COSC 422 Report 1 - mjs351

Outline

The bezier program loads the patches from one of the two files, tessellates it using the algorithm described in the lecture notes, and renders it with ambient and diffuse lighting. The only major change in this program from the specification is the generation of smooth normals as described in the equation section.

This did have some challenges, as when the bezier patch is a triangle then the derivative of the bezier patch algorithm can result in vectors of length zero. This occurs in several places on the gumbo model. To work around this issue, I clamped the tesselation coords u/v to the range of [0.00001, 0.99999] when generating the normals, as the bezier patch derivative only returned invalid results when t=0 or t=1.

The terrain program loads the heightmap and generates a grid of flat patches. This is tessellated with the height data retrieved from the given heightmap. Finally, it is rendered with ambient and diffuse lighting, using textures that are dependant on the height of the vertex.

The transition point between snow and grass is linearly interpolated to provide a smooth texture transition. The transition between water and grass is sudden, and the water is a simple texture when the height of the vertex is below the water height. To have a flat water surface, the vertex is clamped to the water level. The major disadvantage of this technique is the lack of transparent and refractive water.

Cracking was initially an issue, however it was resolved by using the average of the vertices along a given edge, rather than the centre of the patch. Since the vertices are all shared between edges, getting the average will result in the same tessellation level, thus resolving the issue of cracking. This is only true when the vertices are shared between edges, which is true in the terrain program.

Equations

The algorithm for the explosion is a simple physics algorithm, as below.

$$d = -\frac{I\,\hat{c}_y + \sqrt{I\,\hat{c}_y{}^2 - 2Gc_y}}{G}$$

$$p_{out} = p_{in} + (2r\,\hat{c}_x,\ I\,\hat{c}_y r + 0.5Gr^2,\ 2r\,\hat{c}_z)$$

Where c is the centre of the patch, G is gravity, I is the initial velocity, r is t in the range $[0,\ d]$, t is the time in seconds since the explosion, p_{out} is the output position and p_{in} is the input position.

The tessellation level for both programs is based on the distance between the vertex position and the camera, $(1-(d-Dmin)/(Dmax-Dmin))(Llow - Lhigh) + Lhigh$ where d = the distance between the vertex and the camera.

The equations for solving cracking are:

$tessLevelOuter[0] = f((position[0] + position[1]) / 2)$

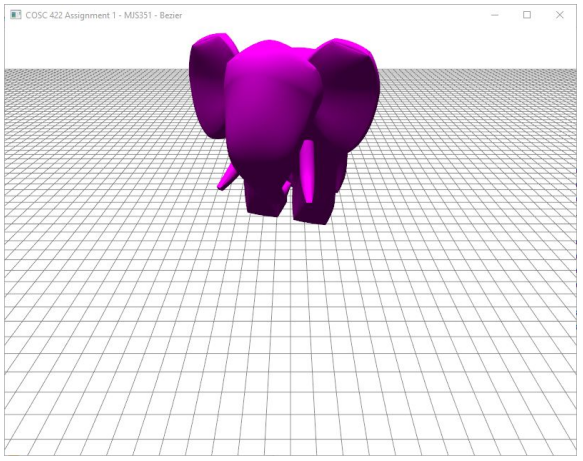
$tessLevelOuter[1] = f((position[0] + position[3]) / 2)$

$tessLevelOuter[2] = f((position[2] + position[3]) / 2)$

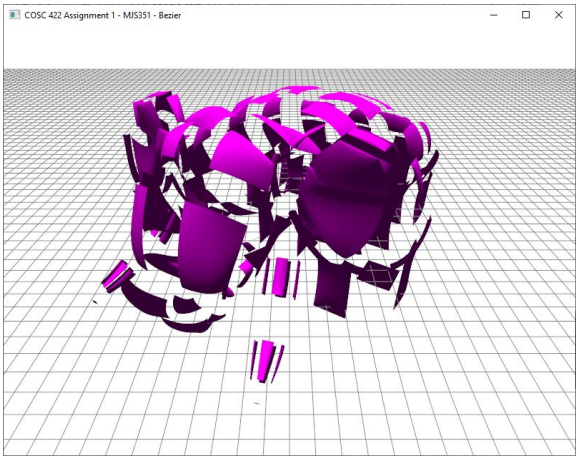
$tessLevelOuter[3] = f((position[1] + position[2]) / 2)$

Where f is the function to calculate the tessellation level (as above), position is an array of vertices, and tessLevelOuter is the tessellation level to set.

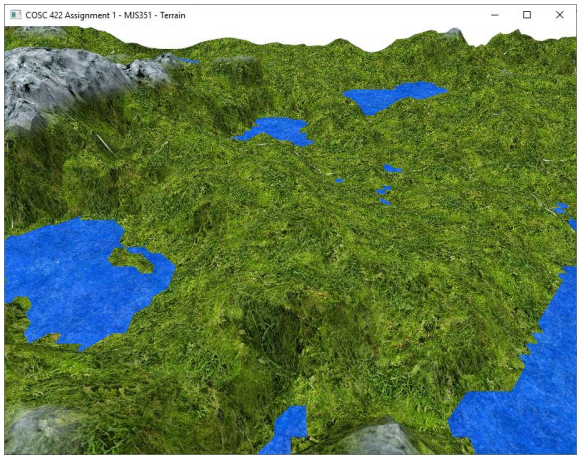
Screenshots



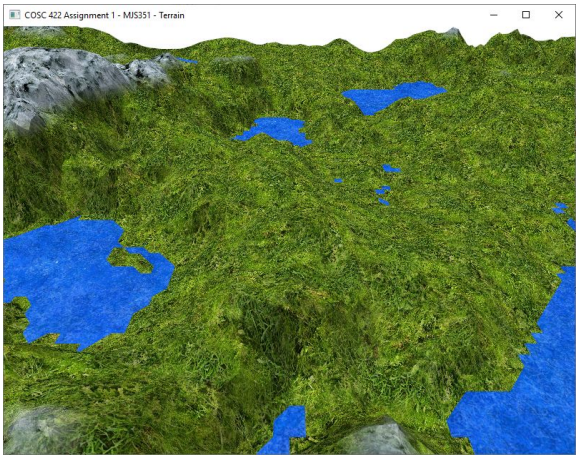
The gumbo model.



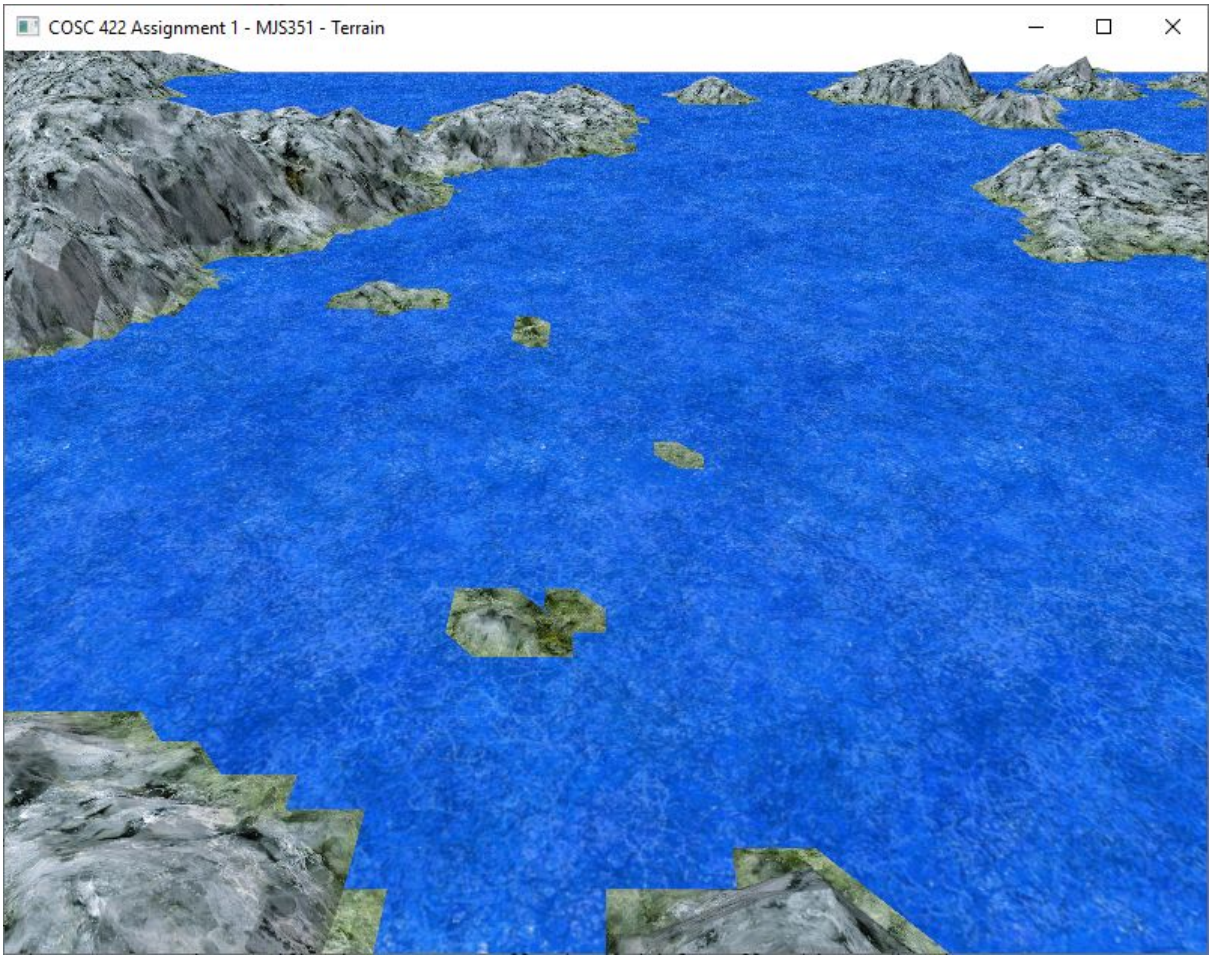
Exploding gumbo model.



Terrain with cracking.



Terrain with cracking fixed.



Terrain with water level and snow level changed.

Requirements

OpenGL 4.5+
DevIL
GLEW
GLUT
GLM

Controls

Bezier:
Up/Down - Move camera forward/backwards
W - Toggle wireframe

Terrain:
Up/Down/Left/Right/+/- - Move camera forward/backwards/left/right/up/down
W - Toggle wireframe
1/2 - change heightmap
v/b - lower/raise water level
n/m - lower/raise snow level

Credits

Heightmap: http://adventuresinsilicon.com/wordpress/?page_id=250
Textures from: https://opengameart.org/node/10593
https://opengameart.org/content/terrain-textures-pack-from-stunt-rally-23?page=4