MIT 6.4400 COMPUTER GRAPHICS FINAL PROJECT CHECK-IN

DAVID M. BERTHIAUME

Abstract. This document summarizes my current progress with the final project for MIT 6.4400 Computer Graphics.

Contents

1. Project Summary

The goal of this project is to create a ray tracer completely from scratch using Python and Numba and to render an artistic miniature planet based on real-world data.

2. Progress Made

The core ray tracer and algorithms to produce most of the geometry for the planet have been completed. The following lists the features that have been implemented.

- *Gathering and processing data for Massachussets.* This was a lot of work, but it went relatively quickly. I gathered LiDAR data, building footprints, land cover data, land use data, tree cover data, and imperviousness data and created a framework to turn each of these (polygon or raster) layers into a unified dataset. I then implemented a routine which for a given address clips these layers into a unified raster (EPSG:32619 projection) at a given resolution for sampling during the geometry and material construction of the planet.
- *Creating a ray tracer from scratch.* I significantly underestimated how much work this was going to be, especially to make the ray-tracer performant enough to render tens of millions of triangles in under a minute at 2000x2000 resolution.

Date: December 6, 2023.

  - I implemented many of the standard algorithms for ray tracing including Phong lighting.
  - The most difficult part was making it performant enough to handle tens of millions of geometries in a reasonable amount of time to allow for rapid iteration and debugging. Taking advantage of the the particular geometry of this scene, with its nearly spherical shape and symmetry, I implemented a custom octtree(ish) data structure that allows me to recursively perform axis-aligned bounding box intersections optimally to reduce the number of expensive intersections.
  - In many areas of my rendering, I implemented perlin noise. This is used in the terrain, material types, and to the (spherical) background to make it more interesting.
- *Geometry generation.* I converted a projected set of data into two hemispheres and stitched them together. To accomplish this, I turned the heightmap into a mesh and then warped this into a hemisphere. I then mirrored this mesh around the equator to create the 'dark side' of the planet. I place trees based on the tree cover layer and land cover types, and generate building geometries. I sample from the land use and land cover data layers to create custom materials for each location of the planet. For example, water is blue, reflective, and shiny. Certain elements are procedurally generated such as building heights and transitions from one land type to another.

3. Challenges Faced and Next Steps

Overall, this project has been *significantly* more work than I expected it to be.

Now that the core algorithms are implemented, I plan on implementing better lighting, shadows, reflections, artistic effects, and perhaps a simple asteroid ring around the center of the planet to help hide the obvious symmetry.

I wanted to do a live presentation, but I have a conflict with a final exam at Harvard. I will record a video instead.

4. GitHub Repository

I have made the GitHub repository for my ray tracer public. It can be found here: `https://github.com/davidberth/MITCGProject`.

5. Screenshots of Current Progress

Figure 1. A rendering of '236 Lake Ave Newton, MA 02460' (no shadows and only basic lighting implemented so far)

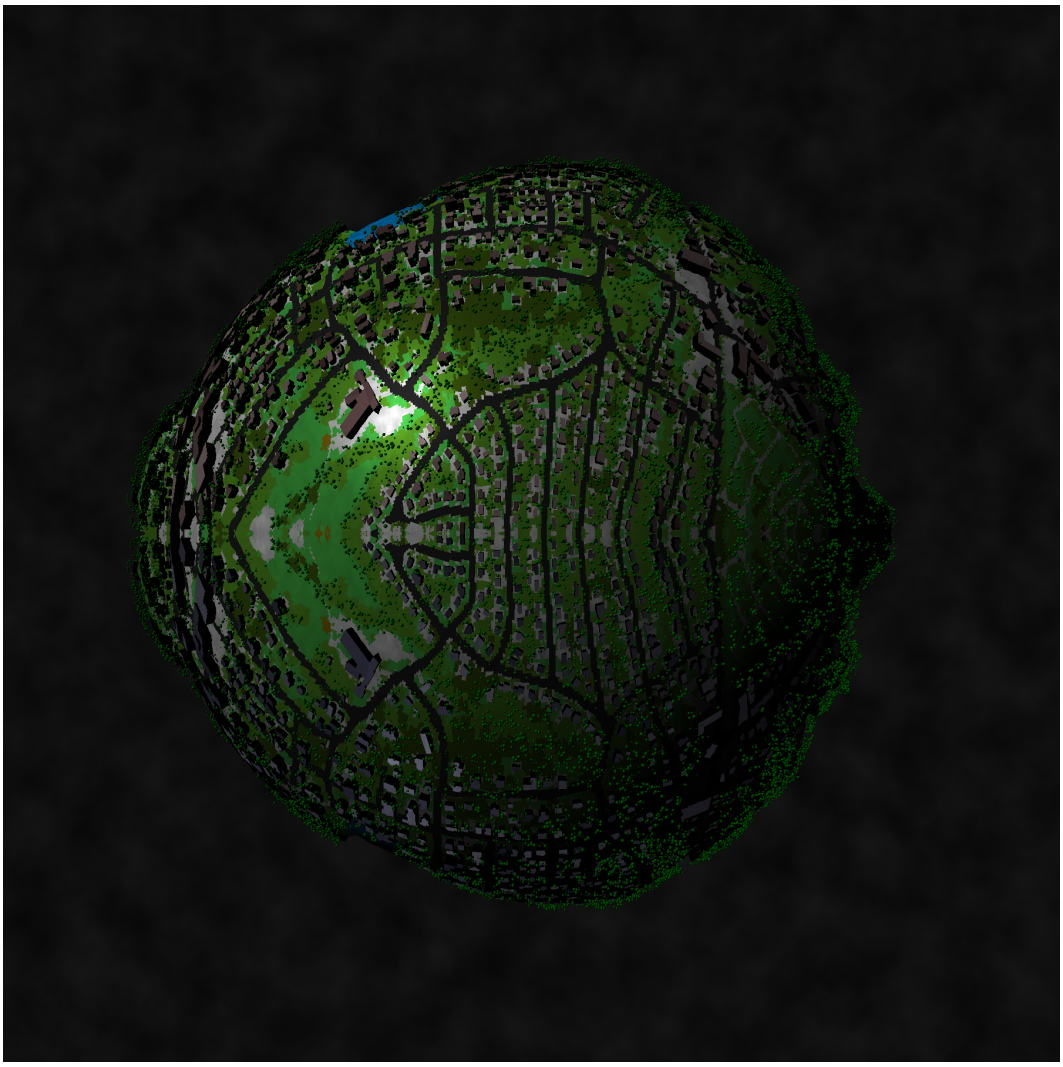


FIGURE 2. A rendering of '62 Jacqueline Rd, Waltham, MA 02452'

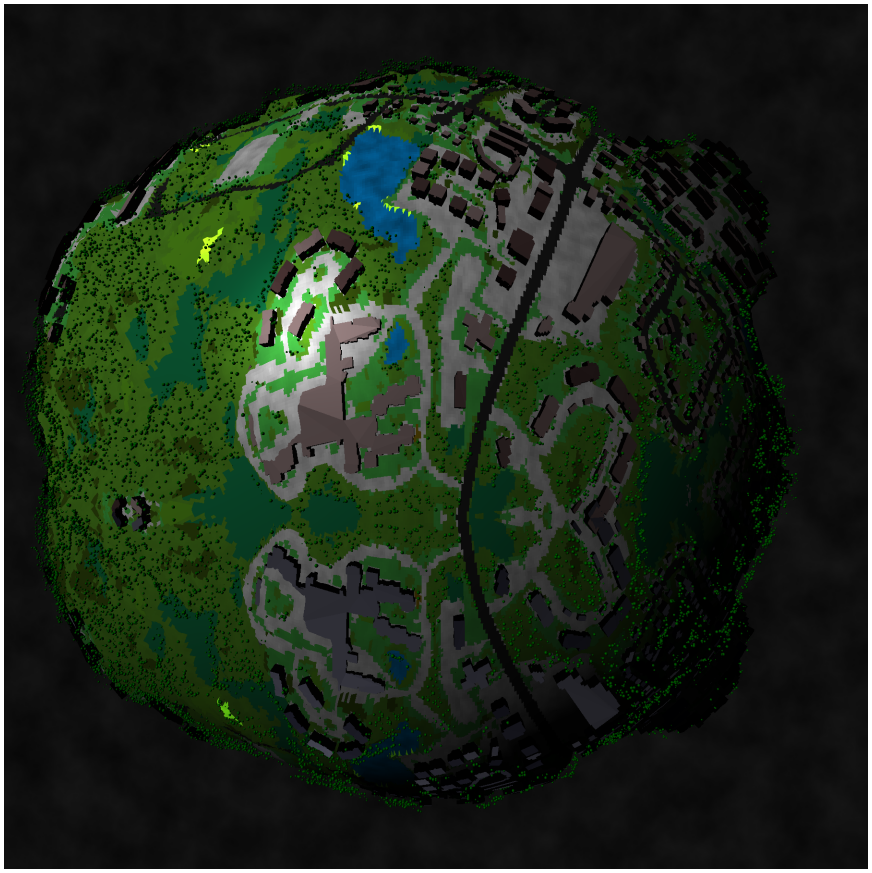
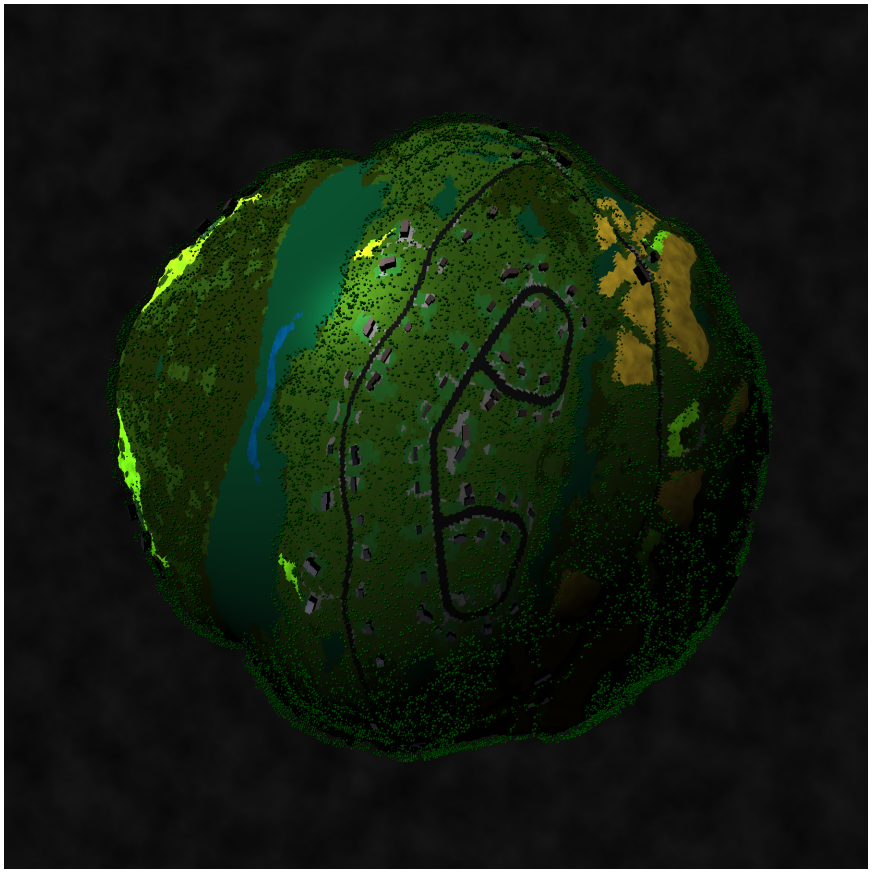


FIGURE 3. A rendering of a remote region in Spencer, MA



6. Land Cover Types Implemented

Figure 4. A list of land cover types that have been implemented so far

Class Number	Class Name
2	Impervious
5	Developed Open Space
6	Cultivated Land
7	Pasture/Hay
8	Grassland
9	Deciduous Forest
10	Evergreen Forest
12	Scrub/Shrub
13	Palustrine Forested Wetland (C-CAP)
14	Palustrine Scrub/Shrub Wetland (C-CAP)
15	Palustrine Emergent Wetland (C-CAP)
16	Estuarine Forested Wetland (C-CAP)
17	Estuarine Scrub/Shrub Wetland (C-CAP)
18	Estuarine Emergent Wetland (C-CAP)
19	Unconsolidated Shore
20	Bare Land
21	Open Water
22	Palustrine Aquatic Bed (C-CAP)
23	Estuarine Aquatic Bed (C-CAP)

7. 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