GIS714, Spring 2021
Final Project
Updating and Generalizing Blender Rendering for Tangible Landscape

Caitlin Haedrich

May 4, 2021

Introduction

Tangible Landscape is an open-source Tangible User Interface (TUI) that allows users to interactively visualize and modify landscapes [5]. The system, shown in Figure 1, uses a projector to project a digital map onto a physical model of the landscape, a scanner (such as the Xbox Kinect) to sense changes in the landscape, GRASS GIS to perform geospatial computation and data visualization, and Blender to render scenes from the modeled landscape. Rendering scenes with Blender is one of the newer features of Tangible Landscape and opens many novel potential applications in landscape architecture, education, and science engagement, communication, and visualization. The renderings can even be viewed in Virtual Reality using goggles such as Oculus Rift. However, as a new feature, there are several areas that could be updated and generalized.

The initial Blender add-on was built by Dr. Payam Tabrizian and Dr. Anna Petrasova [7, 6] specifically for designing a park. It uses the point cloud from the scanner to create a surface then covers the resulting surface with a grass material. Then, it translates vectors representing waterbodies, trees and trails into 3D objects on the terrain. Users can set a viewpoint using a pin on the physical model and create flythroughs and walkthroughs using a marker or laser pointer. While effective, this initial build has not been updated since it's original release. It uses an old release of Blender's API (2.79) and cannot run on the most current Blender release (2.92 at time of writing) [1]. Additionally, the add-on was built to work at a specific scale of landscape. Rendering settings are not generalized enough to handle areas of varying scales. For example, trees are imported at a fixed height which would be incorrectly scaled for regions smaller or larger than the park scene.

In this project, I work towards a new Blender add-on that runs in the new Blender API (2.8+) and that can be used at many different scales. This generalized add-on builds off Dr. Tabrizian and Dr. Petrasova's work but is my own code. Given the substantial learning curve required and limited scope of a semester-long project, the resulting code will not be a polished add-on, ready for publication. Instead, this project will lay the foundation for a future add-on by creating a script that renders a Digital Elevation Model (DEM), water surface, buildings and a tree (to scale) in Blender. The user can give the script any DEM, water raster, building shapefile and a location point for a tree and the script will render a scene. Code, sample data, installation

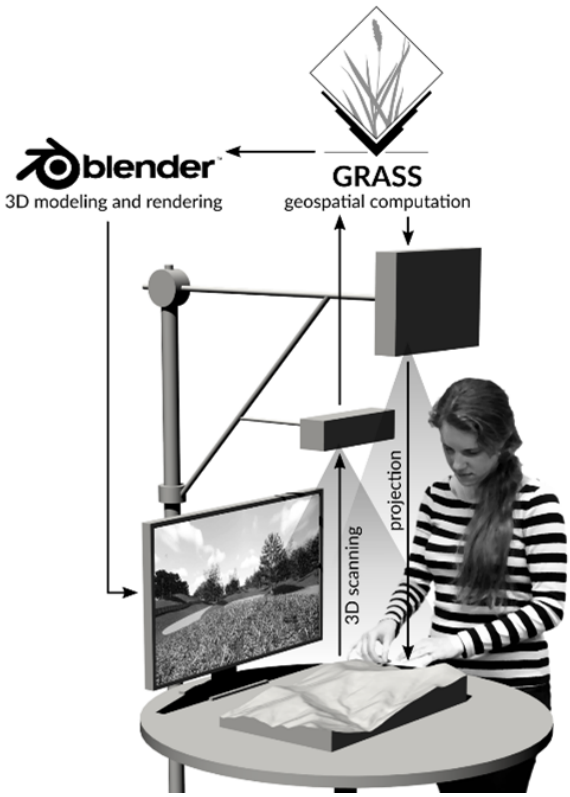


Figure 1: Diagram of Tangible Landscape. Illustration adapted from [6]
.

instructions and a copy of this report are found on github (`github.com/chaedri/GIS_blender_immersion`).

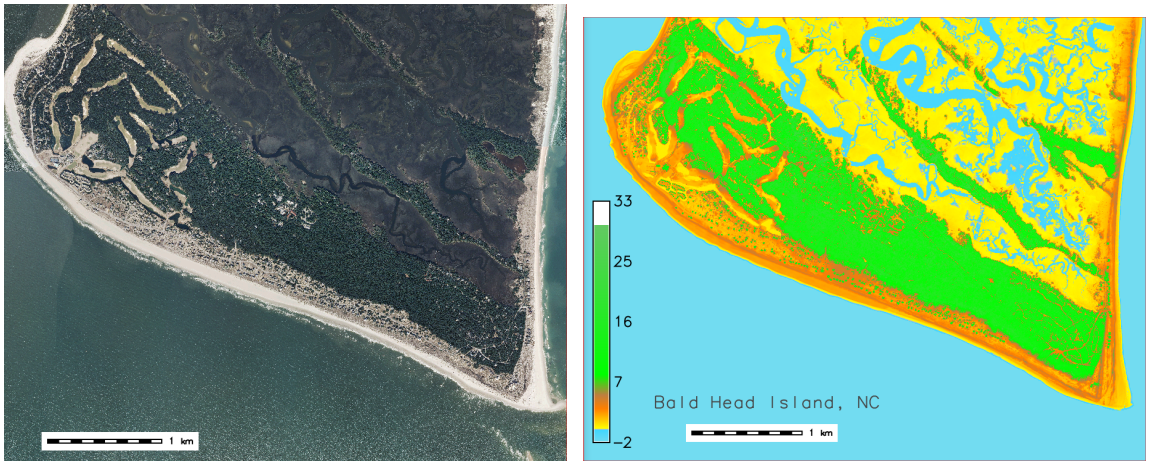
Methods

Sample Data

For sample data, I use a dataset from Bald Head Island, NC. The small island, shown in Figure 2, is part of North Carolina's outer banks. Conveniently, a dataset of the island's southern tip has been used in other projects by the GeoForAll Lab and is readily available as a GRASS GIS database. The dataset includes an elevation time-series, buildings, and an orthophoto of the area.

Software Environment

To set up the environment, I installed Blender 2.92 [2]. As downloaded, Blender does not handle geospatial data. Therefore, we also install the BlenderGIS add-on [3] which allows users to load DEMs, rasters and shapefiles through the GUI interface and through a Python API. The original plug-in [7] also required BlenderGIS.



(a) Orthophoto from 2014.

(b) Digital Surface Model from 2014.

Figure 2: Southern tip of Bald Head Island, North Carolina.

The original Blender addon for Tangible Landscape used a shared folder to pass data between GRASS GIS and Blender. GRASS GIS exported files to the folder and Blender watched for new or updated files to render. We adopted this structure since the GRASS GIS export workflow is already in place. The resulting software architecture is shown in Figure 3. The watch folder contains the DEM, water raster, building shapefile, location point for a tree, the tree object and an optional orthophoto (as a .png). The georeferenced files in the Watch folder should have the same coordinate reference system (CRS) and use projected coordinates (units of meters). The raster resolution should match and have 1 meter resolution. The configuration file contains the name and file path for each of the inputs along with any other setting for the main script. The user opens a new, blank Blender file, deletes the default cube and opens the main script in the scripting window. From there, users can edit and run the script. The main script reads the input variables from the configuration file and imports data from the watch folder. The rendered scene appears in the rendering pane. In the next section, we discuss the structures and features of the main script.

Script Components

The script begins by importing the DEM. BlenderGIS provides a raster import option that allows the raster to imported as a 3D terrain where cell values are the z-coordinate of the cell. Then, we create a material for the terrain using the `texture_terrain` function. This function has a flag, `z_color`, that allows the user to specify whether to use an orthophoto (`z_color = False`) or shade the terrain by elevation (`z_color = True`). It should be noted that the orthophoto needs to be a non-georeferenced '.png' image. A '.png' can be easily exported from GRASS GIS using the 'r.out.png' module. Materials in Blender are made by connecting nodes. Each node provides a texture or transforms an existing texture. The node trees used the create the materials are shown in Figure 4.

After importing and texturing the terrain, the water raster is imported and textured. Although the sample water raster provided in the github repository contains 1 where there is water and 0 where there is not, this is unnecessary. An empty raster would suffice since the entire water raster is covered with a water material. The water's visibility is based on it's height above the terrain. Where the elevation of the land is

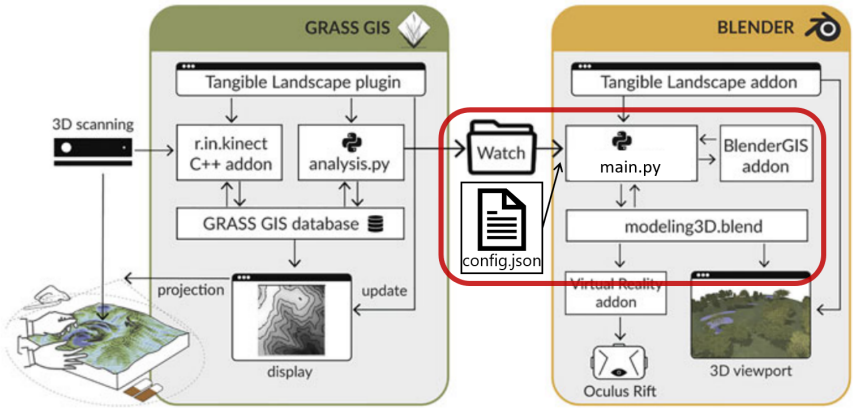


Figure 3: Diagram showing the software architecture of the Tangible Landscape Blender addon. Diagram adapted from [6].

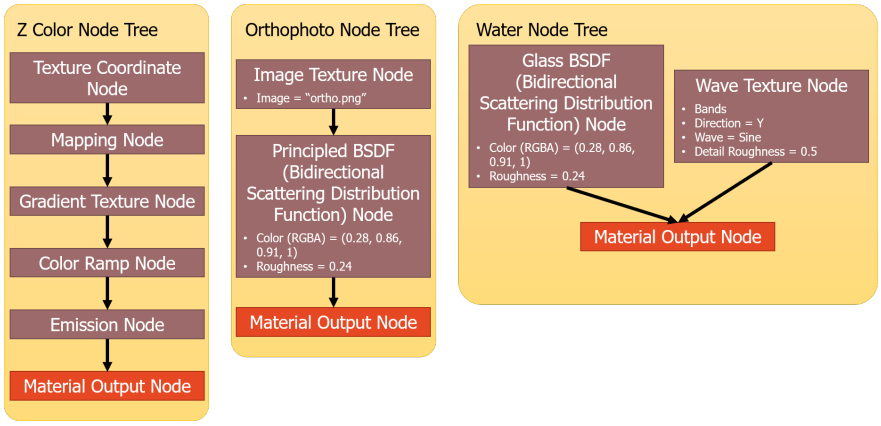


Figure 4: Material node trees for the terrain with either the elevation shading (left) or orthophoto overlay (middle) and the water (right).

above sea level, the terrain occludes the water. This allows us to easily simulate flooding. Using the Flood variable in the configuration file, users can translate the water layer vertically to simulate inundation during a storm surge or flood event.

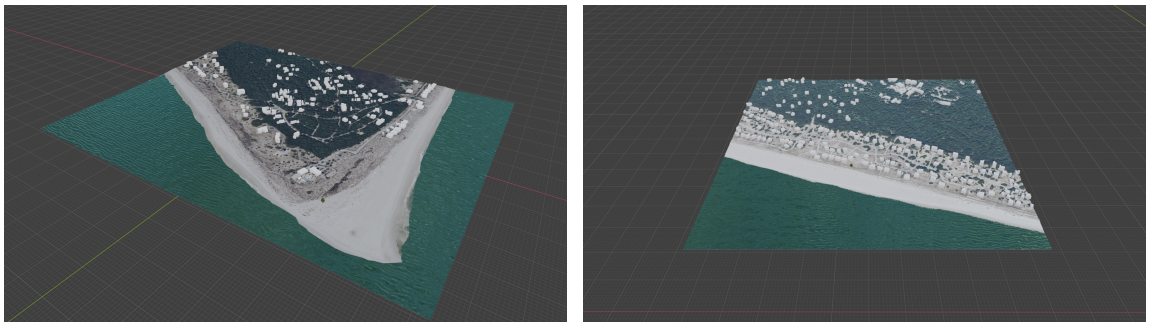
Next, the buildings are imported from a shapefile. BlenderGIS shapefile import function includes options to set the buildings base elevation and height (extrusion). However, to use this, the shapefile must contain height and elevation fields. These fields were not included in included in the original Bald Head Island dataset. Therefore, I sampled the base elevation from the 2014 DEM and the building height from a 2014 DSM provided that were included in the dataset. The buildings currently render without any material or texture coating.

Finally, the script imports an American Linden tree. The .obj tree file was sourced from the original Blender addon which had a library of vegetation objects for the park. The American Linden tree object is

the largest file we import at 27 MB. The American Linden tree is an logical choice of tree for this scene since it is native to North Carolina and grows to be 60 to 80 feet tall (18-24 meters) [4]. To scale the tree to a reasonable height (say 18 meters), we use the Blender's transform function. The Bald Head Island dataset is in a projected coordinate system so it uses meters. Additionally, the dataset has a resolution of 1 meter. Thus, the units of the scene, when imported are also in meters and scaling the tree to 18 corresponds to a actual height of 18 meters. The watch folder also contains a shapefile with a point at the desired tree location. We use Blender's transform function again to move the tree to the point.

Results

Renderings of two different sections of Bald Head island are shown in Figure 5. Rendering the whole region is possible but with 1 meter resolution, the script takes inconveniently long to run. Therefore I selected two smaller sub-regions to demonstrate the rendering. Using two smaller regions also demonstrates the location independence of the script. From the renderings shown in Figure 5, users can zoom in and simulate views from different points on the scene.



(a) Southern tip of Bald Head Island.

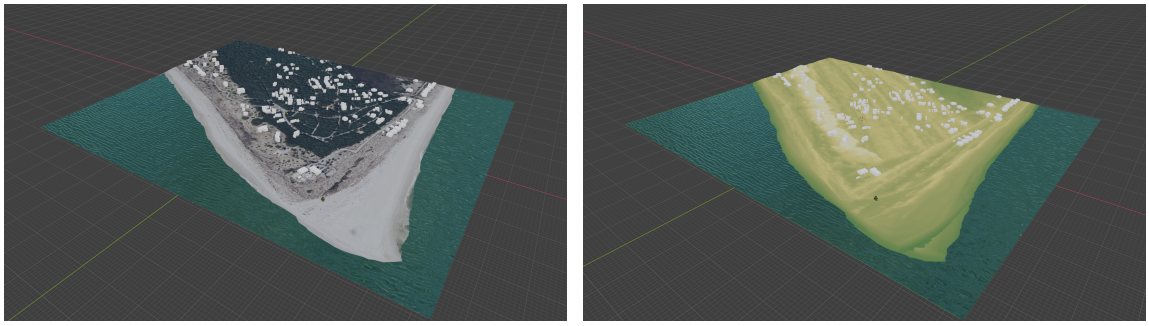
(b) Southern coast of Bald Head Island.

Figure 5: Renderings of two different areas of Bald Head Island, NC with an orthophoto overlay.

The two renderings in Figure 5 use the orthophoto for the terrain texture. The scene can also be rendered using shading by elevation (`z_color=True`), as shown in Figure 6.

Using the Flood variable, set in the configuration file, we can simulate inundation flooding due to storm surge or a change in sea level. The flood level is set in units of meters and translate the water layer vertically, super-imposing it over the terrain. Figure 7 shows the result of raising the water level by 1, 2, and 5 meters. As we can see, a 5m storm surge would be cataclysmic for Bald Head Island.

Though the script accomplishes the objectives of the project, it takes a significant amount of time to run, especially given the final objective is for real-time interaction as required for Tangible Landscape. The scene shown in Figure 7 is created from a 6.7 MB DEM and a 15.2 MB orthophoto, resulting in a runtime of about a minute on a personal laptop. To test the relationship between scene size and runtime, I created several subsets of the scene shown Figure 5b, beginning with it's original extent and DEM of size 5.1 MB and decreasing to a scene with a DEM of 0.3 MB. The orthophoto is about twice the size of the DEM which could contribute significantly to the total runtime. To test the effect of using an orthophoto, I repeated the scenes with z-color shading instead of an orthophoto. The results are shown in Figure 8. We can see that the relationship between scene size and runtime is linear and that adding an orthophoto overlay does not



(a) Orthophoto (b) Elevation Shading

Figure 6: Renderings with both elevation materials: an orthophoto (left) and elevation shading (right). We can also clearly see the 18m-tall American Linden tree in the center of the elevation shading rendering.

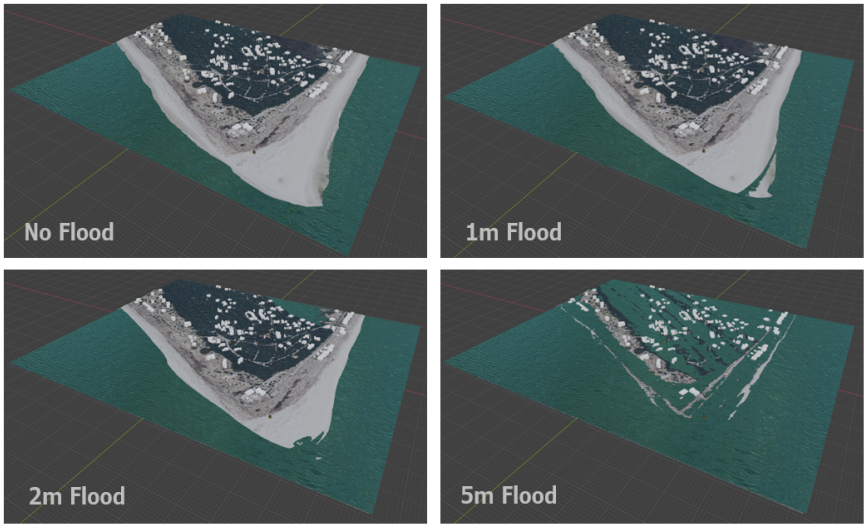


Figure 7: Flooding scenarios created by translating the water layer vertically.

have a significant impact on total runtime. A small scene with a few houses with a bit of beach takes about 7 seconds to render.

Discussion

While successful in rendering a terrain, water surface, buildings and a non-geospatially referenced object (the American Linden tree), this script will need several major improvements before it can be used as a Blender addon.

First, planting one tree at a time is not an effective way to create a forest. The original add-on [7] allowed users to plant trees using patches; users provided polygons representing patches of forest then

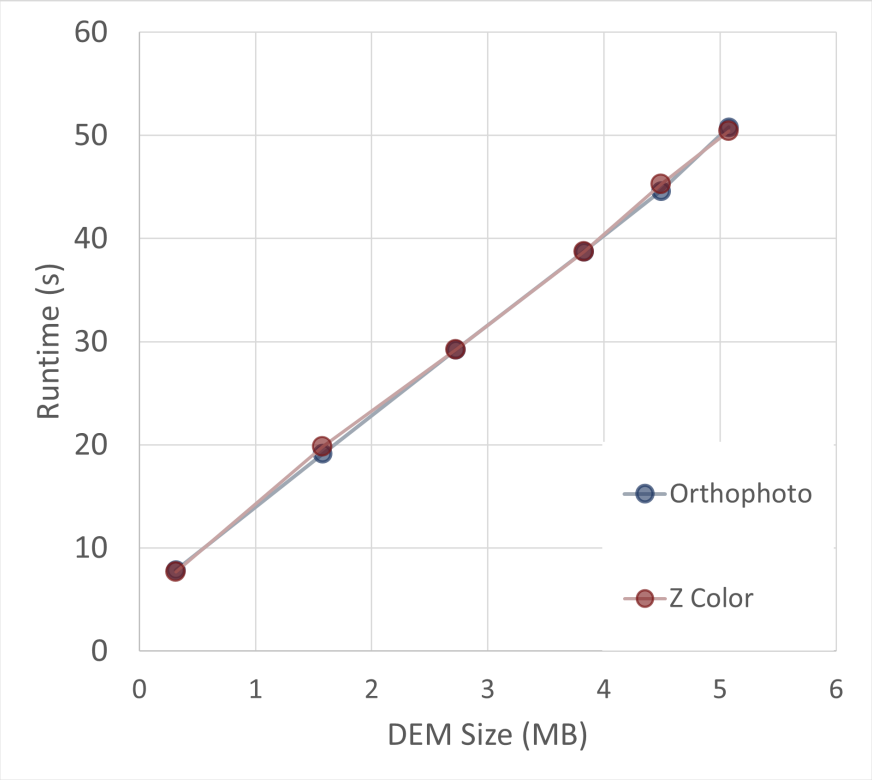


Figure 8: Script runtime with different scene sizes and terrain materials.

Blender populated the patches with trees at a set density. Future versions of the script should adopt this method instead of creating trees one-by-one.

In this project, we rendered buildings as rectangles without any materials. The result of is not realistic. In future editions, adding a material to cover the buildings and possibly changing their shape to include a roof would make the rendering more realistic.

One of the largest drawbacks of using a flat water surface that extends through the whole is scene is that the script could not handle water at different elevations. For example, if a scene had two ponds at different elevations, this methods would no longer work with one continuous sheet of water at a fixed elevation. Likewise, this methods does not work well for simulating how the ridge of dunes between the beach and homes on Bald Head island might protect the homes. If we increase the sea level, the water will also rise behind the dunes even though they might act as a dam.

Finally, the script is also too time consuming for real-time modification. As shown in Figure 8, the runtime for scene created from a 5 MB DEM is about 50 seconds. In order to make the scene update in real time, consideration will have to given to identifying the computationally intensive parts of the script and reducing the runtime..

References

[1] Blender Foundation. Blender 2.80: Python api changes, 06 2019. URL `https://wiki.blender.org/wiki/Reference/Release_Notes/2.80/Python_API`.

[2] Blender Foundation. blender.org - home of the blender project - free and open 3d creation software, 2019. URL `https://www.blender.org/`.

[3] domlysz. domlysz/blendergis, 04 2021. URL `https://github.com/domlysz/BlenderGIS`.

[4] Educational programs of the Kentucky Cooperative Extension Service. American linden tree, 04 2021. URL `https://www.uky.edu/hort/sites/www.uky.edu.hort/files/pages-attachments/tiliaamericprint.pdf`.

[5] NCSU GeoForAll Lab. Tangible landscape, 2016. URL `https://tangible-landscape.github.io/`.

[6] A. Petrasova, B. Harmon, V. Petras, P. Tabrizian, and H. Mitasova. *Tangible Modeling with Open Source GIS*. Springer International Publishing, second edition, 2018. doi: 10.1007/978-3-319-89303-7. URL `https://link.springer.com/book/10.1007/978-3-319-89303-7`.

[7] P. Tabrizian, A. Petrasova, B. Harmon, V. Petras, H. Mitasova, and R. Meentemeyer. Immersive tangible geospatial modeling. *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 10 2016. doi: 10.1145/2996913.2996950. URL `https://doi.org/10.1145/2996913.2996950`.

Appendix 1: Installation Directions

1. Install Blender (`www.blender.org/`) 2.8 or later and the BlenderGIS addon (`github.com/domlysz/BlenderGIS`).

2. Download and unzip the github repository (`github.com/chaedri/GIS_blender_immersion`).

3. Create a folder in the base directory of your computer called `'TL\_coupling'`. Move the 'Watch' folder from the downloaded repository to the `'TL\_coupling'` folder. The path to the 'Watch' folder should now be `'C:\TL\_coupling\Watch\'`. This is the shared folder that GRASS GIS will write it's output to and Blender will watch for data to render. If the watch folder path does not match `'C:\TL\_coupling\Watch\'`, change the path in the `'config.json'` file to the correct.

4. Open a blank template in Blender 2.8 or later and delete the default cube. Switch to the Scripting layout. Open `'main.py'` in the scripting window.

5. In the upper right hand corner of the rendering window, toggle the viewport shading to 'Material Preview'.

6. At the beginning of `'main.py'`, change the `'config_file'` variable to the path to repository. Press the 'play' button. A scene from NC Bald Head Island will render.