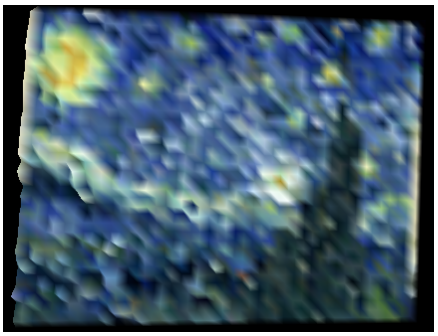
Mini project 2022

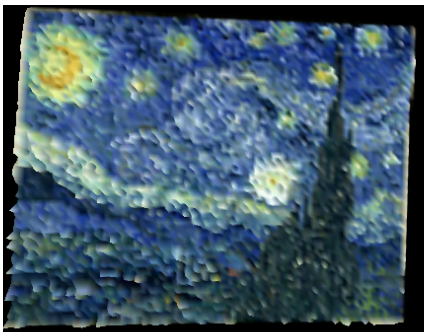
Submitted by: Noam Ariea Spiegelstein 315770453 and Hadar Lieber 208827352

In this project, we developed a software application that uses altitude interpolation to transform 2D images into 3D models using mesh generation. The program is written in C++, using the libraries OpenGL and OpenCV. The generated 3D models can be translated, scaled, and rotated in the X-axis, Y-axis, Z-axis, and Zoom-in and out.

The program enables 3D model representation in two resolutions (high and low); wherein the high resolution model is constructed of 4x triangles from the number of triangles in the low resolution.



LOW-RES



HIGH-RES

The Program represents the triangles used to draw the 3D model with the following structs:

```
typedef struct TriPoint {
        GLint x;
        GLint y;
}TriPoint;

typedef struct TriPointColor {
        GLfloat z;
        GLfloat r;
        GLfloat g;
        GLfloat b;
}TriPointColor;
```

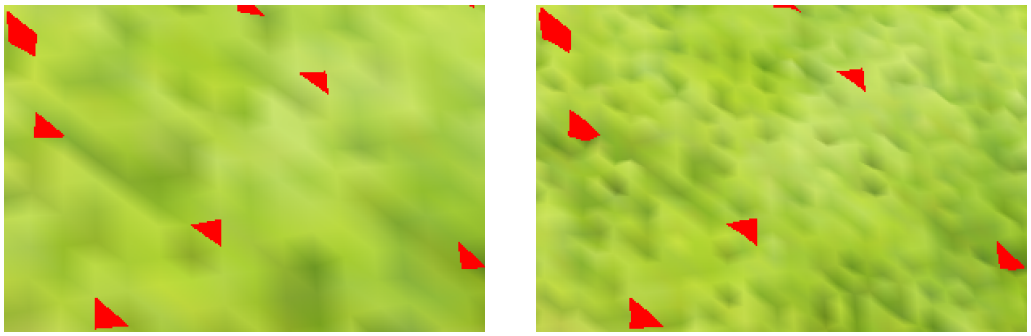
```
typedef struct Triangle {
        TriPoint p1;
        TriPointColor c1;
        TriPoint p2;
        TriPointColor c2;
        TriPoint p3;
        TriPointColor c3;

        bool isPicked;
        GLfloat left_r;
        GLfloat left_g;
        GLfloat left_b;
        int parentId;
        int vectorIndex;
        bool collision;
        int bfsParentId;
        int maxZ;
}Triangle;
```

The triangles are initially loaded once per image specification at the beginning of the program into 3 different Triangle vectors. The triangles vectors are Low-Resolution, High-Resolution and High-Resolution-Collision. Low-Res and High-Res vectors enable picking persistence throughout the program, and High-Res-Collision vector enables efficient path-finding in high resolution. Pre-loading the vectors substantially lowered the buffer time between moving from high-res to low-res.

The program enables picking of triangles - The user can choose specific triangles with the mouse. We implemented that by giving each triangle a unique ID (which is stored in its special coloring members left_r/g/b). When the user clicks the screen, the model is drawn with its color-ID for a short time. The color pointed to by the mouse is then read and translated to a triangle-id (its location in the current vector), and the triangle at the specified id is signed as picked through its member isPicked. Picked triangles are pushed into a picked-triangles map to enable efficient picking translation from high-res to low-res. The function is reversible; when the user clicks a selected triangle, it returns to its original color.

low res picking and its translation to high res picking and vice versa



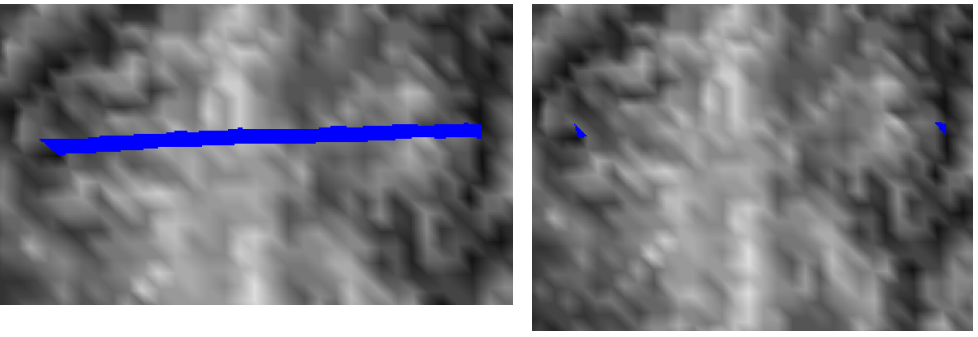
Special Operations

1.Shortest path - implementation with BFS

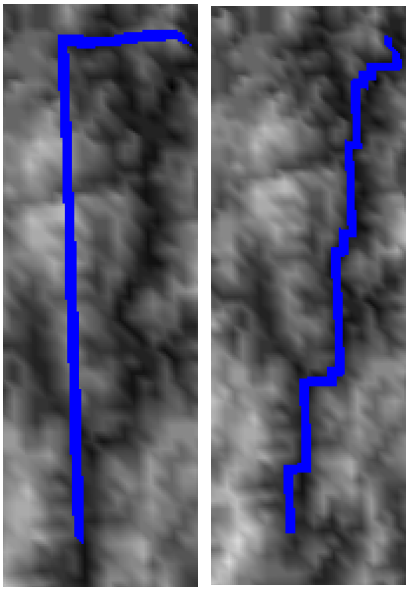
2.flight path finding (path without collision) - Implementation with BFS

Shortest Path and collision comparison

Collision failure case

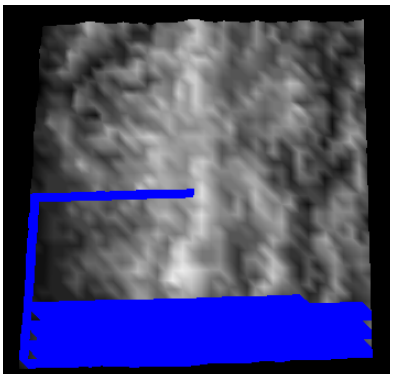


Collision success case



3.path between 2 points - implementation with DFS

DFS path example



Interaction with the Program is done via the following keyboard and mouse prompts.

Up	Zoom-In
Down	Zoom-Out
Left	Y rotation
Right	Y rotation
Z	Z rotation
X	Z rotation
M	X rotation
N	X rotation
H	High-resolution
L	Low-resolution
B	BFS
C	No collision path
D	DFS
Left click	Pick a triangle