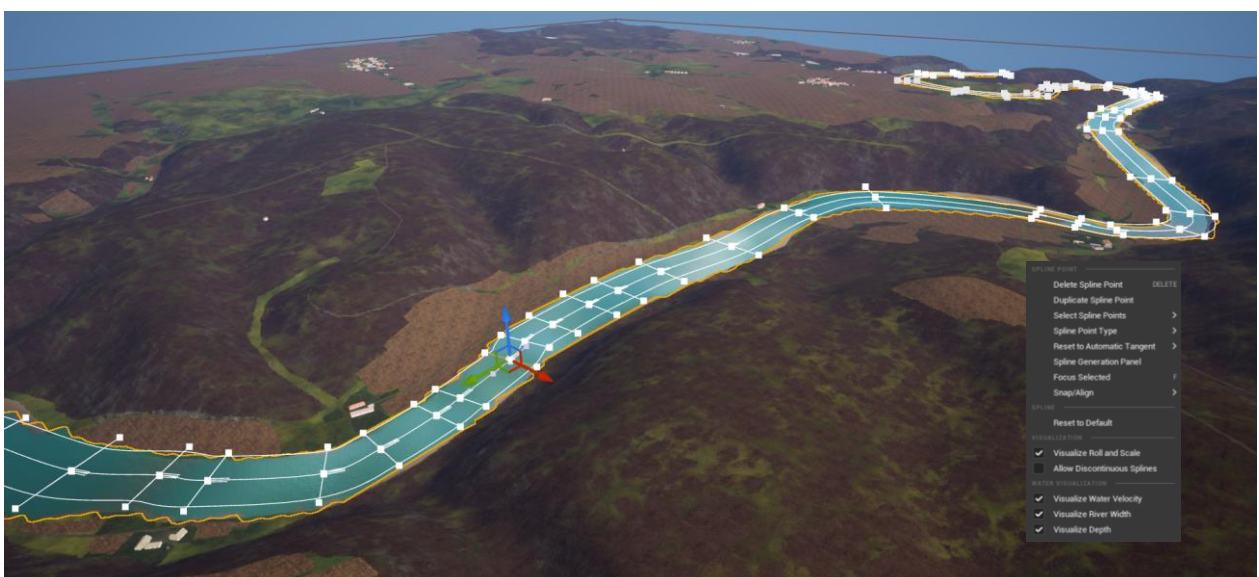
Step 5—Water

The last part of the visualization is the addition of water bodies. Unlike in GIS, where water is usually symbolized just by a (animated) texture, water in UE is a real physical object and, as such, is significantly more complex and harder to create and work with. As the goal of this exercise is just visualization, the easiest method of using the built-in plugin Water is used.

The built-in plugin, Water adds multiple new actors that can be used to create rivers, lakes, or general water surfaces, which can be found under the *Quickly add to the project>All classes>Water*.



As the names suggest, each water actor is defined differently and is used for a different type of water body. Each water actor is shaped by spline points, which can be added by right-clicking on the spline and adjusted using the actor details panel.





3D data in ArcGIS Pro

The aim of this exercise is to learn the basic skills of working with 3D data in ArcGIS Pro. The focus is mainly on data visualization, however, many outputs are preceded by basic data analysis. The analyses make possible, for example, the semi-automatic conversion from 2D to 3D - i.e. the addition of a third dimension, not only in the context of adding elevation coordinates only, but also the modelling of more complex 3D objects based on 2D geometry. The general concepts of working in 3D, the differences between a map and a scene, and the specifics of data containing the third dimension are explained at the beginning. It is seamlessly linked to existing GIS teaching - thus starting with common types of geometry (point, line, polygon). And through their Z-coordinate containing variants (pointZ, lineZ, polygonZ), it progresses to actual 3D geometry such as Multipatch FC and 3D Object FC. The output of the exercise is a 3D scene built from objects created by all the different methods discussed during the exercise (terrain alignment, attribute-driven extrusion, procedural symbology, etc.).

3D GIS data has many advantages over 2D, but its use is not appropriate in all cases. In some cases they can make the output more complex, which can be counterproductive in the end. For some outputs it is then appropriate to combine the two types of data. For example, using 3D data for analysis, which then results in a visualisation in a 2D map. In general, 3D data is very useful in the field of visualisation - i.e. the representation of real space. The template of such a space can be reality (e.g. a photogrammetric model of a city), a fictitious design (e.g. a study of the future state of a plot of land). These two possibilities can also be combined - i.e. supplementing the image of the real world with attribute information, which is one of the basic ideas of the so-called digital twin.

In ArcGIS Pro, there are 3 main types of space for viewing content: map, local scene and global scene. The map displays 2D space (but can display symbology based on height), and the scene displays 3D space. The local scene displays the space as a Cartesian coordinate system, so it is suitable for using data in the projected coordinate system. The global scene displays data on the surface of an ellipsoid (planet Earth), so it is suitable for geographic coordinates. For this reason, the global scene is limited to the WGS 84 coordinate system. Both scene types allow for on-the-fly transformation settings. The map, local, and global scene divisions also apply to the ArcGIS Online web interface.

Basic tools for viewing the scene in ArcGIS Pro:

Explore Tool - middle mouse button = orbit scene, left mouse button = pan on the surface, right mouse button = zoom

Select Tool - middle mouse button = pan on the surface, left mouse button = make a selection of features

C key temporarily overrides the active tool with the Explore Tool

Navigating the 3D web scene is different from the desktop version of ArcGIS.

Summary of data types related to 3D geometry

point, line, polygon no Z coordinates

pointZ, lineZ, polygonZ, multipatch contains Z coordinates

Scene Layer (I3S) 3D geometry with cache (for the web)

During analyses it is possible to affect the Z coordinate entry in the Environments settings - item "Z Values". When creating a new feature class, the "Z Values" option must be checked.

Manual editing of Z coordinates - Editing tool "Edit Vertices"

Bulk editing of Z coordinates:

- - Feature To 3D By Attribute (3D Analyst)
- - Adjust 3D Z (Data Management)

Add Z coordinates based on attribute or surface

- - Feature To 3D By Attribute (3D Analyst)
- - Update Feature Z (3D Analyst)
- - Interpolate Shape (3D Analyst)

Convert 3D symbology to Multipatch Feature Class

- - Layer 3D To Feature Class (3D Analyst)

Position geometry in scene (height)

The following settings can be selected in the scene layer properties (Elevation tab):

- - On the ground - the height from the surface marked as Ground will be used.
- - At an absolute height - either the attribute or directly the Z coordinates written in the element will be used
- - Relative to the ground - the Z coordinate will be added to the height of the surface
- - Relative to the scene - The Z coordinate is added to the height of the tallest object in the scene (applies to all 3D geometry)

For each of the options, you can also set an offset value.

The "Ground" layer can be set to "Vertical Exaggeration" (or "Z Factor"). All heights in the scene are multiplied by this value. This setting is used when you need to visually emphasize height differences. However, it has no effect on the analysis.

You can also set the color and transparency of the terrain ("Surface Color" option) or enable navigation below the surface ("Navigate Underground").

3D symbology

In addition to adding Z coordinates to feature vertices, 3D symbology can also be added. The specifics of this symbology depend on the geometry type.

Point geometry can be assigned basic 3D shapes or even an arbitrary 3D model to be placed at the position of each point.

Line geometry can be represented by profiles of different shapes and sizes - for example Tube, Strip, Wall or Rectangle.

Polygons can be set to extrude based on either an expression or an attribute.

Import of 3D formats

ArcGIS Pro has native support for reading the following non-GIS 3D formats: CAD drawing (DWG), BIM model (IFC, RVT)

The following formats can then be imported using the "Import 3D Objects (Data Management)" tool:

- - COLLADA (.dae)
- - Drawing (.dwg)
- - Autodesk Filmbox (.fbx)
- - Graphics Library Transmission (.glb)
- - JSON Graphics Library Transmission (.gltf)
- - Industry Foundation Class (.ifc)
- - Wavefront Object (.obj)
- - Universal Scene Description (.usdc)
- - Compressed Universal Scene Description (.usdz)

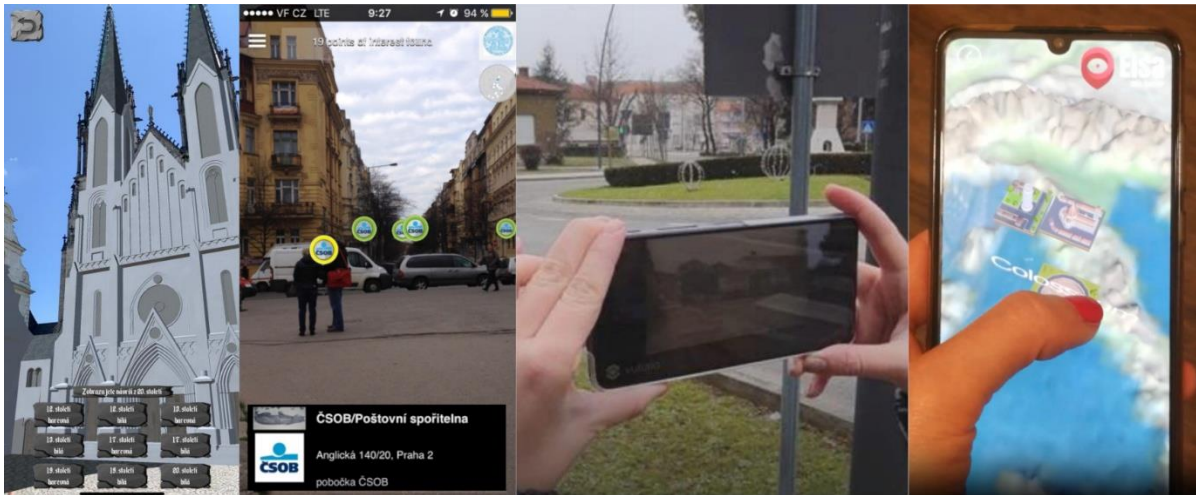
You can also export to DWG format using the "Export To CAD (Conversion)" tool.

Export to web scene

To export to an ArcGIS Online web scene, you must convert the 3D data to the I3S, or Scene Layer, format. This type of layer can then be uploaded to ArcGIS Online via ArcGIS Pro and then displayed in the 3D scene.

Spatial data visualisation in Augmented reality

Augmented reality (AR) is a real-time view of a physical world that has been enhanced by virtual information and is a part of extended reality (XR). The use of augmented reality is rapidly increasing due to the technological evolution, so it is being used not only in cartography, but also in many other domains. The main area of use is still entertainment, such as sports, video games or books. Nevertheless, AR is being present in many more serious domains, being culture (e.g. cultural heritage, arts, museums), medicine, education, interior design or industry. That is why it is important for students of civil engineering, geodesy or architecture to know the basics of this visualization method. Knowing how to visualise their work via AR can help them find an interesting job in the future.



Examples of spatial data visualisations in AR

AR can be viewed through a web page or in a standalone mobile application. Students are taught both of these approaches. AR in web can be viewed both on Android and iOS devices without no need for special modifications. On the other hand, standalone applications need to be deployed separately on every system. Also, one needs Mac for iOS app deployment. As for Android, one can build the application on either Mac or Windows. Web AR is better suited for smaller and less complex visualisations, while standalone application is better for visualising more complex objects with more features (such as interactivity).

Web AR

Several open-source JavaScript libraries can be utilized for web AR visualisation. Commercial libraries are also shown and used with the use of free trial versions.

Three.js + AR.js

- AR.js is an open-source library built on Three.js that enables markerless AR using the WebXR API
- Features:
 - Support for markerless AR (plane detection via WebXR)
 - Rendering 3D objects in a scene with Three.js
 - Well-documented and easy to use
 - Supports marker-based AR
 - Image tracking support (recognition and tracking of 2D images).
- Advantages:
 - Speed.
 - Works on most modern devices supporting WebXR.
- Usage:
 - Suitable for simple AR projects directly in the browser.

A-Frame

- A-Frame is a framework based on HTML and Three.js for creating VR/AR applications with WebXR support
- Features:
 - Simple AR scene setup using HTML tags (<a-scene>)
 - Support for markerless AR via WebXR

- Advantages:
 - Easy integration for developers with minimal JavaScript knowledge
 - Extendable through components
- Usage:
 - Great for rapid prototyping and visually-oriented AR applications

MindAR

- MindAR is a lightweight library focused on marker AR
- Features:
 - Support for 2D image and surface detection
 - Compatible with Three.js and A-Frame
- Advantages:
 - Lightweight and fast
 - Easy to use for simple AR applications
- Usage:
 - For developing applications with low performance requirements

Model-viewer

- Web component developed by Google that enables easy viewing of 3D models and integration of augmented reality (AR) into web pages
- Features:
 - It is designed to be simple to use while offering advanced rendering and interaction capabilities
 - Used as an HTML tag, making it possible to embed a 3D model into a web page without advanced programming skills
 - Supports GLTF/GLB formats
 - Interactive controls - rotation, zoom, and pan
- Great for model viewing on mobile devices, including AR functionality.

Markerless AR

The most basic visualisation method for augmented reality is markerless AR, which displays the virtual model based on the user's scan of an environment (e.g. a table or floor). Once the environment has been scanned, the virtual element can be placed in space and can be therefore moved (on scanned horizontal or vertical planes), rotated or resized.

Firstly, a plain html page is created using [Glitch](#) (free web page hosting platform). Secondly, the model-viewer is added to the page with a simple sample visualisation. The sample model is then changed with 3D model of spatial data – in this case church. As a final step, the page is visually edited and enhanced with some information about the shown model.

Zaniklý kostel na Zhůří u Čachrova

Kostel Nejsvětější Trojice byl postaven v roce 1762 (podle jiných údajů 1761) na místě bývalé kaple, která zde stála již v roce 1684. Kostel byl vysvěcen v roce 1763 a v roce 1766 došlo k založení farnosti. Matriky jsou vedeny od roku 1784. K rozšíření kostela došlo v roce 1793 a v roce 1809 kostel vyhořel – jednalo se pravděpodobně ještě o dřevěnou stavbu. Na náklad majitele bystřického panství hraběte P. Gundelfingena byl nově „z kamene vystavěn“.

Při prudké přestřelce na konci 2. světové války (5. 5. 1945) mezi americkými a německými vojáky, byla zničena střecha kostela. Ta byla po válce obnovena a bohoslužby se tady konaly až do dubna roku 1952. Pak celé území pohltil Vojenský újezd Dobrá Voda a došlo k úplné likvidaci celé obce. Na místě kostela byla vystavěná nová kaple a opět zasvěcena Nejsvětější Trojici. K slavnostnímu vysvěcení došlo 3. října 1999.

Zdroj textu: <https://kaplicky.cesty.in/zhuri>



Simple web page with church visualisation done by model-viewer

The final web application is freely available using web browser on both Android and iOS. When the page is loaded on a mobile device, the 3D model can be viewed in AR, if enabled.



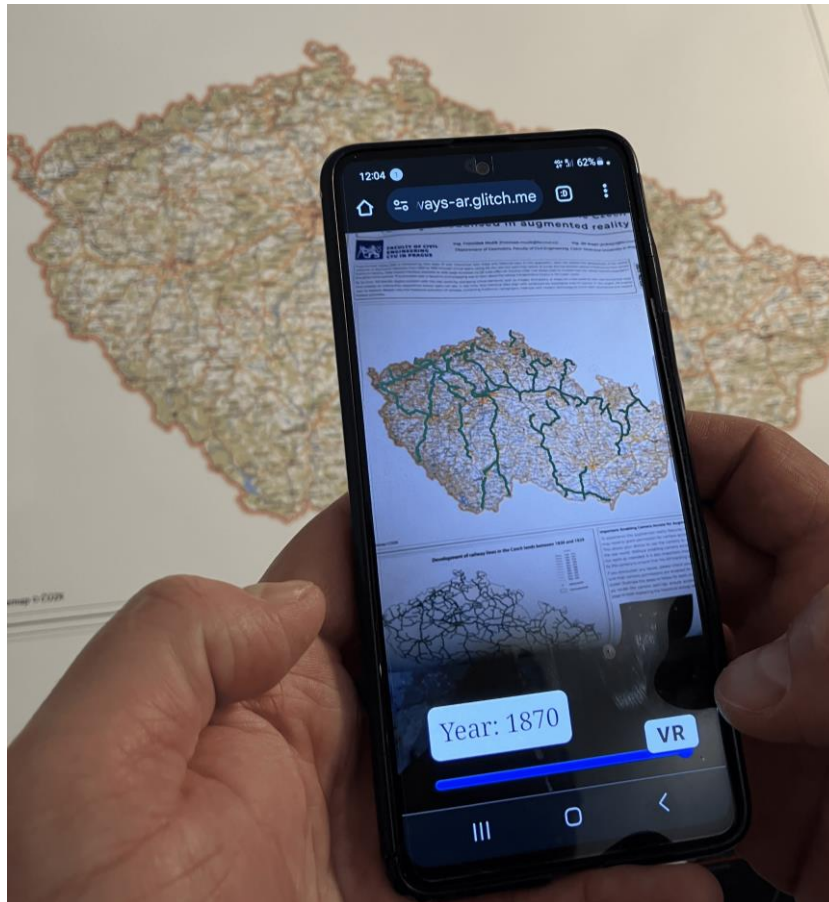
CAD model of a church viewed in AR using model-viewer

Marker AR

Marker AR places the virtual object based on the recognition of a specific marker. The object is directly tied to this marker. The applications are then divided into those that allow the user to move, rotate and resize the object - as in the previous case. The second type of applications uses the marker as an anchoring element that directly defines the size, rotation and position of the virtual model. In this case, the displayed virtual model is directly tied to the marker.

The example of marker AR uses a combination of two JavaScript libraries – A-Frame and MindAR. Firstly, it is needed to instal both libraries from their documentations. Secondly, Image Target needs to be created based on the printed map that is used as a marker anchoring the AR visualisation. The creation of Image Target is done with the use of web tool available from the MindAR documentation. It is necessary to keep the same scale (size) and aspect ratio for the base map and visualized AR layers – this has to be prepared in GIS in advance. Then the Image Targets Compiler tool is used to create a preparation for mounting the AR at several scales, including the display of anchor points.

After generating the Image Target, it is possible to create basic AR visualisation, now only with one AR map layer. Used samples from documentation are replaced with paths to desired map layer that will be viewed in AR and anchored to printed map. After successful implementation of the previous step, final visualisation can be completed by adding more AR map layers and time slider to sort and filter them by user input as seen on the picture bellow. The final web application is freely available using web browser on both Android and iOS.



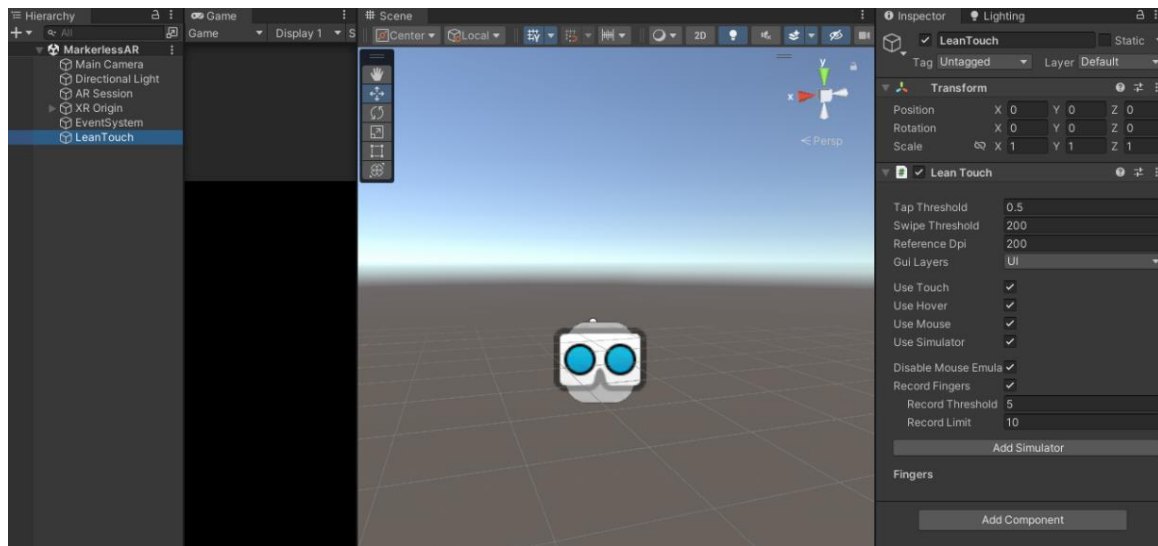
Additional map layer added through AR on a printed map

Unity

Unity is a game engine and a development platform that concentrates common features used in computer games, allowing faster and cheaper development of new games/applications. It is used for creating standalone applications. Unity is a very robust tool that offers wide range of possibilities for developers and therefore it can be challenging to use at start. During this course, only small part of Unity possibilities is utilized so that students are not overloaded with new information.

Firstly, students are taught how to navigate in Unity UI and how to create simple application that consists of two scenes (menu and AR visualisation). Then some navigating buttons are created to switch between scenes. These buttons are operated by simple C# script attached to them. This is the first example of combining visual Unity UI with C# scripting, which will be needed in next steps. Students are also introduced to the most common C# Unity methods, such as `Awake()`, `Update()` and `Start()`.

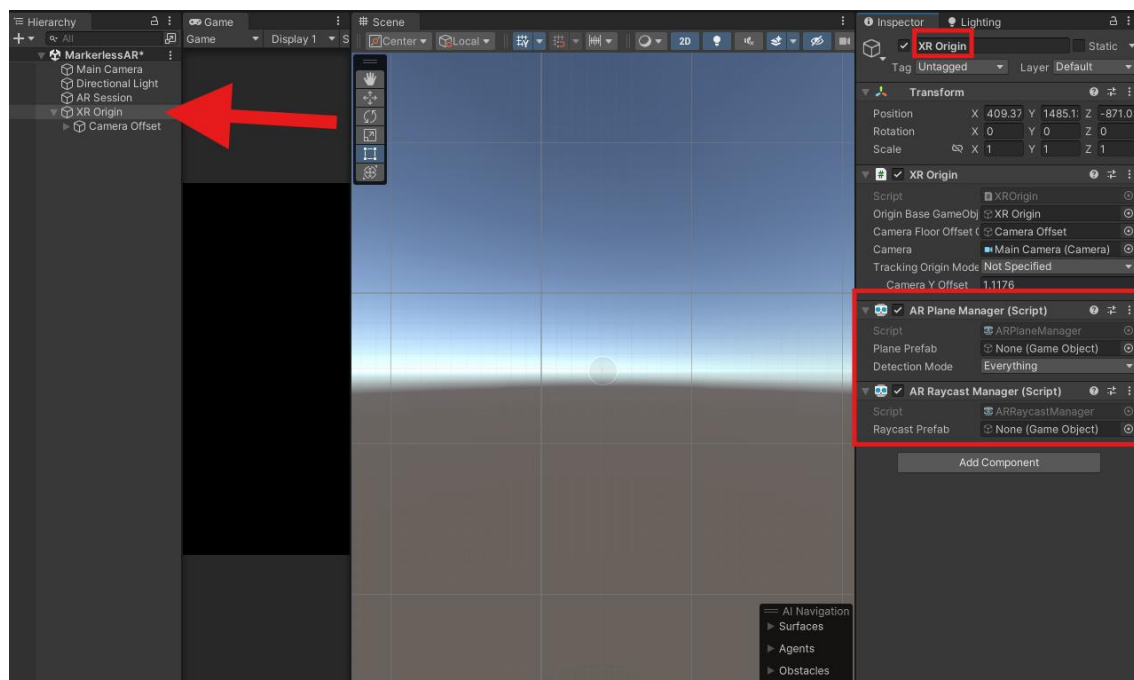
Then, the Android is set as a target build platform, which results in setting the project properly for Android development. Current state of the applications with two scenes and operating buttons is then built and sent to Android device for testing. Following this step, the creation of the first AR visualisation in Unity can be started. It will utilize the markerless AR method, which students already know from the previous lesson (web AR), so the basic principles should be familiar to them. This is important because they need to know how to use different approaches for creating different applications (web/mobile).



Unity user interface

Markerless AR in Unity

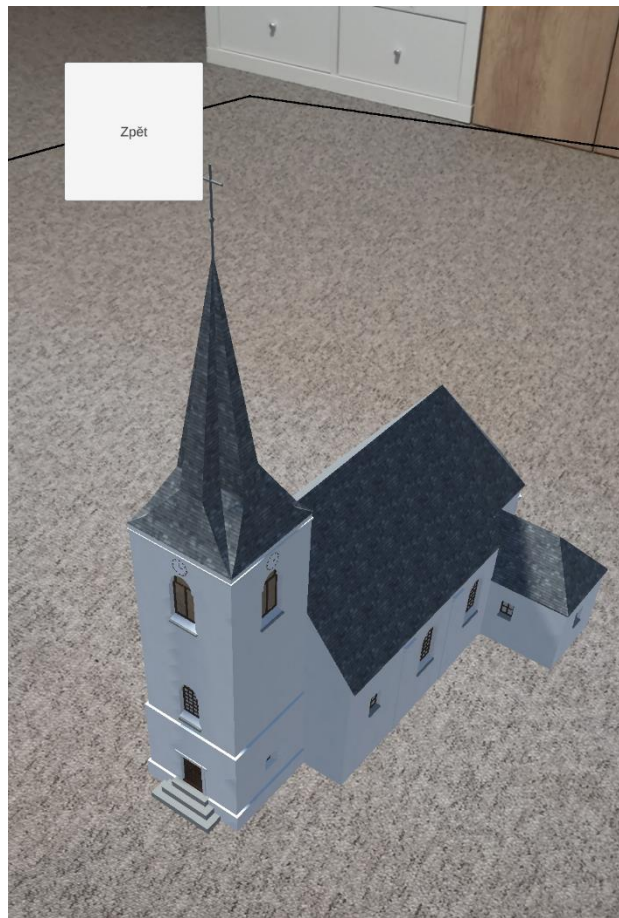
The creation of markerless AR visualisation in Unity is more complex compared to the web AR, which is why students were first introduced to the web AR. Using AR Foundation documentation, the new blank scene is extended with game object providing smooth and correct AR functions, such as AR Session for controlling the AR session and XR Origin, which manages the position and orientation of AR objects in the space. Then two more AR game object are added to the active scene as child object of XR Origin. AR Plane Manager - enables detection of flat surfaces (e.g., tables, floors) and AR Raycast Manager - facilitates interaction with the real world, such as placing objects on detected surfaces.



Adding new game objects to the active scene

After this, the AR plane prefab is created and added as component to the XR Origin. This ensures that scanned horizontal or vertical planes are visualised and the user is able to interact with them by tapping and placing 3D model in AR space. Two final steps are remaining for smooth AR visualisation – first of them consists of writing a simple C# script that places the 3D object on scanned scene. This script uses the AR Raycast Manager component in XR Origin. As a final step, Lean Touch asset is added to the project from Unity Asset Store. This asset deals with the 3D model controls based on user inputs – such as scaling, dragging and rotating viewed 3D object.

Finally, the application can be built and sent to the connected Android device for testing. Screenshot of this very simple application is showed below.



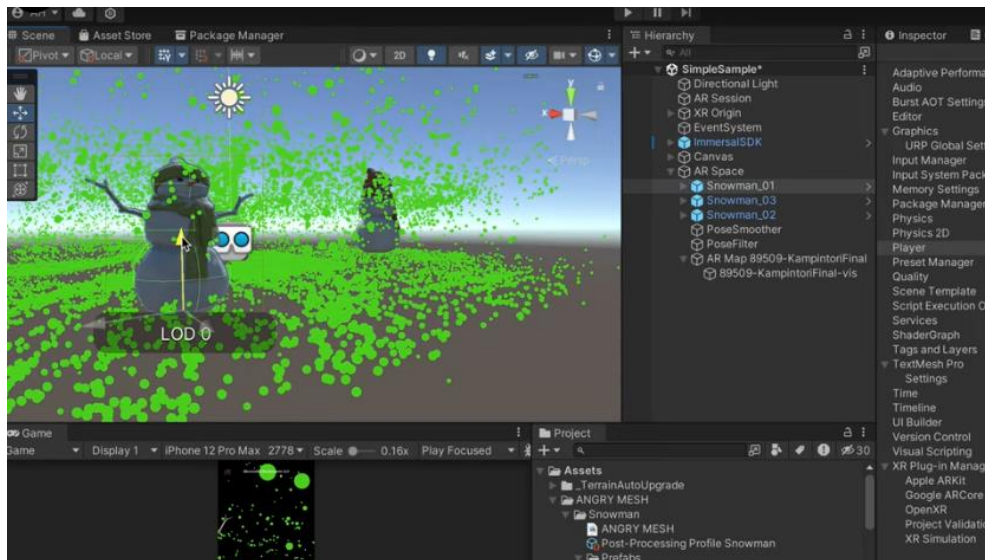
CAD model visualised in mobile application created in Unity game engine

Location-based AR

Location-based AR places and visualises the virtual model at a predefined location in the environment (e.g. city street or field). Therefore, the model cannot be displayed at a different location using this method, as it is tied to a specific location.

For this demonstration, the asset [Immersal](#) is used. Immersal is a commercial Unity asset but it offers quite wide free trial version, which is used at the lesson. Firstly, students need to register at Immersal website and download the Mapper application available for both iOS and Android. With the Immersal Mapper, they will scan the classroom using mobile phones. This process will create point cloud from dozens of photos taken by phone. The point cloud will be saved on

cloud, from where it can be downloaded and used in Unity or its mesh can be downloaded in PLY format.



Setting the position of a 3D model in AR; Source: <https://developers.immersal.com/docs/unitysdk/tutorial/>

After a successful point cloud generation, the work further continues in Unity. Point cloud is added to the active scene and the position of selected model is set accordingly to the position of the point cloud. With the use of Immersal sample scene, the whole process in Unity is very simplified, so the application is now ready for being deployed to mobile phone and tested.



Example of location-based AR to visualise a clergy house in the extinct village Zhůří

3D printing of spatial data

3D printing is a popular and fast growing industry that enables the creation of physical models made of plastic for many purposes. One of them can be 3D printing of spatial data (e.g. terrain, haptic map, CAD or photogrammetric model). Nowadays, this is already an established way of presenting 3D models and it is therefore highly desirable that students know at least the basic principles of this technology. Students should be able to create a model for 3D printing (or obtain it from the Internet and then modify it in appropriate software). Furthermore, it is important that they understand the 3D printing process to the extent that they can prepare the model on the computer to be printed without problems. It is also important to take into consideration the economic costs of 3D printing and therefore be able to optimise the parameters of 3D printing, including the use of the proper filament types.

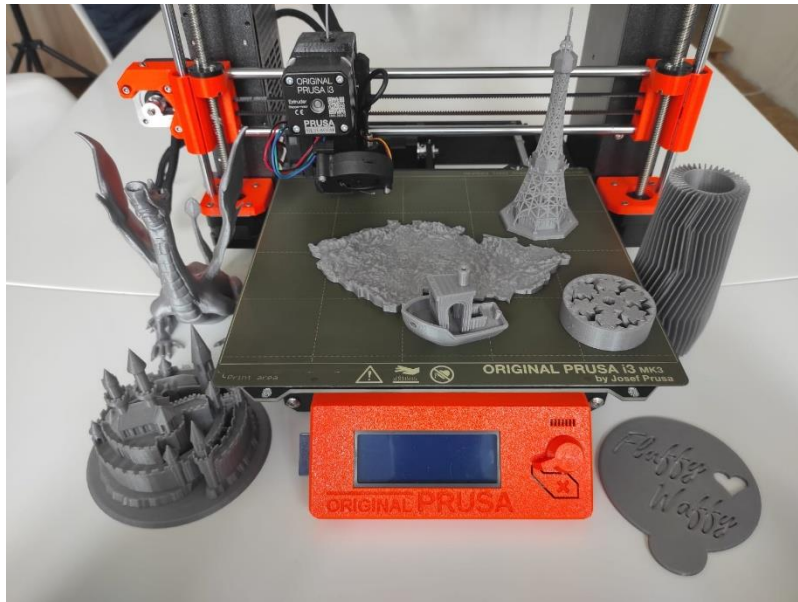


3D printed spatial data; Sources: https://blog.prusa3d.com/photogrammetry-3d-scanning-just-phone-camera_7811/, <https://3dprintingindustry.com/news/3d-printing-tactile-maps-versotek-29132/>, <https://www.chapelprints.co.uk/3d-maps>, https://blog.prusa3d.com/how-to-print-maps-terrains-and-landscapes-on-a-3d-printer_29117/

3D printing basics

3D printing or additive manufacturing (AM) is a process of creating three-dimensional (3D) solid objects from a digital file (Additive Manufacturing File - AMF). In additive processes, an object is created by laying down continuous layers of material (filament) until the entire model is complete.

Why to use a 3D printer? Traditional machining is demanding on resources and the machines themselves are very expensive. It is necessary to employ trained staff to operate and inspect the machines. Another disadvantage is the high initial cost of setting up a business. However, 3D printers are cheap, affordable, small and in many cases reasonably accurate. Thus, they provide a very efficient price-performance ratio and are ideal for use to print spatial data.



FFF printer; Source: <https://sspo.cz/2021/02/12/prusa-je-nejlepsi/>

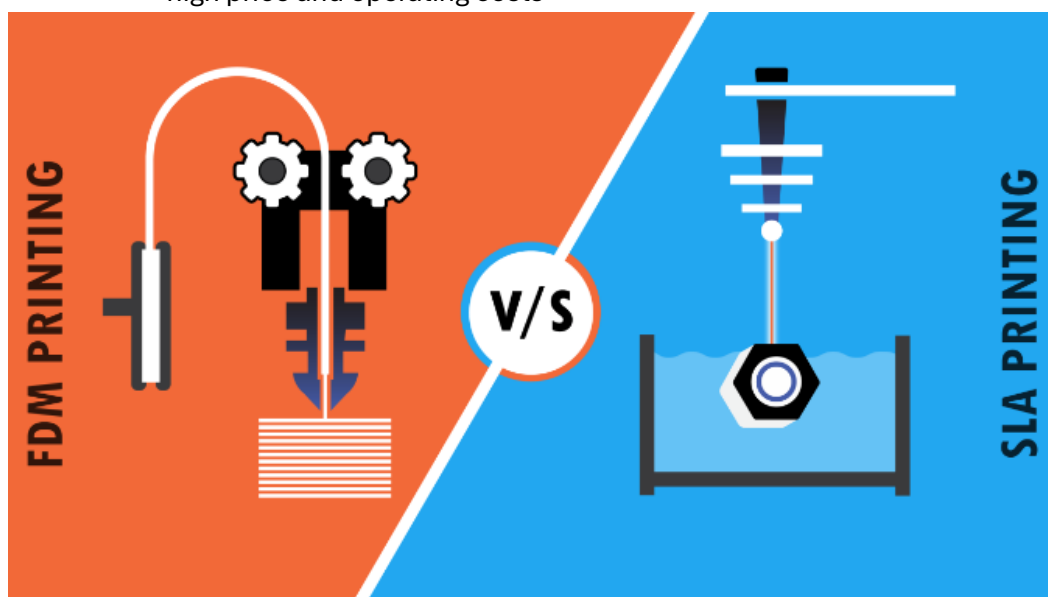
There are two main techniques of 3D printing

1) FFF (FDM)

- Fused Filament Fabrication / (Fused Deposition Modeling - trademark)
- filament = plastic material (wound string) that is fused in a printer at high temperature
- low price and operating costs and a wide choice of filaments
- lower print accuracy compared to SLA printing
- coarser surface of the printed model

2) SLA

- gradual curing of the photoreactive resin (resin) – done by laser or diode
- higher print accuracy and very smooth product surface
- high price and operating costs



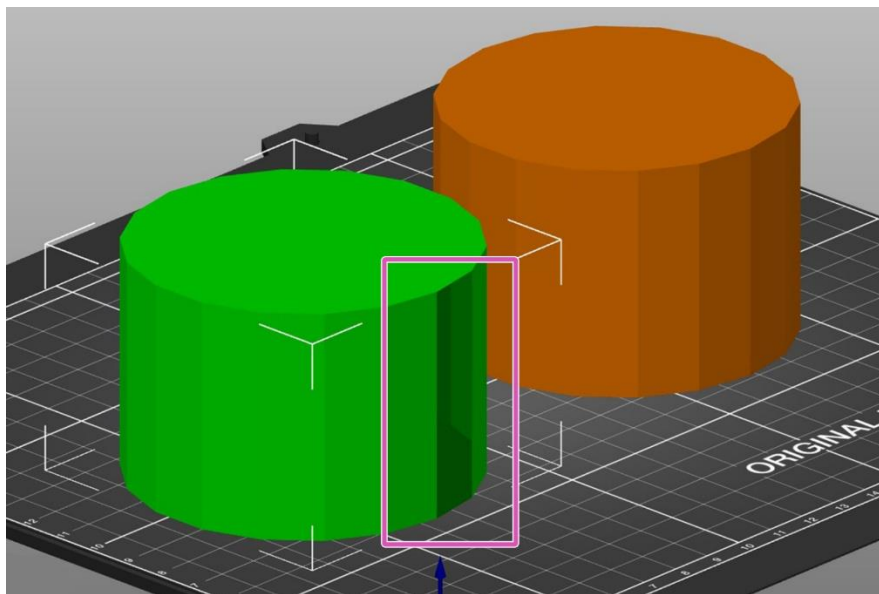
Printing methods; Source: <https://www.donit.co.za/fdm-vs-sla-3d-printing-which-is-right-for-you/>

Printing geodata

Data suitable for 3D printing can be obtained via several internet repositories, e.g. <https://www.thingiverse.com/>, <https://pinshape.com/> or <https://cults3d.com/>. Models downloaded from there repositories can be therefore edited in 3D modelling software (SketchUp, Blender, FreeCAD etc.). Models should be exported to STL data format before printing. STL is a data format that uses unstructured triangulated surface by the unit normal and vertices (ordered by the right-hand rule) of the triangles using a three-dimensional Cartesian coordinate system.

The second option is to create the data from scratch or as a result of photogrammetric or geodetic measurements, GIS data (terrain, map, land cover) or CAD modelling (architecture). This can be done in by several approaches but only few of them will be described further.

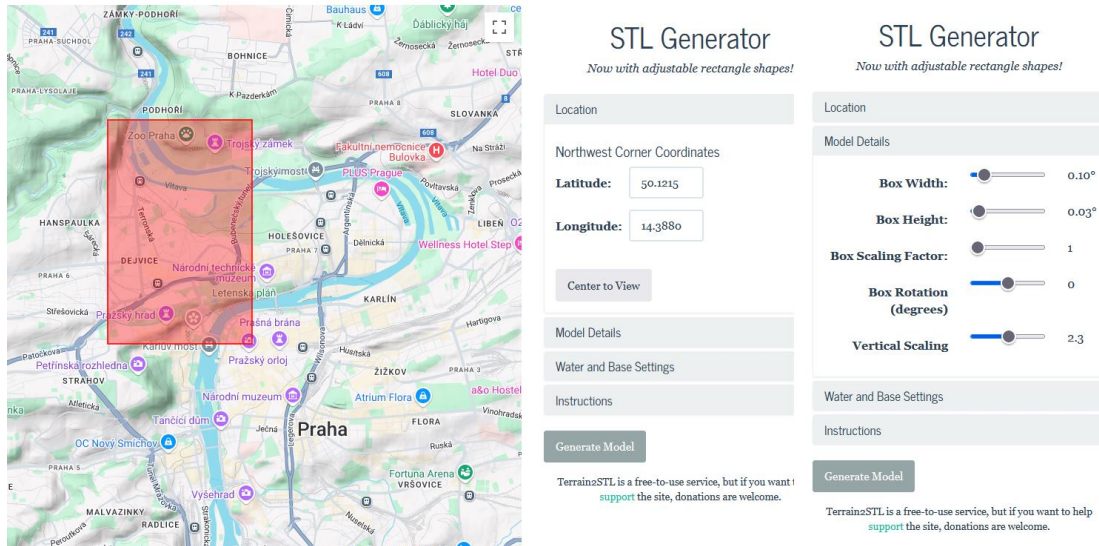
Before printing, it is necessary to follow a few rules to ensure that the 3D model will be printed properly. The first is the size of the model - it is important to set the correct scale, or reduce or enlarge the model as required before exporting. In this step, it is also necessary to think about the details and complexity of the model. If necessary, the model should be generalized or, on the other hand, details should be added. The third important requirement is the need to close all surfaces of the model so that it does not contain holes in the geometry. This can be done with the help of a variety of plugins available for the most commonly used software.



Hole in an object viewed in Prusa Slicer; Source: https://help.prusa3d.com/cs/article/modelovani-pro-3d-tisk_164135

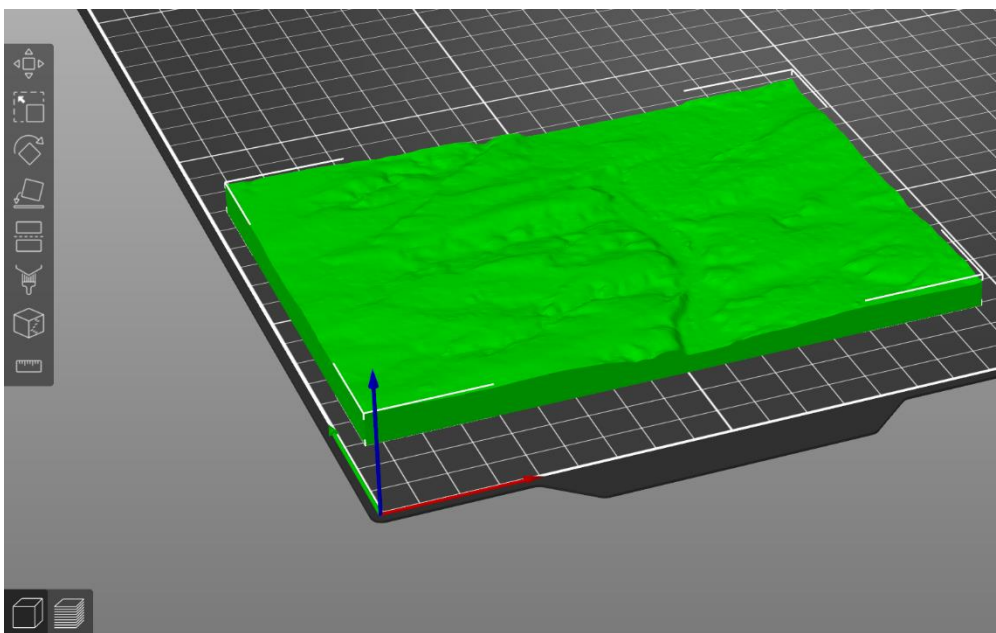
Terrain2STL

[Terrain2STL](#) is a web tool that generates STL model of the surface of Earth based on [SRTM3 dataset](#) from 2000, which has a resolution of about 90 meters on the equator. User has to draw a rectangle around the desired location on the map. Then certain parameters of the terrain model need to be set, such as the size of selected location, its rotation and vertical scaling. This is important to set as higher value if the area of interest is flat resulting in a better visual impression.



Terrain2STL interface

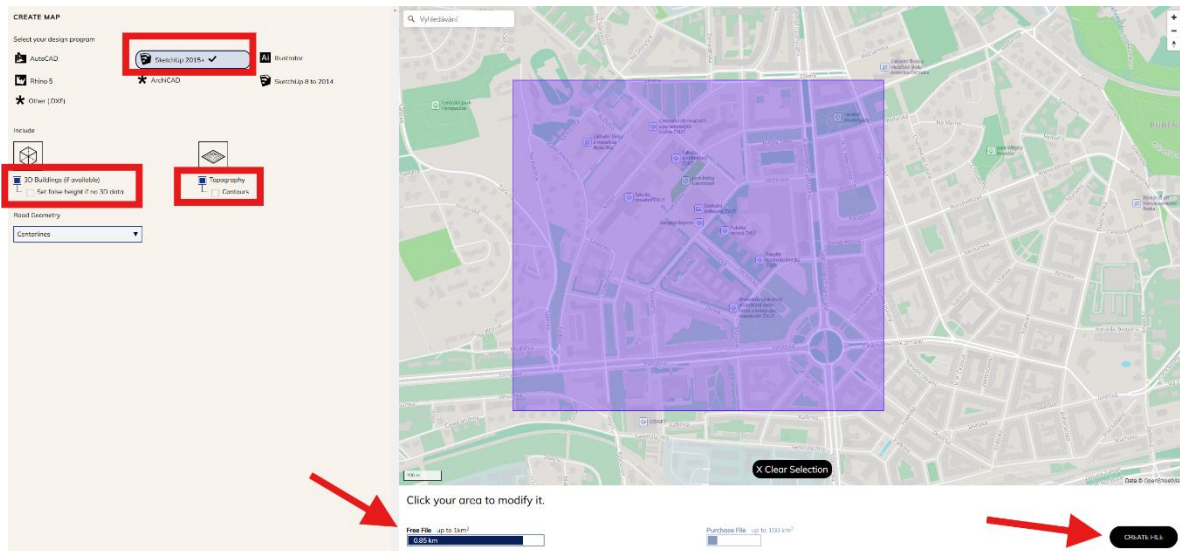
The generated STL model can then be edited in 3D modelling software or directly imported to 3D printer software (in this case Prusa Slicer).



Generated terrain in Prusa Slicer

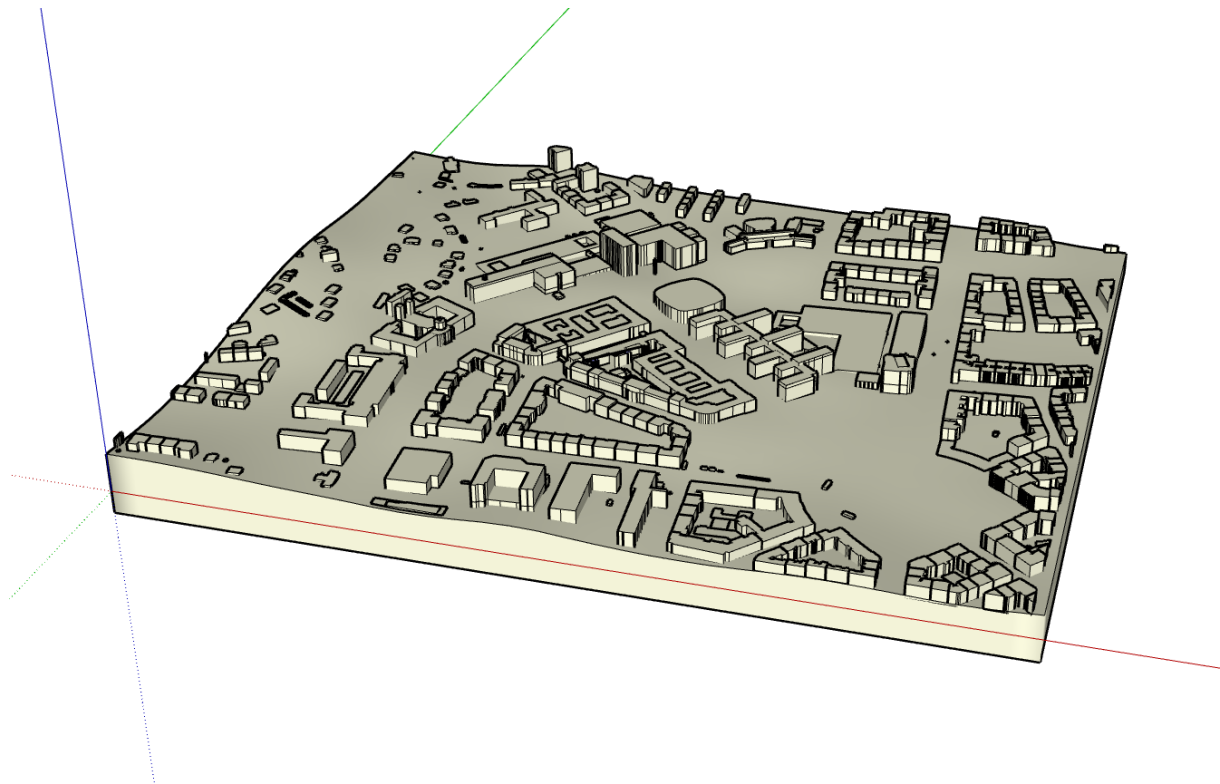
CAD Mapper

Web tool [CAD Mapper](#) allows user to download part of Open Street Map by drawing rectangle around desired location. However, the tool is free to use only for a smaller areas up to 1km². This tool offers wider options for the creation of the model, such as export directly to different 3d modelling software or selecting terrain with or without buildings and road geometry. The generated model can be also viewed on web before downloading.



CAD Mapper interface

The downloaded model can be further edited or generalised, as it consists not only of terrain and buildings, but also roads, tram lines or paths OSM data. After all necessary edits are done, it can be exported to STL format and opened in Prusa Slicer.



Generated terrain with buildings in SketchUp