## CRS

CRS = Coordinate Reference System. There are thousands of CRS but most of them are for local usage (e.g. each country has its own CRS). We will be mainly using two: World Geodetic System 1984 (WGS 84) and Universal Transverse Mercator (UTM). Simply put, WGS 84 is 3D with point of origin in Earth’s center and is measured in degrees. UTM is 2D and is measured in meters. This is very useful when calculating buffers etc. However, putting 3D globe to 2D plane comes with huge map deformations at the edges. Therefore, UTM is divided into 60 zones, there’s a CRS for each zone.

Navigating through the CRS might be confusing at first but using EPSG codes (EPSG is an authority managing CRS) and Google makes it much easier. Use found codes in QGIS to select appropriate CRS.

1. WGS 84
   1. Put to Google: WGS 84 EPSG
   2. You will get code: 4326
2. UTM e.g. for Belgium
   1. Find the right zone e.g. from UTM Wikipedia page (<https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system>)
   2. You will find zone 31, add N as north and put to Google: UTM 31n EPSG
   3. You will get code: 32631
   4. Note: there will be other variations of UTM, but it should be clear which one is the “main” one. If not, look for UTM with datum WGS 84.

## QGIS

QGIS is an open source GIS software for viewing, querying and editing geographical data.

Download link:

<https://qgis.org/en/site/forusers/download.html>

Standard installation process.

Start QGIS.

Displaying geographical data

* drag & drop (drag from directory, drop to QGIS canvas) well known formats such as Shapefiles, KML, GeoJSON, GeoDB
* All options regarding adding layers are in menu Layer > Add Layer

### Setting CRS

In QGIS you must set CRS for each layer. Most GIS formats contain the information about the CRS used and the setting of CRS is not required. But for some formats, including CSV (simple comma separated text file), the CRS must be set. If it’s set incorrectly then data will be displayed incorrectly.

CRS can be also set to the QGIS canvas displaying all the layers. Currently used CRS is displayed in bottom right corner. Click it to change the CRS.

### Plugins

A lot of functionality can be added to QGIS by installing plugins. Go to menu Plugins > Manage and Install Plugins…

Go through the list of plugins, read the description and install what you need.

Recommend plugins to install:

* QuickOSM
  + For downloading OSM data
  + After install, the plugin can be started from an icon of a magnifying glass on a green background
* HCMGIS
  + Offers big selection of base maps such as Google Maps, Esri etc.
  + After install, the plugin appears in the menu

### Issues

QGIS is not perfect. Here are issues I encountered when preparing data for recreations:

* You can save your current work as a ‘project’. However, when you re-open the project the CSV layers are misplaced. It might be better to always start with a blank project.
* Exporting attributes with language specific characters such as “é“, replaces the characters with incorrect characters. The encoding selection doesn’t seem to work properly.

## Getting a bounding box from terrain.party

If you are interested in recreations, you most likely know about terrain.party. Using this great tool, it is possible to select any area around the world and then download heightmaps for that area in the C: S format. While you get the heightmap you also get the coordinates of the selected area in the attached readme file. This is the easiest way of getting a bounding box of the recreated area. However, the disadvantage is that in terrain.party the box is either 17x17 or 18x18 km whereas the real game area’s dimensions is 17.28 x 17.28 km. It’s close enough for most recreations but not accurate enough for very detailed recreation work.

On the other hand, if you created your own bounding box (e.g. by following the steps mentioned below), you can feed the coordinates (in WGS 84) to the terrain.party API directly by specifying the URL:

<http://terrain.party/api/export?name=brussels&box=4.4979367,50.9286481,4.2479785,50.7704117>

When you enter this to the browser, heightmaps of exactly 17.28x 17.28 km will be downloaded.

## Creating a bounding box in QGIS

C:S game area is a box of 17280 x 17280 meters. If we don’t consider scaling down or up, the area you choose to recreate should be also 17280 x 17280.

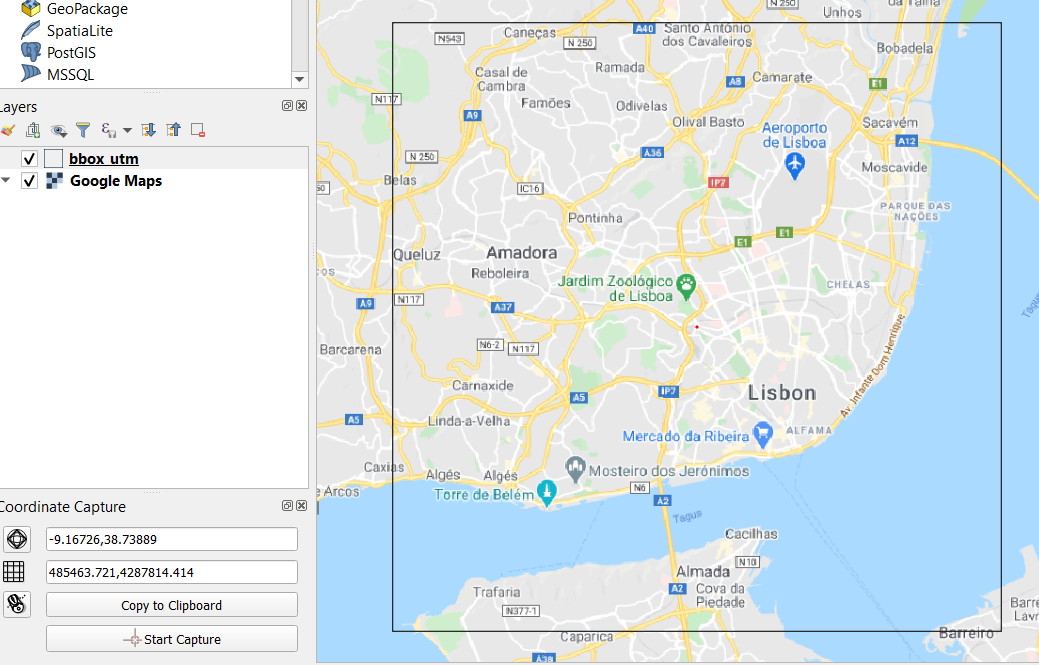
First find out the UTM zone of the area you want to model. You can try googling (e.g. Lisbon UTM zone) but safer is figure it out by looking at the map with the UTM zones, e.g. here:

<https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system#/media/File:Universal_Transverse_Mercator_zones.svg>

For e.g. Lisbon it’s zone 29 and obviously north. Google “UTM 29N epsg” and you will get the EPSG code 32629. Use this code in QGIS to get the correct CRS.

Creating a bounding box in QGIS:

1. Start QGIS, new project
2. Plugins > Manage and Install plugins…
   1. Search for Coordinate Capture and enable (if not enabled already).
   2. Search for HCMGIS and install (if not installed already).
3. Menu > HCMGIS > BaseMap > Google Maps (or other)
4. Navigate to the area you whish to model in C:S
5. In Coordinate Capture plugin (should be left bottom) click on Start Capture
6. Click in the map where you believe the center of the modeled area should be
7. Plugins > Python console
8. Open Script > load provided bbox.py (on GitHub in folder help\_codes)
9. Update the values at the top of the script
   1. Coordinates: copy from the metric coordinates (2nd row) and paste it to the script after the ‘x,y = ’
      1. E.g. x,y = 485463.721,4287814.414
   2. CRS: write the EPSG code for your area to the variable inProjTxt
      1. E.g. inProjTxt = 'epsg:32629'
10. Click the Run Script button
11. Open Windows explorer and in the navigation bar type: %temp%. Search for newest files. You will find:
    1. Bbox\_utm.csv
    2. Bbox\_wgs.csv
    3. Cs\_area\_info.txt (this contains the center lat and lon and URL for terrain.party to get heightmap)
12. In QGIS, Layer > Add Layer > Add delimited text layer
    1. File name: bbox\_utm.csv
    2. Geometry CRS: the UTM zone of your choosing (use EPSG code to find it)



## Creating image overlay in QGIS

C:S mod Image Overlay allows displaying PNG images in the game. This is often used for recreation works (e.g. creating road network). QGIS can be used to produce accurate map images from various data sources.

### Map image from raster data (base maps)

Using plugin HCMGIS you can display wide range of base maps. Choose which base map best fits the purpose of recreation. To create the map image, follow these steps:

1. Start QGIS, new project
2. Add bounding box for your area:
   1. Layer > Add Layer > Add Delimited Text Layer
   2. Settings:
      1. File name: e.g. C:\temp\bbox\_utm.csv
      2. Geometry field: geometry
      3. Geometry CRS: corresponding UTM (e.g. EPSG: 32629)
   3. Add and then close
   4. (This will change the canvas CRS to the UTM zone of your choosing (check the bottom right, it should have the right ESPG code – if not change it).
3. Choose a base map: e.g. HCMGIS > BaseMap > Google Maps
4. Zoom in to display more detail in the map
5. Make sure only layers that you want in the map image are displayed
6. Export map as image:
   1. Project > Import/Export > Export Map to Image
   2. Extent: Calculate from Layer – bbox\_utm
   3. Click Save
7. Copy the created map image to c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\

### Map image from vector data

More customizations are possible when using vector data (points, lines, polygons). Let’s look at generating map image from lines of roads.

1. Start QGIS, new project
2. Add bounding box for your area:
   1. Layer > Add Layer > Add Delimited Text Layer
   2. Settings:
      1. File name: e.g. C:\temp\bbox\_utm.csv
      2. Geometry field: geometry
      3. Geometry CRS: corresponding UTM (e.g. EPSG: 32629)
   3. Add and then close
   4. *Note: This will change the canvas CRS to the UTM zone of your choosing (check the bottom right, it should have the right ESPG code – if not change it.*
3. Change symbology of bbox\_utm layer:
   1. In the layer list, double click the bbox\_utm
   2. Select Symbology (third from top)
   3. Select Simple fill
   4. Fill color: black (RGB: 0,0,0)
4. Add roads dataset (i.e. roads\_rwo.csv). If you don’t have this, see section ‘Preparing OSM vector data for GeoSkylines’
5. Change symbology of roads layer:
   1. In the layer list, double click the roads layer
   2. Select Symbology (third from top)
   3. Select Simple line
   4. Fill color: white (RGB: 255,255,255)
   5. Stroke width: Hairline
   6. *Note: likely scenario is not displaying all road types. In this case you have to filter the data or using Symbology change the ‘Single Symbol’ to ‘Categorized’ based on a column specifying the road type (then click Classify and set the color and stroke width to all lines).*
6. Zoom in to display more detail in the map
7. Make sure only layers that you want in the map image are displayed
8. Export map as image:
   1. Project > Import/Export > Export Map to Image
   2. Extent: Calculate from Layer – bbox\_utm
   3. Click Save
9. Open the saved PNG in an image software and set the background as transparent.
   1. I used IrfanView: Save As > Save Transparent Color & Use main window color for transparency
10. Copy the resulting map image to c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\

With vector data it is possible to do much more (e.g. buffering different types of water ways) but this requires more skill in GIS techniques.

## Creating C:S heightmap

The easiest option to retrieve a heightmap in the C:S format is to use terrain.party. This tool, however, provides global datasets with lower resolution than the game’s resolution of 16 meters per pixel. There might be a more accurate data set available for your region (e.g. national Digital Elevation Model). If you have access to such dataset you can create the heightmap using QGIS:

1. Obtain DEM tile in TIF format for the specified area
   1. tile should be bigger than 17.28 x 17.28 km
   2. tile should be of 16 unsigned integer format
2. Add UTM bounding box layer of your area (i.e. created in steps described above)
3. Add to DEM tile to QGIS (drag & drop)
4. The DEM tile will be displayed in the layer list. Note the numbers below it. It is the lowest elevation and highest elevation of the tile. Write down both numbers for later use.
5. In a notepad write a command from these parts:
   1. gdal\_translate
   2. -scale:
      1. Format: -scale (minElev-60) (1024 or maxElev whichever is higher) 0 65536
      2. Example 1: -scale 136 1024 0 65536
      3. Example 2: -scale 336 1424 0 65536
      4. Note: for coastal areas the first number will be: minElev – 40 (40 is the sea level in C:S)
   3. -of PNG <input\_tiff\_tile\_path> <output\_png\_file\_path>
   4. -projwin:
      1. Get it from cs\_area\_info.txt (generated by provided Python script – see above regarding creating bounding box)
   5. If source DEM tile is not in 16 unsigned integer format, then add option:
      1. -ot UInt16
6. In your Windows search box type: OSGeo4W Shell
7. Start OSGeo4W Shell
8. Copy your command from notepad to OSGeo4W Shell
9. Execute the command
10. Copy the resulting PNG to:
    1. c:\Users\<username>\AppData\Local\Colossal Order\Cities\_Skylines\Addons\MapEditor\Heightmaps\

The resulting command should look like this:

gdal\_translate -scale 136 1024 0 65536 -projwin 653819.734498533 5503970.495422958 671095.4346983728 5486693.562522869 -of PNG C:/data/CS/Olomouc/DMR/DEM\_tile.tif C:/data/CS/Olomouc/DMR/OlHeightMap.png

The scale option rescales the real height values to the C:S height values. There are 4 numbers after the -scale keyword. The first number is the lowest elevation of your resulting PNG heightmap. We want this to be the default C:S flat land (60 meters). Therefore, the first number is calculated as the lowest elevation in the DEM tile – 60 meters (e.g. 193 – 60). The second number is the highest elevation in the resulting PNG heightmap. The maximum elevation in C:S is 1024 meters. Therefore, the second number is the higher of either 1024 or maximum elevation of the source DEM tile. Third and fourth number are representing the bit range of the source DEM tile, 0 is the lowest elevation, 65536 is the highest elevation.

## Preparing OSM vector data for GeoSkylines

It is possible to use any source of data available. But in most cases the source of the data will be OSM. Therefore, the documentation will focus on working with OSM data.

There are various ways to obtain OSM data (directly from openstreetmap.org, OverPass API). But since it’s not just about getting the data but also about processing them in a meaningful way, the option I recommend is to use QGIS and plugin QuickOSM (very user friendly compared to any other way). The only small problem I found in this approach is that while outputting data from QGIS, some language specific characters (such as é) are not exported properly. But this can be then fixed in the resulting CSV file by simply opening the file in notepad or Excel and using Ctrl+H (replace all) function, e.g. ś -> š.

Also, plugin HCMGIS is a great help. This plugin offers various base maps. Adding them to your map helps with navigation.

After running QuickOSM query, OSM data will be displayed in QGIS as temporary (in memory) layers. We will save the data as CSV. This will make the data permanent as well as give us the format GeoSkylines requires.

If you are just testing this functionality (e.g. for learning purposes) then I recommend starting with small areas (i.e. small amount of data).

### Road data (OSM tag = highway)

To download OSM road data in QGIS:

1. Navigate to the area you want the data from
   1. Either add bounding box layer (e.g. created in steps described above)
   2. Or just use base map and navigate to the approximate area
2. Start QuickOSM by clicking the icon with a magnifying glass on a green background
3. Settings:
   1. Key: highway
   2. Value: leave empty unless you are interested in specific roads
   3. Spatial context (3rd option):
      1. either Canvas extent (what you see on the monitor)
      2. or Layer extent: your bounding box layer (e.g. bbox\_wgs)
4. Run query and then exit when query is finished
5. There will be 3 new layers added to the layer list. We are interested just in the lines.
6. Right-click the highway lines layer and select Export > Save Feature As...
   1. Settings:
      1. Format: Comma Separated Values [CSV]
      2. File name: e.g. c:\temp\roads\_rwo.csv
      3. CRS: EPSG: 4326 – WGS 84
      4. Layer Options > Geometry: AS\_WKT
   2. OK

In case of processing big amount of data (i.e. more data = more problems) you might experience some errors. In that case I recommend opening the attribute table in QGIS (right-click layer > Open Attribute Table) and addressing the exact problem (e.g. I had an error due to column duplicity so I removed one of the columns) or, if that is not enough, removing all fields except:

* Osm\_id, highway, name, lanes, oneway, bridge

For roads, GeoSkylines requires these columns in the roads\_rwo.csv for properly generating road network:

* Id
  + this one is actually used only internally
  + OSM source: osm\_id
* road\_name
  + used to name the segment in C:S
  + OSM source: name
* road\_type
  + used to find matching C:S road type (NetInfo)
  + OSM source: highway
* one\_way
  + used to find matching C:S road type (NetInfo)
  + OSM source: oneway
* lanes
  + not used at this moment (but might be in the future)
  + OSM source: lanes
* bridge
  + used to find matching C:S road type (NetInfo)
  + OSM source: bridge
* geometry
  + actually just this one is mandatory
  + used to create segments in C:S
  + In the created CSV file the column name will be: WKT

Open the roads\_rwo.csv in notepad or Excel and rename the columns accordingly.

You can remove the remaining columns or leave them (they will be ignored by GeoSkylines).

A screenshot of a cell phone

Description automatically generated

In the above picture you can also notice incorrectly exported characters. This can be fixed by looking up the original name in QGIS (or if you are working with a known area you might know the correct names from memory) and then using Ctrl+H function in Excel or notepad.

### Flowing water data (OSM tag = waterway)

To download OSM waterway data in QGIS:

1. Navigate to the area you want the data from
   1. Either add bounding box layer (e.g. created in steps described above)
   2. Or just use base map and navigate to the approximate area
2. Start QuickOSM by clicking the icon with a magnifying glass on a green background
3. Settings:
   1. Key: waterway
   2. Value: leave empty unless you are interested in specific waterway type
   3. Spatial context (3rd option):
      1. either Canvas extent (what you see on the monitor)
      2. Layer extent: your bounding box layer (e.g. bbox\_wgs)
4. Run query and then exit when query is finished
5. There will be 3 new layers added to the layer list. We are interested just in the lines.
6. Right-click the waterway lines layer and select Export > Save Feature As...
   1. Settings:
      1. Format: Comma Separated Values [CSV]
      2. File name: e.g. c:\temp\waterway\_rwo.csv
      3. CRS: EPSG: 4326 – WGS 84
      4. Layer Options > Geometry: AS\_WKT
   2. OK

For waterways, GeoSkylines requires these columns in the waterway\_rwo.csv for digging the waterway basins:

* name
  + not used at the moment
  + OSM source: name
* waterway
  + used to apply different settings from configuration parameters: ImportWaterWayTypes, ImportWaterWayDepths, ImportWaterWayWidths
  + OSM source: waterway
* geometry
  + actually just this one is mandatory
  + used to dig waterway basins in C:S (basins are not filled with water)
  + In the created CSV file the column name will be: WKT

Open the waterway\_rwo.csv in notepad or Excel and rename the columns accordingly.

### Standing water data (OSM tag = natural:water)

To download OSM standing water data in QGIS:

1. Navigate to the area you want the data from
   1. Either add bounding box layer (e.g. created in steps described above)
   2. Or just use base map and navigate to the approximate area
2. Start QuickOSM by clicking the icon with a magnifying glass on a green background
3. Settings:
   1. Key: natural
   2. Value: water
   3. Spatial context (3rd option):
      1. either Canvas extent (what you see on the monitor)
      2. Layer extent: your bounding box layer (e.g. bbox\_wgs)
4. Run query and then exit when query is finished
5. There will be 3 new layers added to the layer list. We are interested just in the polygons (areas).
6. The water reservoirs are defined as MULTIPOLYGON but GeoSkylines works only with POLYGONS. Therefore, go to menu Vector > Geometry Tools > Multiparts to Singleparts...
   1. Input layer: natural\_water
   2. Run
7. Right-click the new Single parts layer and select Export > Save Feature As...
   1. Settings:
      1. Format: Comma Separated Values [CSV]
      2. File name: e.g. c:\temp\water\_rwo.csv
      3. CRS: EPSG: 4326 – WGS 84
      4. Layer Options > Geometry: AS\_WKT
   2. OK

For water reservoirs, GeoSkylines requires these columns in the water\_rwo.csv for digging the basins:

* Id
  + OSM source: osm\_id
* geometry
  + used to dig water reservoir basins in C:S (basins are not filled with water)
  + In the created CSV file the column name will be: WKT

Open the water\_rwo.csv in notepad or Excel and rename the columns accordingly.

### Amenity/Services data (OSM tag = amenity)

To download OSM amenity data in QGIS:

1. Navigate to the area you want the data from
   1. Either add bounding box layer (e.g. created in steps described above)
   2. Or just use base map and navigate to the approximate area
2. Start QuickOSM by clicking the icon with a magnifying glass on a green background
3. Settings:
   1. Key: amenity
   2. Value: leave empty unless you are interested in specific amenity type
   3. Spatial context (3rd option):
      1. either Canvas extent (what you see on the monitor)
      2. Layer extent: your bounding box layer (e.g. bbox\_wgs)
4. Run query and then exit when query is finished
5. There will be 3 new layers added to the layer list. We are interested just in the points.
6. Right-click the amenity points layer and select Export > Save Feature As...
   1. Settings:
      1. Format: Comma Separated Values [CSV]
      2. File name: e.g. c:\temp\amenity\_rwo.csv
      3. CRS: EPSG: 4326 – WGS 84
      4. Layer Options > Geometry: AS\_WKT
   2. OK

For services, GeoSkylines requires these columns in the amenity\_rwo.csv for creating the service buildings:

* id
  + this one is actually used only internally
  + OSM source: osm\_id
* amenity
  + used to find matching C:S service type (BuildingInfo)
  + OSM source: amenity
* geometry
  + used to create C:S service (building)
  + In the created CSV file the column name will be: WKT

Open the amenity\_rwo.csv in notepad or Excel and rename the columns accordingly.

### Railways data (OSM tag = railway)

To download OSM railway data in QGIS:

1. Navigate to the area you want the data from
   1. Either add bounding box layer (e.g. created in steps described above)
   2. Or just use base map and navigate to the approximate area
2. Start QuickOSM by clicking the icon with a magnifying glass on a green background
3. Settings:
   1. Key: railway
   2. Value: leave empty unless you are interested in specific rail type
   3. Spatial context (3rd option):
      1. either Canvas extent (what you see on the monitor)
      2. Layer extent: your bounding box layer (e.g. bbox\_wgs)
4. Run query and then exit when query is finished
5. There will be 3 new layers added to the layer list. We are interested just in the lines.
6. Right-click the railway lines layer and select Export > Save Feature As...
   1. Settings:
      1. Format: Comma Separated Values [CSV]
      2. File name: e.g. c:\temp\rails\_rwo.csv
      3. CRS: EPSG: 4326 – WGS 84
      4. Layer Options > Geometry: AS\_WKT
   2. OK

For railways, GeoSkylines requires these columns in the rails\_rwo.csv for generating the railway segments:

* id
  + this one is actually used only internally
  + OSM source: osm\_id
* rail\_type
  + used to find matching C:S railway type (NetInfo)
  + OSM source: railway
* Bridge
  + used to find matching C:S railway type (NetInfo)
  + OSM source: bridge
* geometry
  + actually just this one is mandatory
  + used to create C:S railway segment
  + In the created CSV file the column name will be: WKT

Open the rails\_rwo.csv in notepad or Excel and rename the columns accordingly.

## GeoSkylines Prefab output method (right Ctrl + P)

Prefab is basically a game object type such as road type, building type etc. When we are trying to create game objects from geographical objects, first, we need to match the type of the geographical object to the (most relevant) type of the game object. To do that properly, we need to know what prefabs are loaded in the game (it differs based on the mods loaded). To get this information, start a new game and then press right Ctrl + P. All loaded segment, building and tree types will be output to "c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Cities\_Data\output\_log.txt".

## Preparing trees data for GeoSkylines

GeoSkylines offers two methods to import trees:

* Trees Import from raster data (right Ctrl + T)
* Trees Import from vector data (right Ctrl + V)

In both cases we have to first prepare an input file: trees.png for raster import or trees\_rwo.csv for vector import. Trees are not recorded in OSM; therefore, we have to use different data sources. The data sources will vary for different parts around the world.

For Europe’s city areas, decent data source is Urban Atlas Street Tree layer. After registering, you can download data for your region from <https://land.copernicus.eu/local/urban-atlas/street-tree-layer-stl>

### Preparing trees.png in paint editor

GeoSkylines‘ method to import trees from raster data can be used for creating fictional tree layer as well. Just open new picture in paint editor of 1081 x 1081 pixels and add color where you want the trees to be. Only white color is considered to be an empty space, any other color will be considered as a tree (at that position a tree will be created).

### Preparing trees.png from raster tree layer

I didn’t have access to raster tree layer myself but if you do then you can just clip the raster by your bounding box (e.g. bbox\_wgs created in steps mentioned above) and save it as trees.png. If you can export only in TIF format then you need to use image SW to save the image as PNG.

### Preparing trees.png from vector tree layer

This example will work with Street Tree layer. This layer is stored in Shapefiles, geometries are polygons. After some symbology changes the creation of trees.png is similar to creating a map image mentioned above.

To create a trees.png from Street Tree layer:

1. Drag & drop ‚\_STL.shp‘ to QGIS
2. In the bottom right corner change the CRS of the QGIS canvas to UTM CRS fitting your area (e.g. EPSG: 32633)
3. Add bounding box area in UTM to QGIS (e.g. bbox\_utm33n.csv, creation steps mentioned above).
   1. Layer > Add Layer > Add Delimited Text Layer
   2. Settings:
      1. Geometry CRS: UTM matching your area (e.g. EPSG: 32633)
   3. Add and then close
4. Tick off the bbox\_utm layer so it’s not displaying in the map
5. Clip Street Tree Layer by bbox\_utm
   1. Vector > Geoprocessing Tools > Clip
   2. Settings:
      1. Input layer: STL
      2. Overlay layer: bbox\_utm
   3. Run and then close
6. In the layer list, double-click the Clipped layer and select Symbology > Simple fill
   1. Settings:
      1. Fill color: green (though green is natural for displaying greenery, the color doesn’t matter for GeoSkylines tree import method. Any color except white will be considered as a tree).
      2. Stroke style: No Pen
   2. Apply then close
7. Right click on layer bbox\_utm > Properties > Information
   1. In the first section, copy the Extent information to a temp notepad file
   2. Note: the extent has format west, south : east, north
8. In the layer list, untick all layers except Clipped
9. Export map as image:
   1. Project > Import/Export > Export Map to Image
   2. There’s an option to calculate the export extent from a layer. This however adds extra margin to the extent. We don’t want that.
   3. Instead copy the west, south, east and north coordinates from the extent of your bbox\_utm layer.
   4. Set dpi so the pixel resolution is EXACTLY 1081x1081.
      1. Alternatively, you can save at higher resolution and then use image SW to resize to 1081 x 1081
   5. Click Save
10. Copy the created map image to c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\

Example trees.png output:

A screenshot of a cell phone

Description automatically generated

Actually, for reasons at this moment unknown, I have trouble aligning the trees.png perfectly with other layers (heightmap, \_rwo.csv files). Therefore, I introduced the configuration parameters ImportTreesRasterOffsetX and ImportTreesRasterOffsetY. Using these parameters, I can shift the trees.png to align with other layers.

### Preparing trees\_rwo.csv from vector tree layer

This example will work with Street Tree layer. This layer is stored in Shapefiles, geometries are polygons. We will use randomly generated points within the polygons that will represent the trees.

To create trees\_rwo.csv from vector tree layer:

1. Drag & drop ‚\_STL.shp‘ to QGIS
2. In the bottom right corner change the CRS of the QGIS canvas to UTM CRS fitting your area (e.g. EPSG: 32633)
3. Add bounding box area in UTM to QGIS (e.g. bbox\_utm33n.csv, creation steps mentioned above).
   1. Layer > Add Layer > Add Delimited Text Layer
   2. Settings:
      1. Geometry CRS: UTM matching your area (e.g. EPSG: 32633)
   3. Add and then close
4. Tick off the bbox\_utm layer so it’s not displaying in the map
5. Clip Street Tree Layer by bbox\_utm
   1. Vector > Geoprocessing Tools > Clip
   2. Settings:
      1. Input layer: STL
      2. Overlay layer: bbox\_utm
      3. Run and then close
6. Add regular points:
   1. Vector > Research Tools > Regular points...
   2. Settings:
      1. Input extent: click the ... button on the right > Use Layer extent > select your bounding box layer or the Clipped layer
      2. Point spacing: 10 meters (you can make the distance smaller if you want denser greenery or make the distance bigger if you get over the game’s limit of 250,000 trees)
      3. Tick ‚Apply random offset to point spacing‘
      4. Output layer: variation of UTM
7. But now the points are everywhere, we want them just inside the polygons. We can do this by clipping the Regular points layer by the Clipped layer.
   1. Vector > Geoprocessing Tools > Clip
   2. Settings:
      1. Input Layer: Regular points
      2. Overlay layer: Clipped
   3. Run and then close
   4. Note: this process will take a long time to finish
8. New Clipped layer will be added to the layer list, but this time it will be points. Those are the representation of trees we want.
9. The resulting geometries of the Clipped points layer is MULTIPOINT. GeoSkylines works with POINT. Therefore, we simplify the geometry:
   1. Vector > Geometry Tools > Multiparts to Singleparts
   2. Settings:
      1. Input layer: Clipped points layer
   3. Run and then close
10. We also want the coordinates to be in WGS 84 (lat, lon). So, we reproject the layer.
    1. Vector > Data Management Tools > Reproject Layer
    2. Settings:
       1. Input layer: Single parts layer
       2. Target CRS: EPSG:4326 – WGS 84
    3. Run then close
11. Now we output the Reprojected layer.
    1. Right-click on the Clipped point layer > Export > Save Feature As
    2. Settings:
       1. Format: CSV
       2. File name: e.g. c:\temp\trees\_rwo.csv
       3. CRS: EPSG:4326 – WGS 84

There’s an alternative way to get the points within the polygons (Vector > Research Tools > Random points in Layer bounds) but this method takes even longer (by a lot) than the method of clipping regular points.

The preparation of the trees\_rwo.csv is cumbersome and lengthy, therefore, I rather used the raster method myself. But this import from vector data is an alternative that can be in a way more accurate than the raster method. Also, if you happen to have data source where you have point representation of the trees then this vector method is preferable.

Additionally, I’m thinking of updating the trees vector method in the future to calculate the points (as trees) from the given polygons directly in the methods code which would make the tree creation much easier. But at the moment use the steps described above if you want to use the tree import from vector data method.

## Preparing zoning data for GeoSkylines

For zoning you can use OSM data with the tag ‘landuse’ but a better choice would be a local dataset (if it’s available). There will be a lot of differences in the obtained local zoning datasets (data format, data structure, CRS, etc.) so there’s no point of listing specifying steps of preparing zones\_rwo.csv.

Anyway, you should familiarize with your local zoning dataset, decide which field best describes the type of zoning and how it can fit to the C:S zone types (see below for matching zone types). You might need to combine values from two fields into one in order to get desired zone type (that was my case). When you know what to output, then reproject the zoning layer to WGS 84 and save layer as zones\_rwo.csv. The required columns are id, zone\_type and geometry.

Example of zoning dataset:

A picture containing text, map

Description automatically generated

## Preparing matching files for GeoSkylines

When importing roads, railways, services and zones, corresponding matching files are required. It’s also important to configure the matching file properly to get the best results when generating data.

### rwo\_cs\_road\_match.csv

Matching file for road types must be named rwo\_cs\_road\_match.csv and stored in the folder c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.

You can edit the matching file with notepad or Excel but beware of programs messing up the formatting (e.g. Excel creating tabulator as a separator instead of a comma).

The format is very simple:

* Format: source\_data\_type,game\_object\_type (prefab name)
* Example: tertiary,Basic Road

To do the matching properly, you should know the types of source data and the types of game roads. Following examples are working with OSM road types. To get all the road types loaded in the game, use the GeoSkylines prefab output method (right Ctrl + P). What you want is the prefab name (not the title, which is the name visible in game).

The C:S base version has limited amount of road types. It is recommended to install additional mods (at least Network Extensions) to get more road types in the game so the matching is more accurate.

Example of simple road matching:

A screenshot of a cell phone

Description automatically generated

Rules:

* If you want to exclude certain type of roads, either don’t add it to the first column or leave the game type column (2nd) empty
* You can add new road type by combining the existing road type + „\_oneway“. That way you can e.g. assign ‚primary‘ to ‚Small Avenue‘ but ‚primary\_oneway‘ to ‚Oneway Road‘.
* If the „\_oneway“ type is misspelled, then the standard equivalent is used. E.g. ‚motozway\_oneway‘ will be ignored and matching ‚motorway‘ to ‚Highway2L2W‘ will be used instead.

Example of extended road matching:

A close up of text on a white background

Description automatically generated

If you are having issues, it’s most likely due to typos. Double check that both the source type names and game type names are spelled correctly.

### rwo\_cs\_rail\_match.csv

Matching file for rail types must be named rwo\_cs\_rail\_match.csv and stored in the folder c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.

You can edit the matching file with notepad or Excel but beware of programs messing up the formatting (e.g. Excel creating tabulator as a separator instead of a comma).

The format is very simple:

* Format: source\_data\_type,game\_object\_type (prefab name)
* Example: rail,Train Track

Matching rail types is very simple because there are not many different types of railways. Basically, in my tests I discovered only one rail type being useful for the purposes of the game and that is: ‚rail‘ to ‚Train Track‘ (maybe type ‚tram‘ could be useful as well).

Example of rail matching:

A screenshot of a cell phone

Description automatically generated

In this rail matching example all source data types except ‚rail‘ will be ignored. Source lines of type ‚rail‘ will be recreated into Train Track segments. The same rules as for road matching applies here as well (see rules above).

### rwo\_cs\_service\_match.csv

Matching file for services/amenity types must be named rwo\_cs\_service\_match.csv and stored in the folder c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.

You can edit the matching file with notepad or Excel but beware of programs messing up the formatting (e.g. Excel creating tabulator as a separator instead of a comma).

The format is very simple:

* Format: source\_data\_type,game\_object\_type (prefab name)
* Example: school,Elementary School

To get all the service building types loaded in the game, use the GeoSkylines prefab output method (right Ctrl + P). What you want is the prefab name (not the title, which is the name visible in game).

Example of service matching:

A screenshot of a cell phone

Description automatically generated

### rwo\_cs\_zone\_match.csv

Matching file for zone types must be named rwo\_cs\_zone\_match.csv and stored in the folder c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.

You can edit the matching file with notepad or Excel but beware of programs messing up the formatting (e.g. Excel creating tabulator as a separator instead of a comma).

The format is very simple:

* Format: source\_data\_type,game\_object\_type (prefab name)
* Example: RDLow, ResidentialLow

Matching zones is specific to the used zoning dataset. There are no functional zoning data in OSM therefore, local dataset – if it’s available - must be used instead. Every local dataset will most likely have its own types. You have to understand the source types and what is the appropriate matching to the C:S zones.

Example of zone matching:

A close up of a newspaper

Description automatically generated

## Configuring GeoSkylines import and export methods

File import\_export.txt offers several parameters that can adjust the behaviour of GeoSkylines’s import and export methods. File import\_export.txt must be stored in c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.

### MapName

Description: provides information about the area modelled (actually not used to adjust method’s behavior).

Example: Brussels

### CenterLatitude

Description: defines the latitude of the chosen mid-point of the modeled area

Example: 50.849591

### CenterLongitude

Description: defines the longitude of the chosen mid-point of the modeled area

Example: 4.372754

### ImportRoadsCoordMax

Description: Specifies the max coordinate (in both directions - positive and negative) for creating roads. Game area is 17280 x 17280, thus axes x a z range from -8640 to 8640. This represents 9x9 tile. If no value is defined, then the absolute 8640 will be used.

Example: 4800 (this represents the area of 5x5 tiles, game objects won't be created past this)

### ImportRailsCoordMax

Description: Specifies the max coordinate (in both directions - positive and negative) for creating railways. Game area is 17280 x 17280, thus axes x a z range from -8640 to 8640. This represents 9x9 tile. If no value is defined, then the absolute 8640 will be used.

Example: 4800 (this represents the area of 5x5 tiles, game objects won't be created past this)

### ImportBuildingsCoordMax

Description: Specifies the max coordinate (in both directions - positive and negative) for creating buildings (e.g. service buildings). Game area is 17280 x 17280, thus axes x a z range from -8640 to 8640. This represents 9x9 tile. If no value is defined, then the absolute 8640 will be used.

Example: 4800 (this represents the area of 5x5 tiles, game objects won't be created past this)

### ImportZonesCoordMax

Description: Specifies the max coordinate (in both directions - positive and negative) for creating zones. Game area is 17280 x 17280, thus axes x a z range from -8640 to 8640. This represents 9x9 tile. If no value is defined, then the absolute 8640 will be used.

Example: 4800 (this represents the area of 5x5 tiles, game objects won't be created past this)

### ImportTreesCoordMax

Description: Specifies the max coordinate (in both directions - positive and negative) for creating trees. Game area is 17280 x 17280, thus axes x a z range from -8640 to 8640. This represents 9x9 tile. If no value is defined, then the absolute 8640 will be used.

Example: 4800 (this represents the area of 5x5 tiles, game objects won't be created past this)

### ImportTreesRasterOffTolerance

Description: Sometimes the created map image is not exactly 1081 x 1081 but instead few pixels off (but still close enough). In this case you can specify the number of pixels the map image is off. The code will work only with map images that are within range (1081 - ImportTreesRasterOffTolerance) to (1081 + ImportTreesRasterOffTolerance).

Example: 1

### ImportTreesRasterOffsetX and ImportTreesRasterOffsetY

Description: If the tree map image (and the game trees generated based on the map image) doesn't align with other layers (roads, water basins) then it's possible to use ImportTreesRasterOffsetX & ImportTreesRasterOffsetY to move it around and align it with other layers.

Example: 100 (metres)

### ImportTreesRasterMultiply

Description: To make tree coverage denser or less dense (e.g. to avoid reaching the game limit 250 000 of trees created), you can use parameter ImportTreesRasterMultiply. The number specifies a step at which tree creation will be skipped (if number is negative) or an additional tree will be created (if number is positive).

Example A: -2 (every second tree creation will be skipped, i.e. the total number of trees will be divided by 2)

Example B: 1 (at every tree creation, new additional tree will be created, i.e. multiplying the number of tree by 2).

### ImportTreesTreeTypes

Description: For adding diversity in the tree creation process. List of TreeInfo/prefab (type of tree) names. For each tree creation, one TreeInfo is randomly selected from the provided list. Use the Prefab output method (right Ctrl + P) to find out all tree prefabs (types) loaded in the game.

Example: Tree2,Tree2variant,Tree3variant,Bush02,Corsican Pine 01,Green Tree 01

### ImportWaterWayTypes

Description: List of waterway types that the code will work with. Any other water way type will be ignored.

Example A: river

Example B: river, stream, canal

### ImportWaterWayDepths

Description: List of depths for water way types defined by ImportWaterWayTypes.

Example: 15, 10, 10 (assuming "ImportWaterWayTypes=river, stream, canal" then river basins will be 15 meters deep, stream and canal basins will be 10 meters deep).

Note: the depths must be exaggerated for the game water dynamics to work properly

### ImportWaterWayWidths

Description: List of additional widths for water way types defined by ImportWaterWayTypes. One "width" represents 16x16 meters on 1081 x 1081 grid.

Example: 1, 0 (assuming "ImportWaterWayTypes=river, stream" then river basins will width=2, stream basins will have width=1)

### ImportWaterDepth

Description: Defines the depth of standing water basins.

Example: 15

### ExportCoordsBox

Description: Xmin, Zmin, Xmax, Zmax game coordinates - only game objects within this bounding box will be exported. If not defined then bounding box is set to max (i.e. all game objects are exported).

Example: -250, -250, 1000, 1000

Note: avoid using 0 as 0 will be considered as failed attempt to set the coordinate (and thus set to max coordinate). Instead of 0 use -1 or 1.

## Import methods of GeoSkylines mod

**GeoSkylinesImport.ImportRoads():**

* Run by hotkey combo: right Ctrl + R
* Requires: roads\_rwo.csv, rwo\_cs\_road\_match.csv, import\_export.txt
* Description: loops over all road segments in roads\_rwo.csv, matches road types according to rwo\_cs\_road\_match.csv, creates game nodes and then game roads, names the roads according to geodata originals, creates a bridge if original data says bridge = yes, creates one way roads.
* Note: it’s better to call this method in actual game not the map editor. That way you can see the progress on the screen (segments appearing) and also the roads stick better to the surface. In map editor the roads are bit elevated.

**GeoSkylinesImport.ImportRails():**

* Run by hotkey combo: right Ctrl + L
* Requires: rails\_rwo.csv, rwo\_cs\_rail\_match.csv, import\_export.txt
* Description: loops over all rail segments in rails\_rwo.csv, matches rail types according to rwo\_cs\_rail\_match.csv, creates game nodes and then game rails.
* Note: C:S doesn’t use that many railways as in the real world. The amount of railways created by this method is therefore too much for C:S. Either filter out the geodata first or bulldoze it after creation.

**GeoSkylinesImport.ImportWaterBody():**

* Run by hotkey combo: right Ctrl + W
* Requires: water\_rwo.csv, import\_export.txt
* Description: loops over all records of standing water defined by a polygon in water\_rwo.csv, creates a bounding box around polygon, then every 5 meters within the bounding box calls Ray casting algorithm to find out whether point is within polygon or not. If yes, then lower terrain by defined value (variable ImportWaterDepth, see more details above).

**GeoSkylinesImport.ImportWaterWay():**

* Run by hotkey combo: right Ctrl + Q
* Requires: waterway\_rwo.csv, import\_export.txt
* Description: loops over all segments of water way in waterway\_rwo.csv, lowers terrain by defined value (variable ImportWaterWayDepths, see more details below) every 5 meters between the vertices of each segments.

**GeoSkylinesImport.ImportTreesRaster():**

* Run by hotkey combo: right Ctrl + T
* Requires: trees.png (1081 x 1081 resolution), import\_export.txt
* Description: loops over every pixel and for every non-white pixel it creates a tree. If variable ImportTreesRasterMultiply is defined, method adjust the number of trees created (see more details above). Method adds randomness into the position of the created trees.

**GeoSkylinesImport.ImportTreesVector():**

* Run by hotkey combo: right Ctrl + V
* Requires: trees\_rwo.csv, import\_export.txt
* Description: loops over all trees in trees\_rwo.csv and creates a tree.

**GeoSkylinesImport.ImportZonesArea():**

* Run by hotkey combo: right Ctrl + Z
* Requires: zones\_rwo.csv, rwo\_cs\_zone\_match.csv, import\_export.txt
* Description: sets zones to existing zone blocks (must be called after creating roads, this will create zone blocks as well). First it loops over every zone in buildings\_rwo.csv, finds zone blocks near the position of the building, matches the building type to a game zone (e.g. building type = house to zone = ResidentialLow) according to rwo\_cs\_zone\_match.csv and then assigns selected zone to the zone blocks. !!!!!!$$$$

**GeoSkylinesImport.ImportServices():**

* Run by hotkey combo: right Ctrl + S
* Requires: amenity\_rwo.csv, rwo\_cs\_service\_match.csv, import\_export.txt
* Description: loops over every amenity (service) in amenity\_rwo.csv, matches amenity type to a game service building according to rwo\_cs\_service\_match.csv and creates a service building.
* Note: the service buildings created by this method doesn't seem to work properly but still it might be handy to know where the services are. It can be bulldozed and then re-created manually.

**GeoSkylinesImport.ImportBuildings():**

* Requires: buildings\_rwo.csv, import\_export.txt
* Description: loops over every building in buildings\_rwo.csv, tries to calculate the right building rotation angle and creates the building.
* Note: this method is not used due to many complications. Difficult to calculate the right rotation angle, buildings are often to close to the roads, and mainly: creating buildings directly goes against the game logic where only zones are set. Although this can be overcome by mods, it was still quite unusable.

## Export methods of GeoSkylines mod

**GeoSkylinesExport.ExportSegments():**

* Run by hotkey combo: right Ctrl + G
* Requires: import\_export.txt
* Description: loops over all roads created in the game and exports them as GIS data (CSV format, geometry in WKT, any meaningful information about the road as attributes). Resulting roads\_cs.csv is stored in c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.

**GeoSkylinesExport.ExportBuildings():**

* Run by hotkey combo: right Ctrl + H
* Requires: import\_export.txt
* Description: loops over all buildings created in the game and exports them as GIS data (CSV format, geometry in WKT, any meaningful information about the building as attributes). Resulting buildings\_cs.csv is stored in c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.

**GeoSkylinesExport.ExportZones():**

* Run by hotkey combo: right Ctrl + J
* Requires: import\_export.txt
* Description: loops over all zones created in the game and exports them as GIS data (CSV format, geometry in WKT, any meaningful information about the zone as attributes). Resulting zones\_cs.csv is stored in c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.

**GeoSkylinesExport.ExportTrees():**

* Run by hotkey combo: right Ctrl + K
* Requires: import\_export.txt
* Description: loops over all trees created in the game and exports them as GIS data (CSV format, geometry in WKT, any meaningful information about the tree as attributes). Resulting trees\_cs.csv is stored in c:\Program Files (x86)\Steam\steamapps\common\Cities\_Skylines\Files\.