MIT 16.4400 Computer Graphics Project Proposal

David Berthiaume

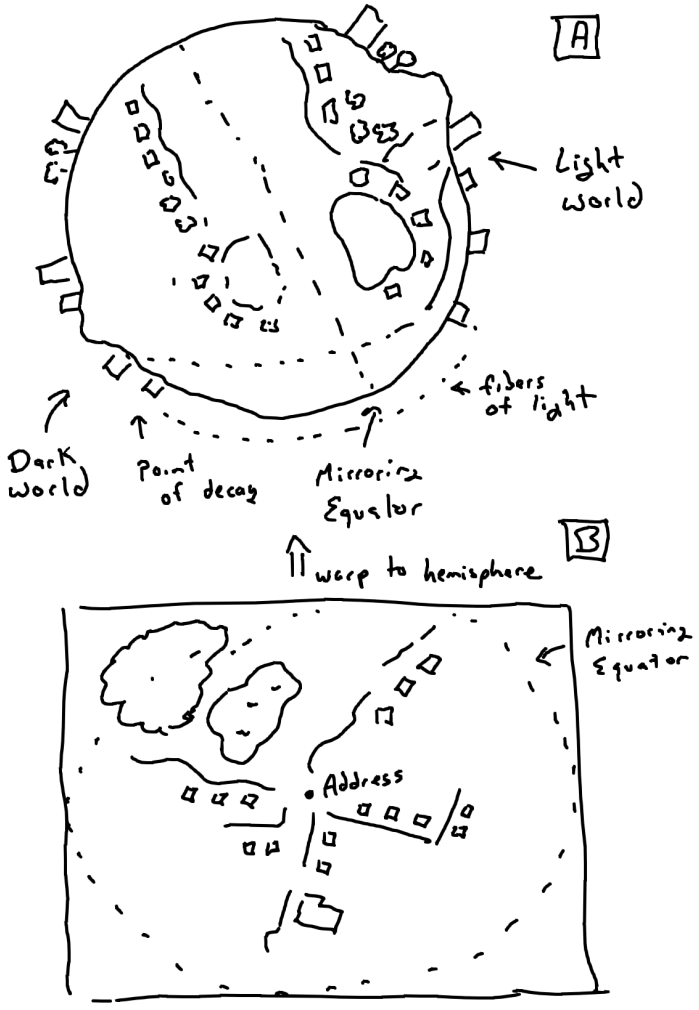
November 12, 2023

Contents

1 Background

One of my most vivid memories from my very early childhood is staring intently into a painting on the wall at the town library. This painting depicted an aerial view of the town where I grew up. While the layout of the town was true to life, the colors and details were starkly different. The painting represented a fascinating mix of real-world data and artistic interpretation. This project will attempt to create such an artwork from a unique geometric perspective using ray tracing.

Figure 1: Conceptual drawing of proposed rendering



2 Project Description

For this project, I will render a small planet using a custom ray tracer from scratch built in python. See figure 1. The terrain, roads, buildings, trees, and

other objects, time permitting, will all be based on real-world data, derived from various Massachusetts geospatial data sources. A disc of radius roughly 1km of this data, projected onto a hemisphere will be used to create the geometry for each half of the planet.

There will be a light side of the planet and a mirrored dark side of the planet. The dark side will contain a reflection of the geometries on the light side with some artistic changes. Fibers of light that look like power lines providing illumination will connect the buildings from one side of the planet to the other.

A user will enter an address for any property in Massachusetts. A planet will be rendered with a central focus on that location. The final result will be a high-quality rendered image with antialiasing.

3 Project Outline

This project will consist of the following steps. Given the scope of this project, I started early, and steps 1, 2 and 3 have already been completed. Significant progress has been made on step number 4 and 5.

1. Gather, process, clean, index, and organize around 800 GB of geospatial data for the entire state of Massachusetts, including LiDAR-derived bare-earth elevation data, land cover and use data, building footprints, imperviousness, and tree cover.

2. Geocode the input address and produce a hemisphere of real-world data from the above repository by clipping and projecting each data set.

3. Create a unified, coregistered raster at a resolution of 25cm/pixel from the above clipped data sets.

4. Place trees and other objects based on sampling of the land use types through gridded placement, perlin noise, and then jittering. Extrude 2D buildings to create 3D shapes. Project these onto the hemisphere.

5. Tessellate the heightmap into two hemispheres at 25cm resolution and assign materials.

6. Create a ray-tracer from scratch in Python and numpy to render this planet using displacement mapping or tessellation. Use simple geometries to render the trees and other objects.

7. Reflect the geometry across the equator to generate the dark side of the planet. Apply material and geometric changes to this dark hemisphere.

8. Create the fibers of light connecting the buildings and implement indirect illumination.

4 Screenshots of Current Progress

Figure 2: Cleaning, processing, registering, and clipping of land-use data

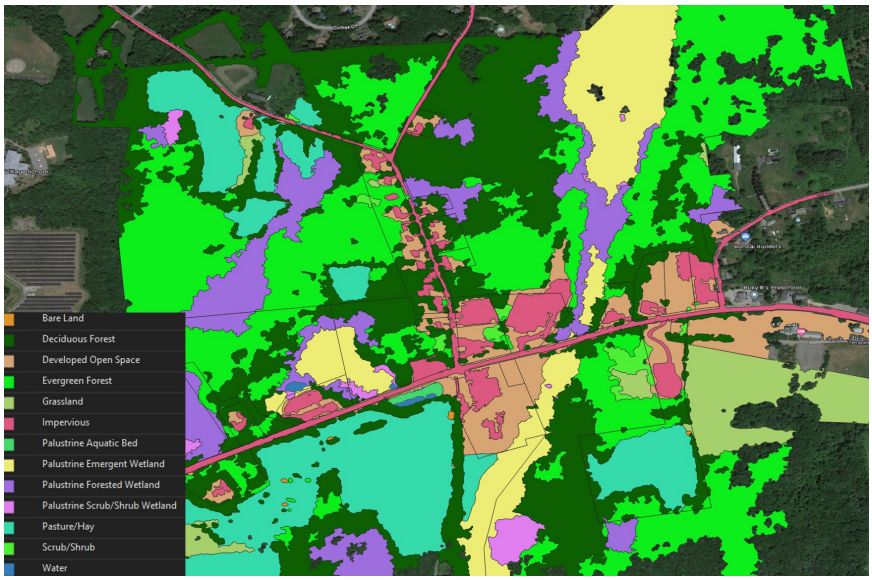


Figure 3: Building footprints

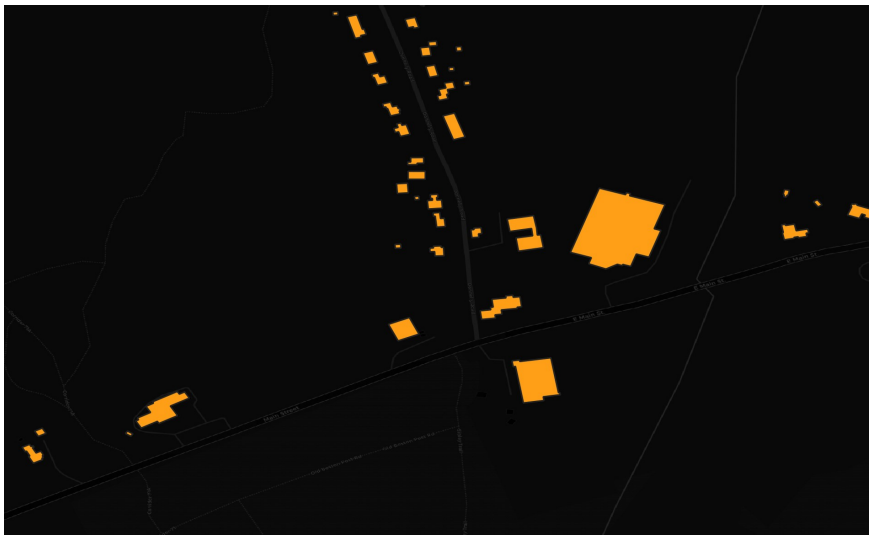


Figure 4: Tessellated bare-earth LiDAR-derived elevation

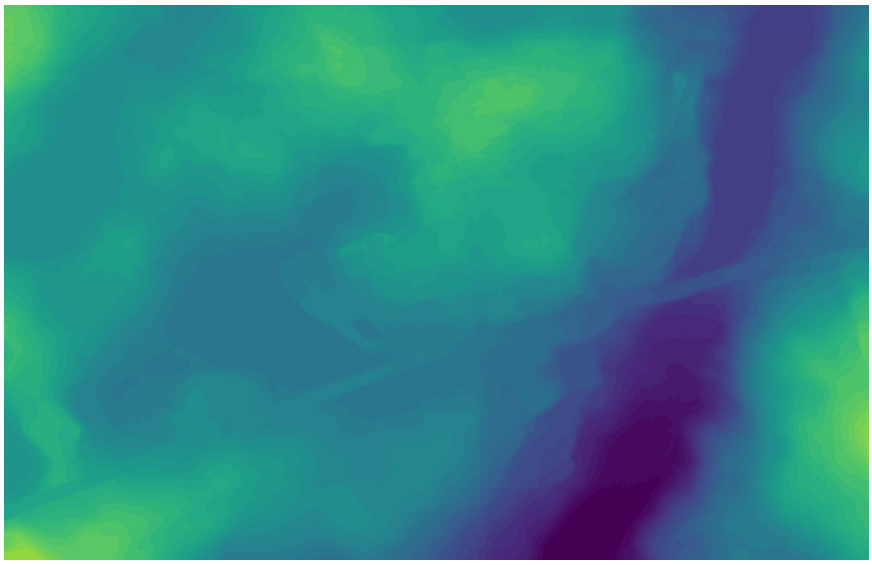


Figure 5: Dynamic gridded placement with Perlin noise and jittering for tree placement in forested landcover regions

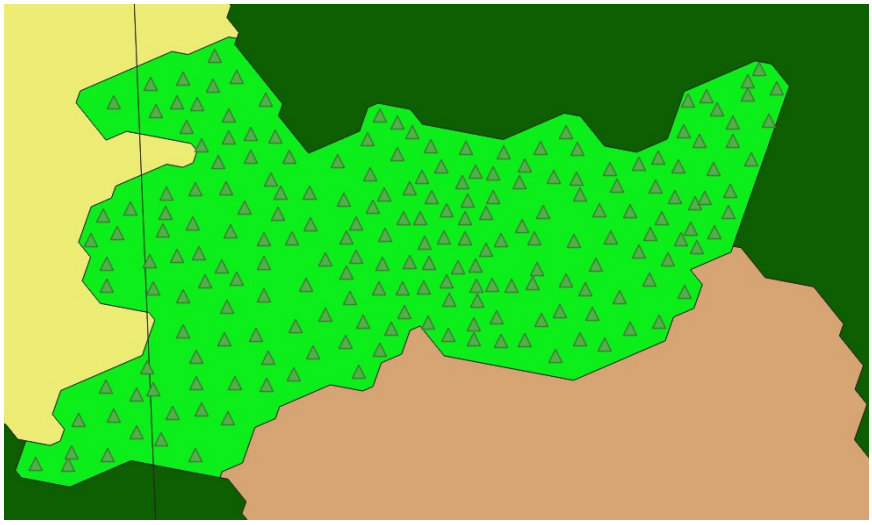
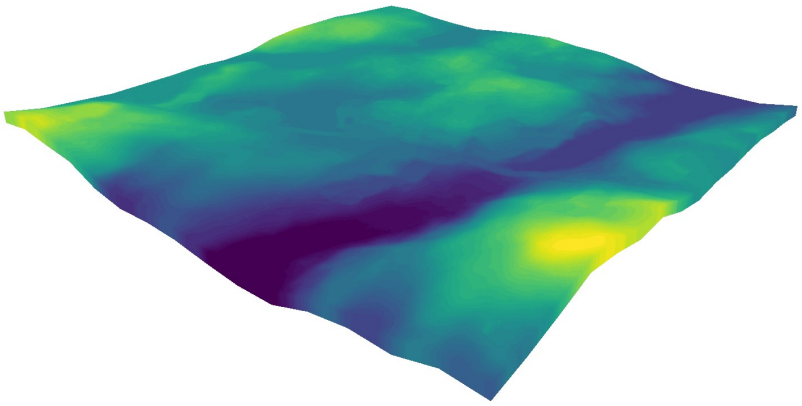


Figure 6: Tessellated bare-earth elevation 3D view (not yet shown as a hemisphere)



5 References

- Book: The ray tracer challenge: a test-driven guide to your first 3D renderer Buck, Jamis (Access through Harvard library)
- MassGIS Bureau of Geographical Information

  `https://www.mass.gov/orgs/massgis-bureau-of-geographic-information`
- Ray tracing from scratch

  `https://raytracing.github.io/books/RayTracingInOneWeekend.html`
- Extracting 3D Bare-Earth Surface from Airborne LiDAR Data

  `https://ieeexplore.ieee.org/document/4562112` (Access through Harvard)
- Perlin noise generator for Python

  `https://pypi.org/project/perlin-noise/`
- Ray tracing and global illumination

  `https://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1100&context=ojii_volumes`