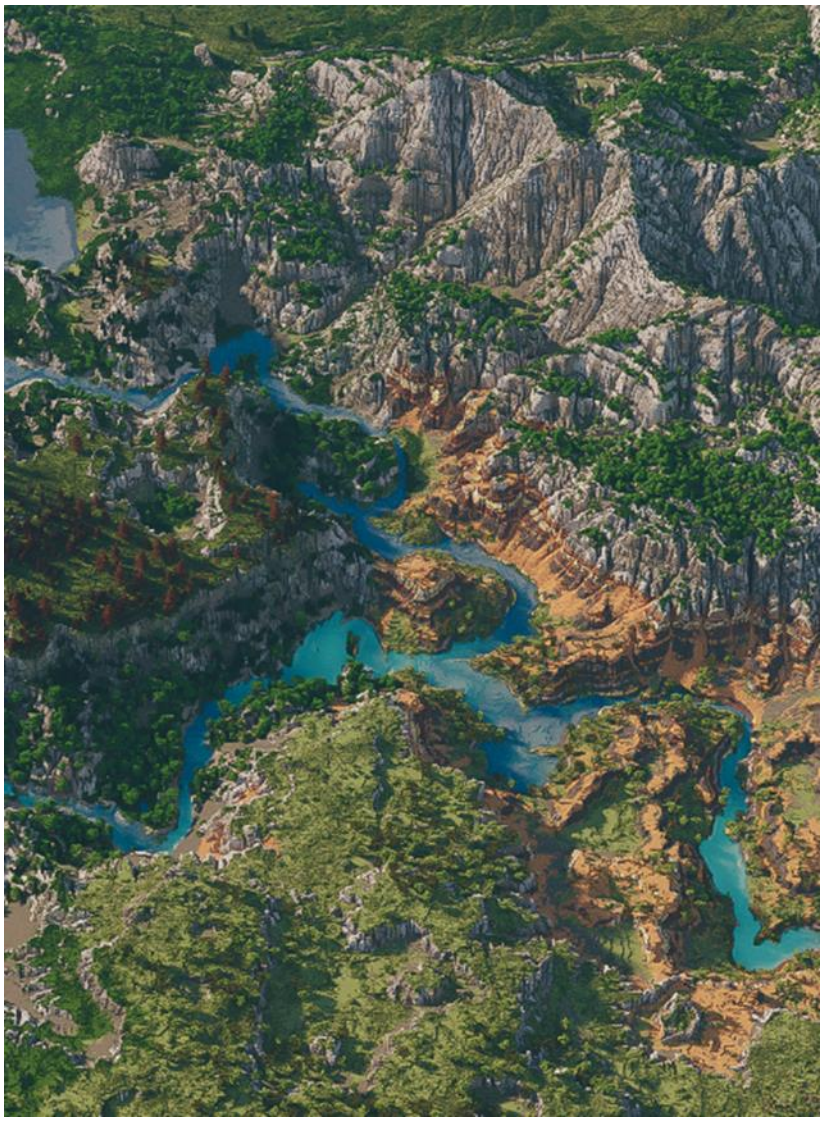


Sector-Based Terrain Generation

EXIT REPORT

Joseph Bourque
Joel Davis
Lior Vansteenkiste

CS 354 Final Project | 11 / 24 / 2016

What is “sector-based” terrain generation?

The short answer, is that sector-based terrain generation takes advantage of one of the fundamental rules of calculus that curves can be used to approximate sums and sums can be used to approximate curves. With this in mind, our team is trying to develop a system which is able to take in data in the form of cube-like sectors and then generate interpolated curves from the data to form a smooth gradient. For this project, we plan to use Perlin Noise to generate a temperature gradient in the xz-plane as well as a height gradient in the positive y direction. After successful terrain generation, our goal is to interpolate over the silhouette of the object using the vertices of the cubes touching the object to form a control point sheet for which a curved surface can be formed.

User Manual

KEY COMMANDS

C - Change from Orbital to FPS Mode

- You start in orbital

A - FPS Pan Left, Orbital Swivel Left

D - FPS Pan Right, Oribtal Swivel Right

W - Zoom In

S - Zoom Out

UP - FPS Pan Up, Orbital Swivel Up

DOWN - FPS Pan Down, Orbital Swivel Down

LEFT - Roll Left

RIGHT - Roll Right

MOUSE LEFT - FPS Drag and Pan, Orbital Rotate

MOUSE RIGHT - Drag and ZoomA – FPS pan left, Orbital

1 – Display Topography

- No temperature, water, or textures

2-Display flat map

- You can use **t** to toggle textures on/off here as well

3-Display cube map

- displays each data point as a cube, with altitude (exploded view of option 2)

4 – Display smooth map

- Adds water back in and creates a smooth terrain map

9 – Regenerate Map

- Generates a new map (in case you don't like the one it gave you)

t – toggle texture on / off

- If texture is not present, temperature is used to determine the color of the sector where Blue = $0°F$ and Red = $100°F$.

o – toggle overlay on / off

- Toggles temperature overlay which is raised a few pixels above the terrain map and was implemented to view temperature data while in textured mode (note: still works in non-textured mode)

FEATURES

- **Temperature to Color Function**
  - Converts a temperature between 0°F and 100°F to a color where Blue is 0° and Red is 100°.
- **Diamond Square Temperature / Altitude Gradient**
  - Creates a random array of point each with varying altitude and temperature. Utilizes the Diamond Square algorithm to generate a grid of points.
- **Surface Silhouetting**
  - Based on an idea from Calculus that curves can approximate sums. Generates a triangle mesh using the positions of the original sectors, and generates a mesh between adjacent sectors that closely models the original cube map.
- **Surface Subdivision Smoothing (Originally B-splines)**
  - Used surface subdivision to better approximate the surface of our tri-mesh and eliminate obvious cube surfaces. While this does deviate from the original cube map slightly, efforts were taken to conserve as much of the original map-data as possible
- **Texture Mapping**
  - Implemented texture mapping based on altitude and temperature
  - TEMP Texture
    - 0~20* Ice
    - 20~32* Snow
    - 33~40* Frozen Plain
    - 40~60* Forest
    - 60~75* Plains
    - 75~85* Jungle
    - 85~90* Scrub
    - 90~100* Desert
  - ALT Texture (Overwrites Temp)
    - If Below Sea && Temp 0~20* ICE
    - If Below Sea && Temp 28~40* Rock beach
    - If Below Sea && Temp 50~60* Mud
    - If Below Sea && Temp > 60 Sand
    - If Above 4 Mountains only (treeline)
    - If Above 4 && Temp < 32* Ice Mountain
    - If Above 4 < 20* Snow

- **Shader Effects**
    - Wrote shader programs to handle switching between the different terrain modes and for handling the movement of the water
    - Also, implemented Phong illumination and implemented smooth shading using interpolation of normals.
    - Water shader added to give moving water effects

ORIGINAL DEVELOPMENT PLAN

- Stage 1: Flat Ground -- Complete!!!
    - Nothing special, generate a flat field composed of cubes
- Stage 2: Perlin Noise Gradient
    - Use Perlin noise to generate random hot/cold spots and then create a gradient between these spots for non-linear generation
- Stage 3: Uneven Ground
    - Add height to our terrain, how to do this?
        - Could use more Perlin noise
        - Could use sign/cos functions
        - Could just generate random cliffs
- Stage 4: Uneven Ground + Altitude Gradient
    - Decrease temperature based on height using a gradient, the higher you go the colder it gets
- Stage 5: LOD Silhouetting
    - Create something that looks a little better, by finding the silhouette of our object in 3D and then interpolating over the vertices to form a tri-mesh
    - Trimesh Generation Notes
        - Check 1: Single cube at lower altitude than all other cubes --> Solution: treat it as if it is also the same height
        - Check 2: Is corner --> Solution: For every direction, orthogonal to the direction we are interpolating in where there is no block check for a block to the back of the current block and create a triangle between the current block and the back block, up to 4 of these triangles could exist
        - Storage: Create a map, Key value = vec2(x, z) position of block; data value = block itself
        - Have each cube only look at its own neighbors, as neighbors are interpolated over, mark them as discovered
- Stage 6: B-Spline Smoothing
    - Terrain still not smooth, what if instead of generating a tri-mesh we create a curved surface using B-Splines?
- Stage 7 (Stretch Goal): Texture Mapping
    - If we have time instead of colors use textures

DEVELOPMENT REPORT

- Stage 1: Flat Ground – Complete
  - Nothing special, generate a flat field composed of cubes
- Stage 2: ~~Perlin Noise~~ Diamond Square Gradient – Complete
  - Use Perlin noise to generate random hot/cold spots and then create a gradient between these spots for non-linear generation
  - Another student told me Diamond Square is not Perlin noise so I changed this from Perlin Noise to Diamond Square as that was what I had in mind when I created the project.
- Stage 3: Uneven Ground – Complete
  - Add height to our terrain, how to do this?
    - Could use more Perlin noise
    - Could use sign/cos functions
    - Could just generate random cliffs
- Stage 4: Uneven Ground + Altitude Gradient – Complete
  - Decrease temperature based on height using a gradient, the higher you go the colder it gets
- Stage 5: LOD Silhouetting – Complete
  - Create something that looks a little better, by finding the silhouette of our object in 3D and then interpolating over the vertices to form a tri-mesh
  - Trimesh Generation Notes
    - Check 1: Single cube at lower altitude than all other cubes --> Solution: treat it as if it is also the same height
    - Check 2: Is corner --> Solution: For every direction, orthogonal to the direction we are interpolating in where there is no block check for a block to the back of the current block and create a triangle between the current block and the back block, up to 4 of these triangles could exist
    - Storage: Create a map, Key value = vec2(x, z) position of block; data value = block itself
    - Have each cube only look at its own neighbors, as neighbors are interpolated over, mark them as discovered
- Stage 6: ~~B-Spline Smoothing~~ Surface Subdivision Smoothing – Complete
  - Terrain still not smooth, what if instead of generating a tri-mesh we create a curved surface using B-Splines?
  - Decided to go with surface sub-division vs. B-spline smoothing due to the fact it fit much more cohesively with the way we were doing the rest of our project.
- Stage 7 (Stretch Goal): Texture Mapping – Complete
  - If we have time instead of colors use textures
  - Implemented texture mapping but not texture blending so textures still appear a bit blocky.

DIVISION OF LABOR

- **Joseph Bourque**
    - Diamond Square Algorithm
    - Key Events
    - Shader Code Implementation
        - Phong Illumination
        - Water effects
        - Temperature – Color
    - Texture Mapping
    - Created edges to water
- **Joel Davis**
    - Silhouetting of cube map
    - Added edges and bottom of map
    - Interpolation of normals for smooth shading
- **Lior Vansteenkiste**
    - Surface Subdivision Shading
    - Code cleanup + QA

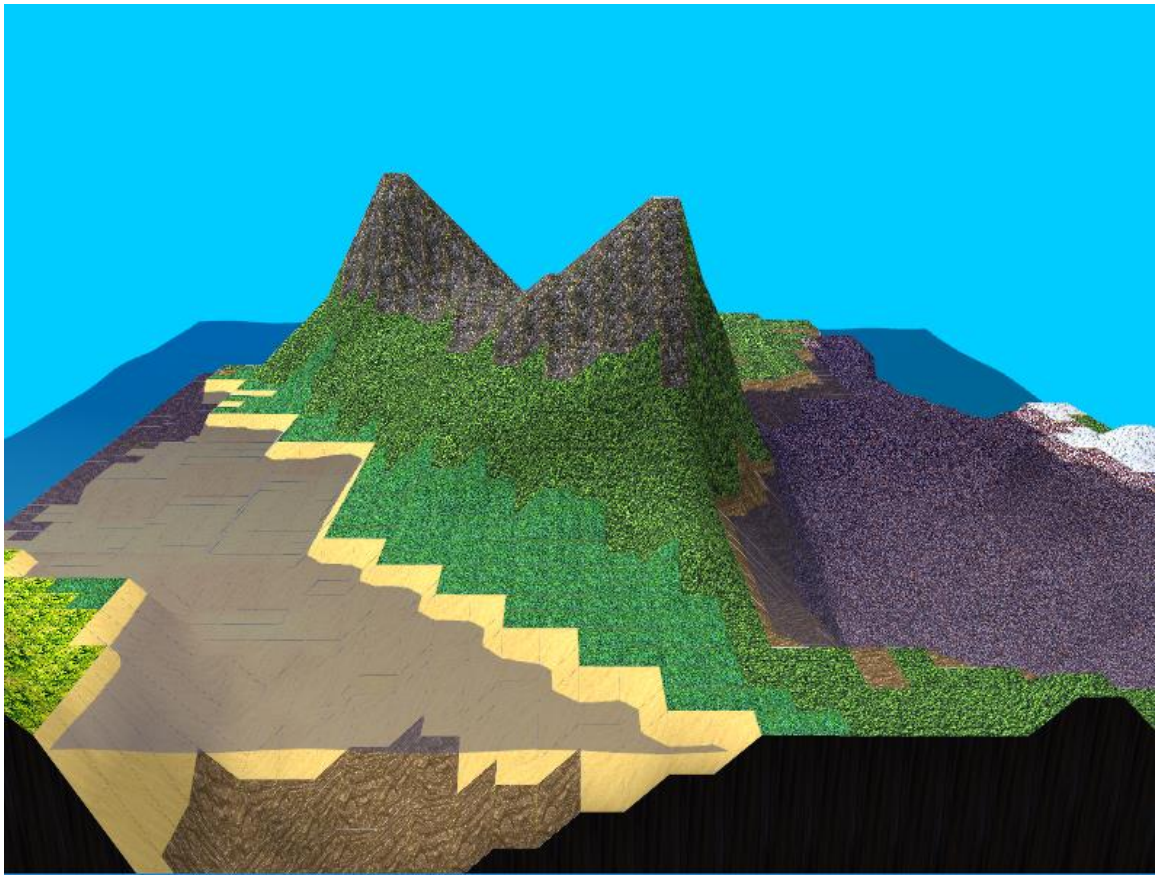
REFERENCES

Diamond Square Algorithm - https://en.wikipedia.org/wiki/Diamond-square_algorithm

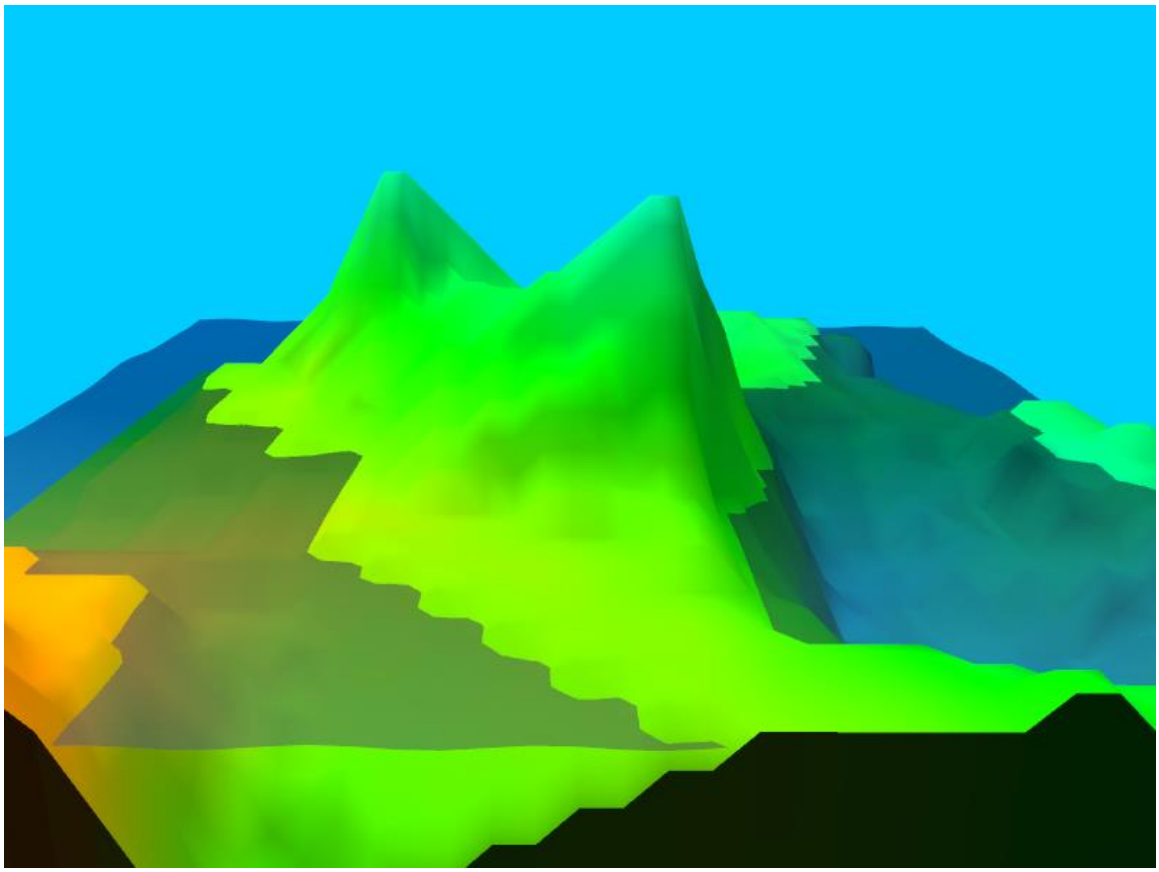
Idea Behind Project (Sums Approximate Curves) - http://brownmath.com/calc/ulsum.htm

StackOverflow – mainly for debugging

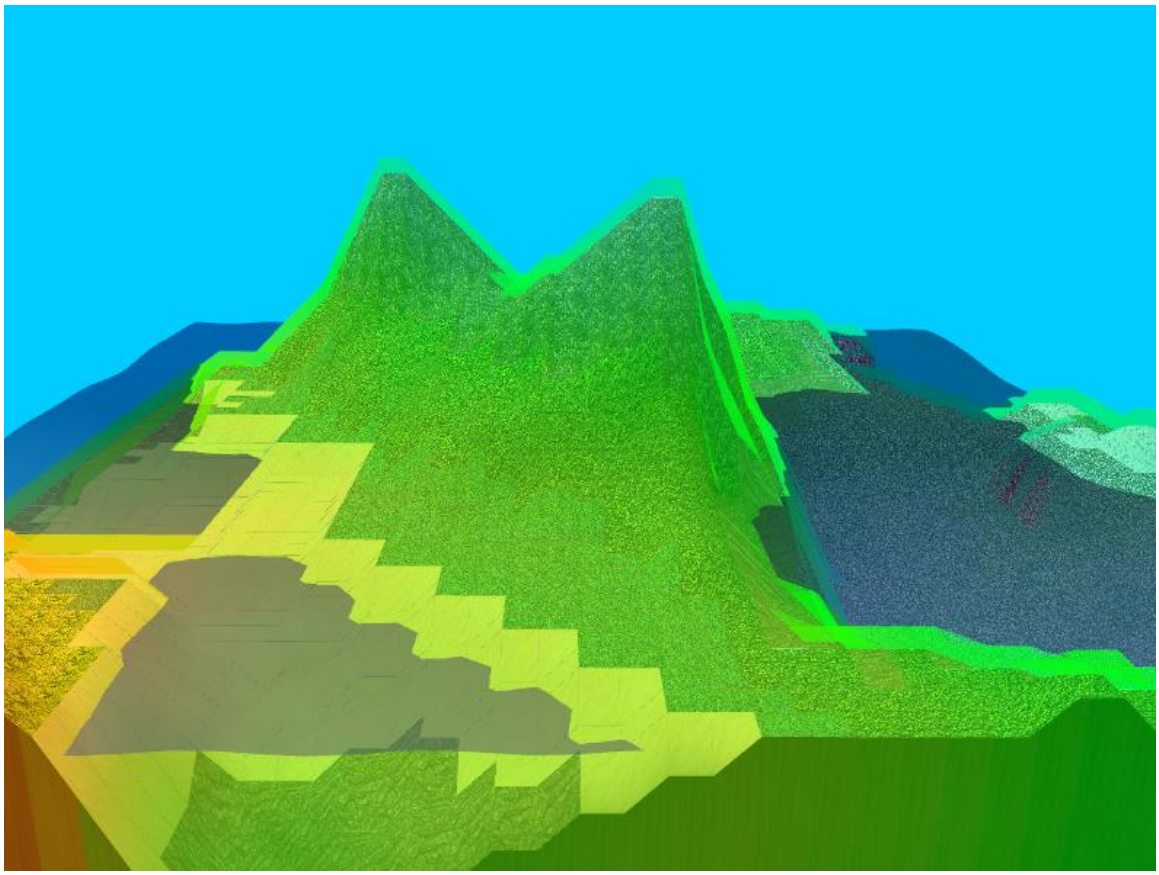
ARTIFACTS



Texture Map



Temperature Map



Temperature Overlay + Texture Map