Earth Heightmap Project

August 15, 2025

Summary

This project renders an interactive, textured Earth using legacy OpenGL (GLUT / GLU) and applies a **heightmap-based displacement** to the sphere so mountains, trenches and other relief features appear on the globe. It also provides a runtime menu to swap colormap textures and displays a small legend ("bar") overlay when available. The code uses `stb_image.h` to load image files.

The implementation emphasizes correctness and robustness: consistent texture/heightmap orientation, smoothed height sampling to avoid spikes at the poles and seams, and proper normals computed after displacement so lighting looks good.

---

1 Features

- Equirectangular color texture mapped onto a sphere.
- Heightmap (grayscale) interpreted as elevation and used to displace sphere vertices along normals.
- Smooth normals computed via central differences to preserve shading quality.
- Runtime right-click menu to switch the color texture among several files.
- Automatic loading and display of `<basename>_Bar.png` overlay legend for color textures (if the file exists).
- Interactive controls: free rotation (left-drag), zoom (mouse wheel), autorotate, reset, heightmap toggles and orientation fixes.
- Runtime toggles for correcting orientation mismatches: flip vertical/horizontal, transpose (swap u/v), 180° longitude offset, smoothing on/off.

2 Files (typical)

- `main.cpp` — full program source (OpenGL + GLUT + `stb_image`).
- `stb_image.h` — single-header image loader.
- `earth.png` — default color texture.

- `earth_elevation_grayscale.png` — grayscale heightmap (white = high).
- `DayTemp.png`, `Rainfall.png`, `SeaSurfaceTemp.png`, `LeafAreaIndex.png` — example alternative color textures.
- `DayTemp_Bar.png`, `Rainfall_Bar.png`, etc. — optional legend/scale images for the corresponding textures.

3 How it works (technical overview)

Mesh

A latitude–longitude mesh is generated with `stacks` (latitude) and `slices` (longitude). For each (u, v) in $[0, 1] \times [0, 1]$ (where u maps to longitude and v maps to latitude, with $v = 0 =$ north pole and $v = 1 =$ south pole) the code computes:

1. Sample height $h(u, v)$ from the heightmap (bilinear interpolation).
2. Convert (u, v) to spherical coordinates (θ, ϕ) and an un-displaced direction (x, y, z).
3. Displace radius:
$$\text{rad} = \text{baseRadius} + (h - 0.5) \cdot 2 \cdot \text{heightScale},$$
so a heightmap value of 0.5 means no displacement.
4. Position = direction × rad.

Normals

Normals are computed per-vertex using **central differences** in texture space: sample displaced positions at $(u + \Delta u, v)$ and $(u - \Delta u, v)$ and at $(u, v + \Delta v)$ and $(u, v - \Delta v)$. The cross product of these tangent vectors yields a stable normal suitable for lighting.

Height sampling orientation

Heightmap sampling supports transformations to fix orientation mismatches:

- `transpose` — swap u and v when sampling (for datasets stored transposed).
- `flip` (U / V) — horizontal/vertical flips of the heightmap sampling.
- `uOffset` — add 0.5 (180° longitude) to fix half-rotation mismatches.
- `smoothing` — average center + neighbors to reduce spikes and seams.

These are provided as runtime keyboard toggles so you can align the heightmap to the color texture without modifying the image files.

Texture swapping & overlays

A GLUT popup menu (right-click) lists available color textures. Selecting a texture loads it as the sphere's diffuse map. The program automatically attempts to find a corresponding bar image named `<basename>_Bar.png` and displays it as an overlay (bottom-right) if found.

4 Controls (runtime)

- Left-drag: manual free rotation.
- Mouse wheel: zoom in/out.
- `'a'`: toggle autorotate on/off.
- `'p'`: toggle autorotate axis perpendicular-to-current.
- `'r'`: reset orientation and zoom.
- `'m'`: toggle heightmap displacement on/off.
- `'['` / `']'`: decrease / increase height exaggeration.
- `'v'`: toggle vertical flip of heightmap sampling.
- `'u'`: toggle horizontal flip of heightmap sampling.
- `'t'`: transpose heightmap sampling (swap u and v).
- `'o'`: add 180° longitude offset ($u \mathrel{+}= 0.5$).
- `'s'`: toggle smoothing of height sampling.
- Right-click: open texture menu to pick a new color texture (auto-loads corresponding `_Bar.png` overlay if present).
- ESC: quit.

5 Build & run

Make sure you have GLUT/GLU and OpenGL development libraries installed, and `stb_image.h` in the same directory.

```
g++ main.cpp -o earth -lGL -lGLU -lglut -lm -std=c++11
```

Run (defaults):

```
./earth earth.png earth_elevation_grayscale.png
```

You may pass a different color texture and/or heightmap as command-line arguments.

6 Troubleshooting & orientation checklist

If geographic features do not align between the color texture and heightmap (e.g., Antarctica appears at the top, Africa looks upside-down, Himalayas flattened):

1. Press `'t'` (transpose) — many heightmaps are saved transposed relative to the colormap.
2. Press `'v'` to flip vertically if the image rows are inverted.
3. Press `'u'` to flip horizontally if left/right is reversed.
4. Press `'o'` to add a 180° longitude offset if the prime meridian is centered incorrectly.

5. Toggle `'s'` smoothing to reduce pole spikes; then adjust height scale (use `']'` / `'['`) to visualize mountains clearly.

If you see seams or spikes at the poles, increase smoothing or reduce `heightScale`. If the terrain looks too flat, increase `heightScale` with `']'`.

7 Performance & tuning

- Lower `meshStacks` and `meshSlices` to reduce vertex count for faster frame rates (e.g., 64×128).
- Convert mesh generation and rendering to VBOs/VAOs for better performance and to avoid re-uploading large client arrays.
- Cache the heightmap as a float array (instead of reading bytes each sample) if you plan to sample it heavily or implement higher-quality filtering.
- Use a shader-based pipeline (GLSL) if you want GPU displacement (tessellation or vertex shader) or normal mapping for more realistic lighting.

8 Possible future improvements

- GPU-based displacement and normal calculation (GLSL + VBOs) to offload work from CPU.
- Add a water layer at fixed radius (sea level) so oceans remain flat while land is displaced.
- Add dynamic LOD: higher resolution near the camera-facing hemisphere, lower elsewhere.
- Implement automatic orientation detection: test combinations of flips/transpose/offset and pick the one where southernmost latitudes contain mostly ocean (heuristic).
- Add UI controls (on-screen GUI) to change textures, toggles and sliders instead of keyboard/menu.

9 Attribution & license

- Image loading: `stb_image.h` by Sean Barrett (public domain / MIT-compatible).
- This code is provided as-is. Feel free to reuse or adapt it; if you publish derived work, a quick mention is appreciated but not required.