Computer Graphics

CMPS 373

Fall 2023

Dr. Osama Halabi

Course Project

Mohammed Al-Obaidly (201801987)

Mohammed Salah (201912733)

Omar Elshenhabi (202106273)

Terrain Sculptor

**Introduction:**

The aim of this project is to help us understand the basics of cubemaps, procedural generation, real-time 3D object manipulation, ray-casting, geometry shaders and sculpting, and the use of cursors to manipulate in-window objects, by creating a terrain sculptor using GLEW and GLFW, as well as other libraries.

**Project Description:**

The terrain sculptor contains a flat, procedurally generated square plane composed of right triangles (modeling). The sculptor also allows the user to traverse scene where the plane is, to give the user more control in how and where to sculpt (interactivity), nevertheless, the user has the choice to switch between Mouse Cursor Mode and Sculptor Mode (interactivity). Most importantly, the sculptor allows the user to sculpt the plane using the cursor (interactivity), when the mouse’s left button is pressed, the terrain is elevated, while getting depressed when the right button is pressed (animation?). The user can also control the strength of the cursor, depending on what they feel suits their current needs (interactivity), with the size of the cursor changing in real time (animation).

**Technical Details:**

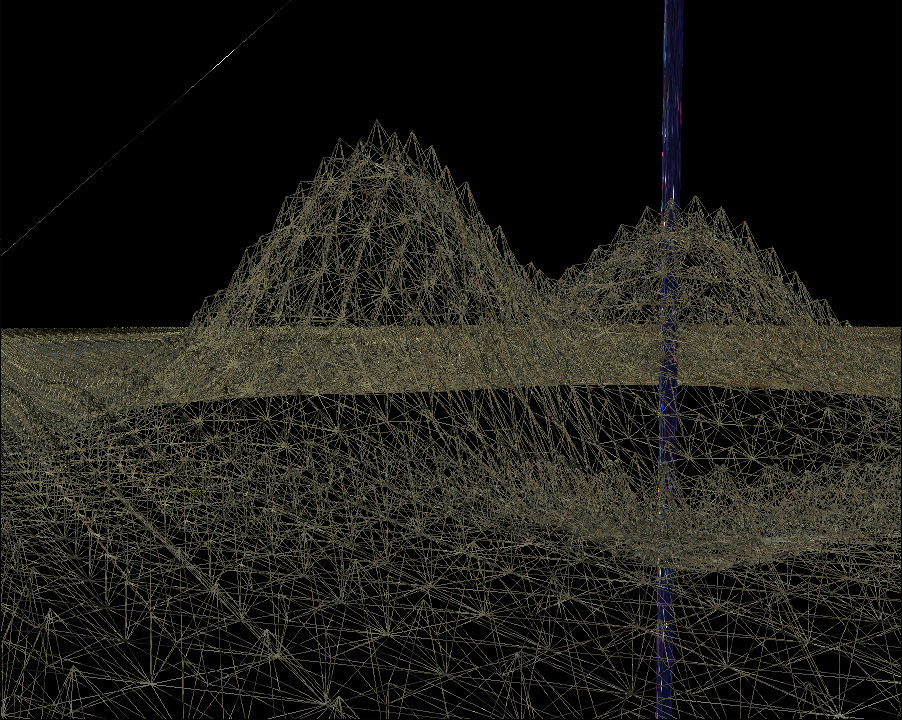
The ray-casting process transforms a 2D mouse click into a 3D world coordinate. It starts by converting the mouse position from screen space to Normalized Device Coordinates (NDC). These coordinates are then transformed to eye space using the inverse projection matrix, and subsequently to world space using the inverse view matrix. This results in a 3D ray extending from the camera into the scene.

The next step involves calculating the intersection of this ray with a horizontal plane at the origin. The process checks if the ray is parallel to the plane to avoid meaningless intersections. If not parallel, the intersection point on the plane is computed using the plane's normal and the ray's direction. The result is the 3D world coordinates corresponding to the 2D mouse click, effectively bridging 2D user input with 3D spatial interaction, then using the same logic for modifying the plane using 3D cylinder but replacing the coordinates with the obtained mouse cursor coordinates.

The plane is generated procedurally, taking two inputs: The width of the sides of the plane (float width), and the number of segments per side (int div). As such, a plane with div = 1 has two tris making up one square. The function achieves that using a loop. Using another loop, the indices of the vertices are generated, and using that information the mVertices vector is initialized. The generation function also takes care of generating the texture coordinates.

The function used to actively modify the plane does so by looping through all the vertices in mVertices and modifying the y-component of the position based on: The distance between the vertex x- and z-components and the cursor’s position, the strength of the cursor, and whether it should elevate or depress the terrain. The function that determines the amount by which the y-component is modified is called “gaussian(float x)”, which is just an exp(-x2) function.

As for geometry shader, we have tried implementing it but the results were not realistic and we thought it was not good enough to be in the final form for the project, below is a screen shot of what we have achieved trying to implement it, also, the code for it is still there in the project files.



**Manual:**

For Movement:

press [WASD] keys to move in the scene.

To move the camera:

press [R] and move your mouse.

To move up/down in the scene:

press [Z/X].

To increase/decrease the sculpting radius:

use mouse scroll [UP/DOWN].

To change to mouse cursor sculpting mode:

press [M].

To switch to wireframe mode:

press [F].

**References:**

[1] https://www.youtube.com/watch?v=DLKN0jExRIM

[2] <https://www.youtube.com/watch?v=FKLbihqDLsg>

[3] <https://en.wikipedia.org/wiki/Gaussian_function>