Daniel Lynch
EECE 2160: Application Programming with Dr. Michael Geiger
Honors by Contract Project Report
Dec. 15, 2017

Proposal

My proposal for the *Honors By Contract* project was to write a program in C that used geographic data to generate STL files (.stl). These files are used in various software to represent surfaces. They can be used in CAD, graphics applications, and the 3D printing process.

Initial Plan

The project consisted of two parts: getting the data and generating the STL file. First, I would research the STL file format come up with a pen-and-paper method for writing an STL file based on a 2D grid of values. Then I would start to implement that method in my program. Next, I would search for publicly available geographic data. Once, I found a useable source I would work on reading that data into a 2D grid of values, so that it could then be written to an STL.

Background Information

STL Files

STL files are text or binary files that contain information that describes a 3D surface. The file format extension comes from the term “stereolithography” [1]. The file format was originally developed by 3DSystems, one of the first commercial 3D printer manufacturers. 3DSystems still makes printers today. STL files can be in binary format or in an ASCII-encoded text file.

Both types of STL files are essentially lists of triangles called facets. The collection of facets (i.e. triangles) is usually referred to as a mesh. Each facet is a list containing three vertices and a normal vector. Each vertex and normal vector have three coordinates representing (unitless) XYZ space [2]. The normal vector of facet is a unit normal that points to the outside of the surface. In general STL files are unitless and the slicer software interprets the value as being a number in inches or mm (usually mm).

There are a few rules when generating STL files that, if not followed, can lead to the file being unreadable in some if not all software:

1. The normal vector must be a unit vector and must point to the outside of the surface
2. The vertices of a facet need to be listed in a counter-clockwise order, looking at the outside of the facet.
3. All vertex coordinates need to be positive. Normal vectors can contain negative numbers.
4. All numbers should be floats in scientific notation

These rules are basically it. The facets can be listed in any order although it is recommended that they are listed in a contiguous order from lowest Z-value to highest Z-value. It is also recommended that the surfaces described are “water-tight” meaning they wrap around a volume that is completely contained.

See Appendix A for an example of an ASCII-encoded STL file.

GIS

GIS or “Geographic Information Systems” refers to collecting, publishing, analyzing, and using geographic information [3]. There are many sources for obtaining geographic data including government agencies, for-profit businesses, and non-profit academic institutions. The United States Geographic Survey (USGS) is a free, publicly available source of GIS data.

There are also many types of geographic data (e.g. elevation, watershed, vegetation, climate, etc.). I chose to work with elevation data in this project. The USGS has a few ongoing projects with elevation data sources, one of which is the National Elevation Dataset (NED) project. The NED database contains elevation data for the entirety of the continental U.S. and some parts of Alaska [4].

3D Printing

The starting point of the 3D process usually begins with some sort of file describing a mesh. This file could be an ASCII STL file or a binary STL file or even another type of file like an OBJ file. This file is brought into a software called a “slicer.” There are various slicer software available,- some are designed for specific printers and some are designed to be used with many different types of printers. The user then enters a bunch of parameters into the slicer software and the slicer than uses the parameters and the mesh file to generate a toolpath (usually in some flavor of G-code) for the printer to interpret. In this project, the slicer software I used is Cura [5], which is an open-source slicer software that supports almost any non-proprietary 3D-printer. I used Cura to open the STL files I generated for viewing and to print the files on a Ultimaker 3 [6] printer at the UML Makerspace.

Method

First, I needed to define what my input to the C program should look like. I decided to use a text file as input. To see an example of two of the text files that I used in the beginning for debugging purposes, see Appendix B1 and B2. The first row of the text file contained the number of rows and column in the grid. After that was the 2D list of values itself with rows separated by line and columns separated by spaces.

My pen-and-paper consisted of three parts. First I would build the top surface of the model. The method for this part is seen in Appendix C1. I would iterate through the grid one entry at a time. At each entry, I would define two triangles (making a square when projected onto the X-Y plane) based on the value at that entry and the three values adjacent to that entry. I would write these facets to the STL file and then move on.

I defined a data structure called *Facet* which represented a single facet of the mesh. So that I wouldn't have to keep too many values in memory, I only worked with two facets at any time. The facets were continually overwritten, printer, overwritten, printed, etc.

To define the sides and bottoms was easier and involved the same iterative process but I only had to iterate over a single row or column for each and instead of using three adjacent values, I only had to use one adjacent value (the third value for each triangle was just zero). My method for defining a side surface is shown in Appendix C1. The bottom only consisted of two facets which were defined by the four corners values of the grid and again just zero.

Progress

By the end of the term I was able to read in data from essentially a delimiter-spaced text file and generate STL files based on the data that could be opened for viewing and printed without throwing any errors in the slicer software.

Hurdles and Changes to Plan

Surprisingly obtaining and parsing the geographic data was much harder than expected. Although elevation data from NED is publicly available to download, it is downloadable only three types of formats: two of which are proprietary file formats and one of which has very little documentation. Usually users open the data with software like ArcGIS and AutoCAD Civil 3D, so they don't parse through the files directly. I did not have enough time to learn how obtain the elevation data from the files available to download from the USGS [7]. So, I looked into other sources of elevation data like the Google Maps Elevation API [8]. After reading through the documentation, I decided not to pursue using an API because I do not have experience with JavaScript. In order to get my program doing something, I decided to use photograph as the input. Although I was not able to make sense of the file formats from the USGS website, each grid of data from the NED database, contained a .jpeg "thumbnail" image of a raster of the data.

I used this photograph to very-roughly obtain some elevation data. I wrote a quick script in MATLAB that would do some image manipulations (Gaussian Blurs, contrasting, resizing, etc.) [9] and then generate a 2D list of values and write it to a text file.

Current State of Project

I was able to write a C program that could generate useable STL files based on a text file representing a 2D grid of values (e.g. elevation values). I was not able to write a C program that could collect elevation data from through file obtained. I was able to roughly obtain elevation data from the USGS by using the thumbnail images of the "tiles" of elevation data available for download and generating a text file of a 2D grid of values based on the images in MATLAB.

Results

Below are some screenshots and photographs of results obtained throughout the process. They start with some of the first surfaces that were able to be generated and end with some of the 3D printed surfaces of elevation models based on thumbnails from the USGS database.

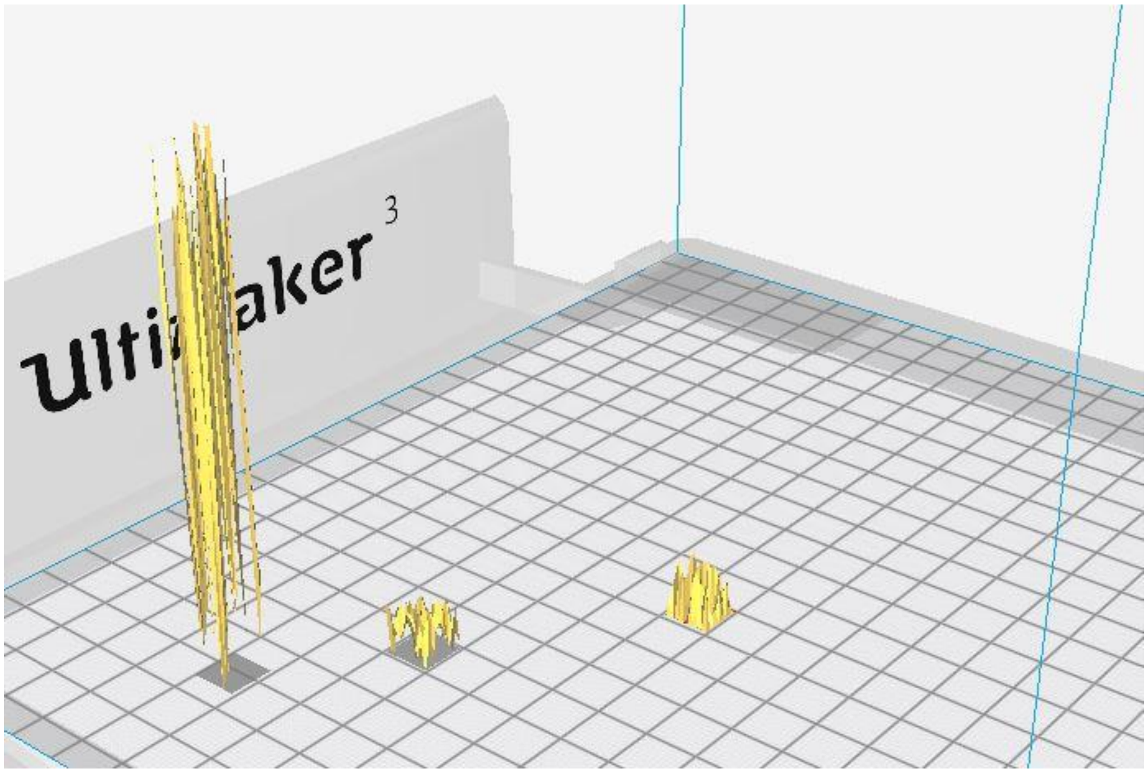
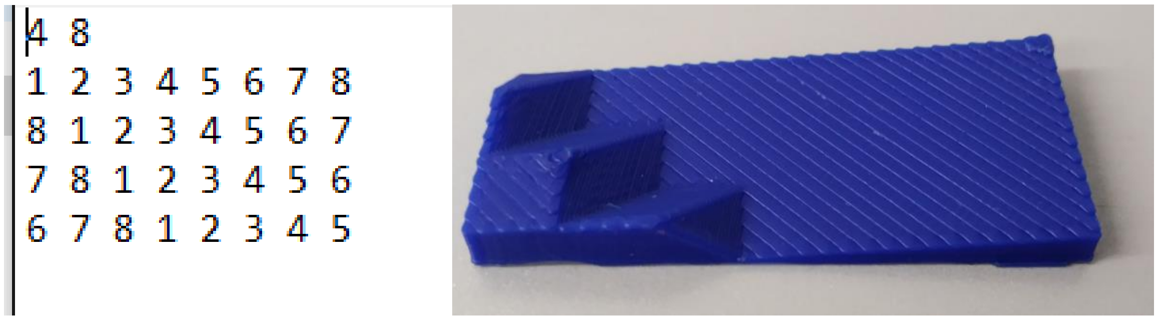
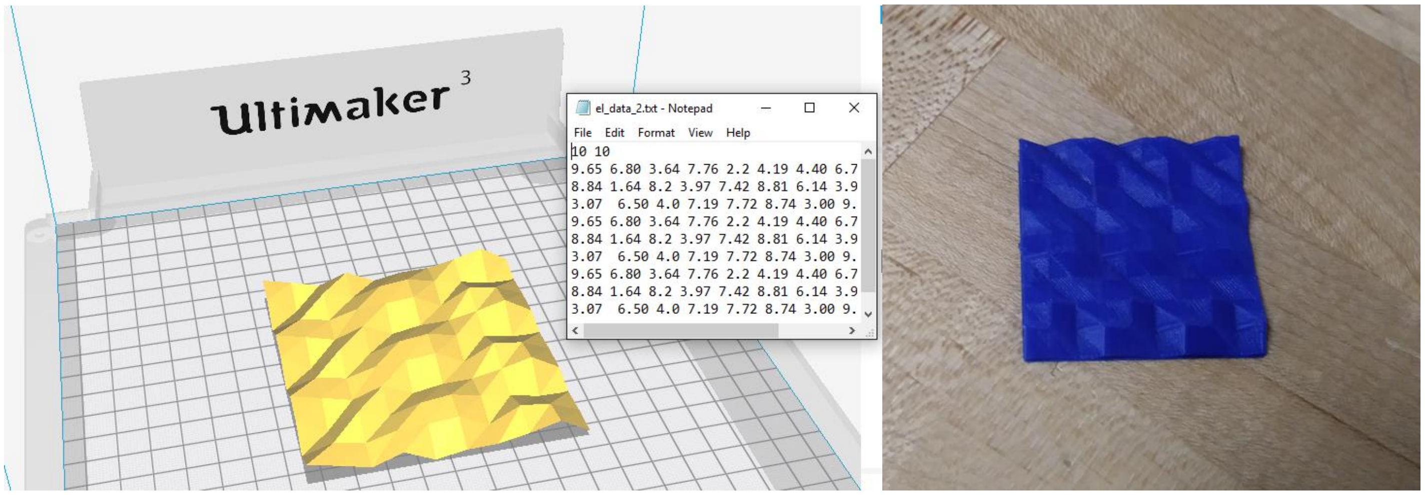


Fig. 1. Examples of some of the first meshes (with errors) made with the program. The one on the left showed me that I needed some sort of mapping of the given elevation values to reasonable heights for prints. These surfaces are also all missing sides and bottoms.



```
4 8
1 2 3 4 5 6 7 8
8 1 2 3 4 5 6 7
7 8 1 2 3 4 5 6
6 7 8 1 2 3 4 5
```

Fig. 2- A screenshot of one of the test 2D grid text files I used in the process of writing the program and then a photo of the 3D print of its generated mesh.



```
10 10
9.65 6.80 3.64 7.76 2.2 4.19 4.40 6.7
8.84 1.64 8.2 3.97 7.42 8.81 6.14 3.9
3.07  6.50 4.0 7.19 7.72 8.74 3.00 9.
9.65 6.80 3.64 7.76 2.2 4.19 4.40 6.7
8.84 1.64 8.2 3.97 7.42 8.81 6.14 3.9
3.07  6.50 4.0 7.19 7.72 8.74 3.00 9.
9.65 6.80 3.64 7.76 2.2 4.19 4.40 6.7
8.84 1.64 8.2 3.97 7.42 8.81 6.14 3.9
3.07  6.50 4.0 7.19 7.72 8.74 3.00 9.
```

Fig. 3- A screenshot of another one of the test 2D grid text files I used in the process of writing the program and its generated mesh in the Cura Slicer (left). A photo of the 3D print of the mesh (right).

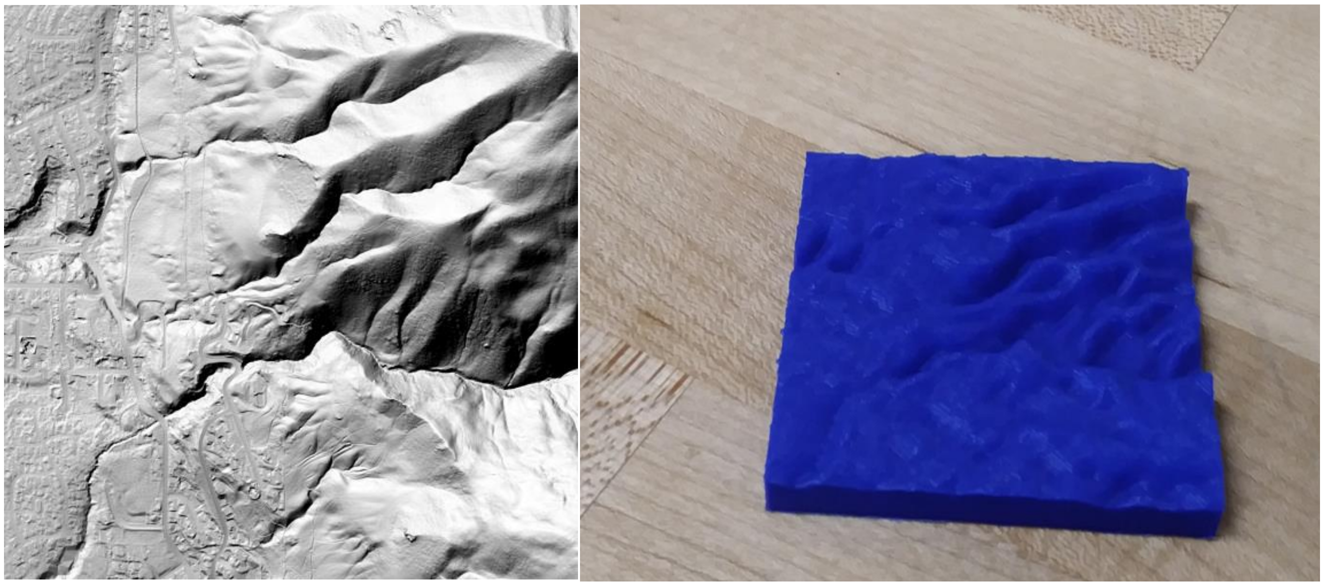


Fig. 4- Thumbnail of a USGS NED [4] raster of an area near Salt Lake in Utah and of Lowell a photograph of a 3D-print of the mesh the program generated based on it.

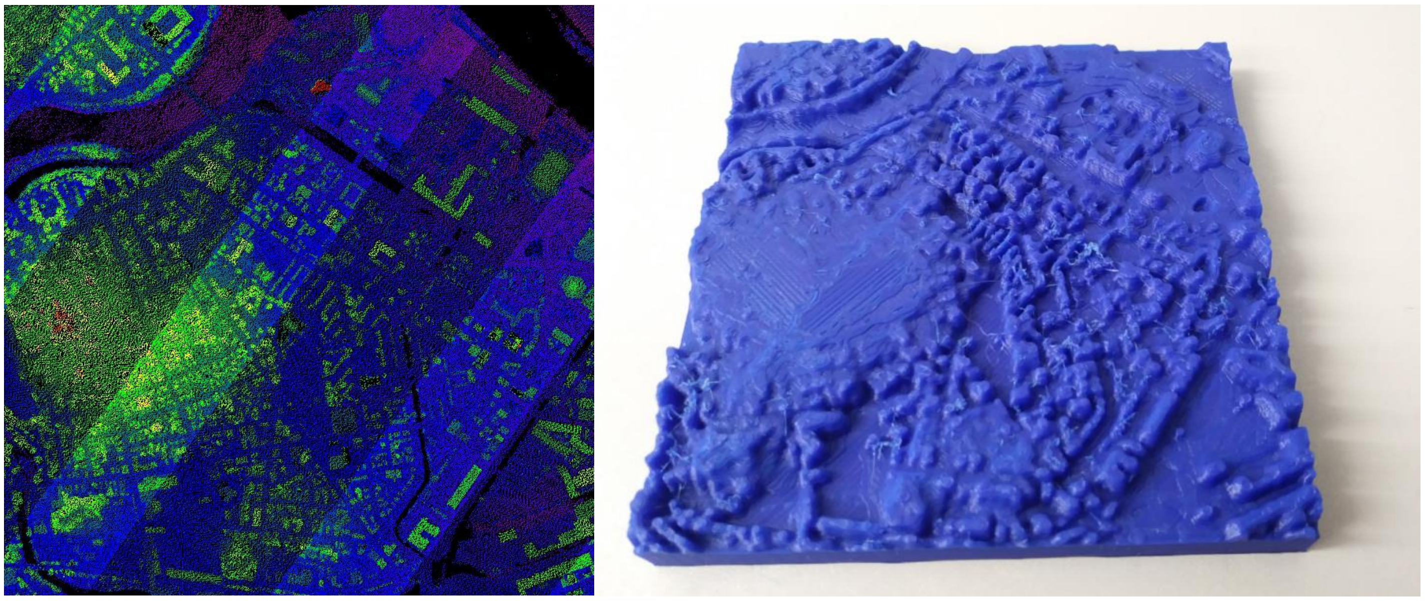


Fig. 5- Thumbnail of a USGS LiDAR point cloud of Lowell and a photograph of a 3D-print of the mesh the program generated based on it. Parts of North Campus (the Lyndon Library and the North Campus Quad) are in the upper left-hand corner.

Code

The C program is called *GeoMesh*. It consists of five files: *geomech.c, geomesh_fcns.c, geomesh_fcns.h, facet.c,* and *facet.h*.. The input filenames need to be specified in *geomesh.c*. The dimensions of the 2D grid need to match the dimension of the 2D grid in the text file being read. They are defined as the constants, "ROW" and "COLUMN" in *geomesh_fcns.h*. They should both be even numbers. If the product (ROW * COLUMN) is greater than about 40,000, then the stack reserve size of the IDE or compiler being used may need to be adjusted from the default setting. The input filenames come from the "/tfiles" folder of the project folder. The output STL files are saved in the "/meshes" folder of the project folder.

The MATLAB program to prepare the 2D grids of values as text files for input is called *IMGmanipulations.m*. The name of the output text file and the input image file needs to be specified in this script. The photo files used as input to this script should be placed in the "/photos" folder of the project folder. The text file outputs of the MATLAB script will be save in the "/tfiles" folder of the project folder so that they will be accessible to the GeoMesh program.

A project folder with the correct structure was submitted with this report. The project folder submitted with this report contains a few example photographs in the photos folder. Both the MATLAB script and GeoMesh program are set to use a 200 by 200 grid of points initially. The Utah thumbnail photo's text file and STL file are also included.

The MATLAB program takes about 4 seconds to run on my old Lenovo Thinkpad. The GeoMesh runtime varies and can be anywhere from a second or so for small grids to 20 seconds or so for larger ones

The STL files can be opened in any slicer software (Cura, Slic3r, Makerbot Print, etc.). They can also be opened in the built-in "3D Builder" viewer in Windows 10. You can also view by using viewstl.com [10], which is a client-side web app that lets you view STL files in your browser.

Sources

[1] 3Ders. “What is an STL?” [Online] Available: https://all3dp.com/what-is-stl-file-format-extension-3d-printing

[2] Fabbers. “The STL File Format.” [Online] Available: http://www.fabbers.com/tech/STL_Format

[3] USGC. “GIS Data.” [Online] Available: https://www.usgs.gov/products/data-and-tools/gis-data
[4] USGS. “NED Database.” [Online] Available: https://topotools.cr.usgs.gov/pdfs/gesch_chp_4_nat_elev_data_2007.pdf

[5] Ultimaker. “Cura Software Download.” [Online] Available: https://ultimaker.com/en/products/ultimaker-cura-software
[6] Ultimaker. “Ultimaker 3.” [Online] Available: https://ultimaker.com/en/products/ultimaker-3

[7] USGS. “NED Downloader.” [Online] Available: https://viewer.nationalmap.gov/basic/?basemap=b1&category=ned,nedsrc&title=3DEP%20View#startUp

[8] Google. “Google Maps Elevation API.” [Online] Available: https://developers.google.com/maps/documentation/elevation/start

[9] MathWorks. “MATLAB Image Processing Toolbox Documentation.” [Online] Available: https://www.mathworks.com/products/image.html

[10] ViewSTL. [Online] Available: https://www.viewstl.com/

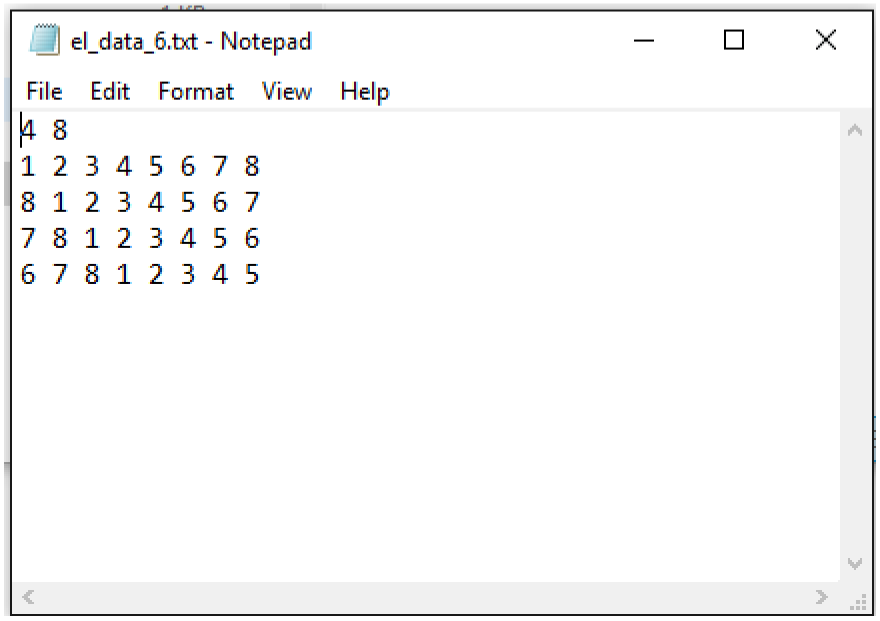
Appendix

A1. An ASCII STL File describing a box.

Since there are 6 sides there should be 12 triangles and facets. However, I cut part of the file out (replaced by ellipses) to save space and paper.

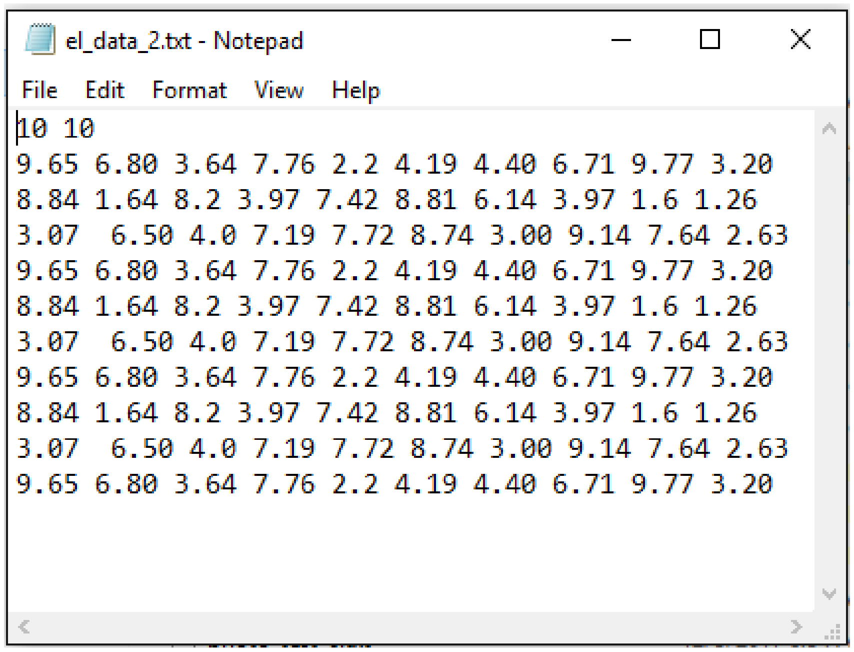
```
solid ASCII
  facet normal 0.000000e+00 1.000000e+00 0.000000e+00
    outer loop
      vertex   3.000000e+01 1.000000e+01 2.500000e+01
      vertex   3.000000e+01 1.000000e+01 1.000000e+01
      vertex   5.000000e+00 1.000000e+01 2.500000e+01
    endloop
  endfacet
  facet normal 1.000000e+00 0.000000e+00 0.000000e+00
    outer loop
      vertex   3.000000e+01 1.000000e+01 1.000000e+01
      vertex   3.000000e+01 0.000000e+00 2.500000e+01
      vertex   3.000000e+01 0.000000e+00 1.000000e+01
    endloop
  endfacet
  facet normal -0.000000e+00 0.000000e+00 1.000000e+00
    outer loop
      vertex   5.000000e+00 1.000000e+01 2.500000e+01
      vertex   5.000000e+00 0.000000e+00 2.500000e+01
      vertex   3.000000e+01 1.000000e+01 2.500000e+01
    endloop
  endfacet
  facet normal 0.000000e+00 0.000000e+00 1.000000e+00
    outer loop
      vertex   3.000000e+01 1.000000e+01 2.500000e+01
      vertex   5.000000e+00 0.000000e+00 2.500000e+01
      vertex   3.000000e+01 0.000000e+00 2.500000e+01
    endloop
  endfacet
endsolid
```

B1. A n by n text file used as input



```
4 8
1 2 3 4 5 6 7 8
8 1 2 3 4 5 6 7
7 8 1 2 3 4 5 6
6 7 8 1 2 3 4 5
```

B2. A n by n text file used as input



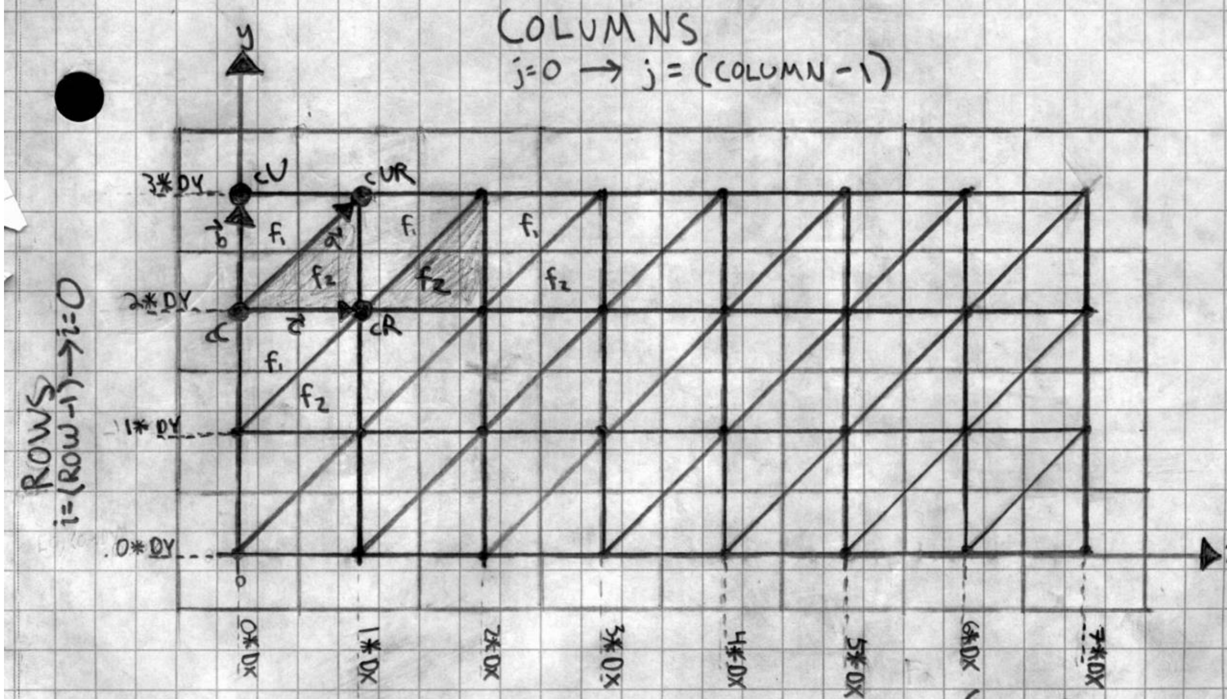
```
10 10
9.65 6.80 3.64 7.76 2.2 4.19 4.40 6.71 9.77 3.20
8.84 1.64 8.2 3.97 7.42 8.81 6.14 3.97 1.6 1.26
3.07  6.50 4.0 7.19 7.72 8.74 3.00 9.14 7.64 2.63
9.65 6.80 3.64 7.76 2.2 4.19 4.40 6.71 9.77 3.20
8.84 1.64 8.2 3.97 7.42 8.81 6.14 3.97 1.6 1.26
3.07  6.50 4.0 7.19 7.72 8.74 3.00 9.14 7.64 2.63
9.65 6.80 3.64 7.76 2.2 4.19 4.40 6.71 9.77 3.20
8.84 1.64 8.2 3.97 7.42 8.81 6.14 3.97 1.6 1.26
3.07  6.50 4.0 7.19 7.72 8.74 3.00 9.14 7.64 2.63
9.65 6.80 3.64 7.76 2.2 4.19 4.40 6.71 9.77 3.20
```

C1. Diagram of the Method for Writing the Top Surface

COLUMNS

ROWS

	e[][0]	e[][1]	e[][2]	e[][3]	e[][4]	e[][5]	e[][6]	e[][7]
e[0][]	1 e[0][0]	2 e[0][1]	3 e[0][2]	4 e[0][3]	5 e[0][4]	6 e[0][5]	7 e[0][6]	8 e[0][7]
e[1][]	8 e[1][0]	1	2	3	4	5	6	7 e[1][7]
e[2][]	7 e[2][0]	8	1	2	3 e[2][4]	4	5	6 e[2][7]
e[3][]	6 e[3][0]	7 e[3][1]	8 e[3][2]	1 e[3][3]	2 e[3][4]	3 e[3][5]	4 e[3][6]	5 e[3][7]



f_1 { V_1: cC, V_2: cUR, V_3: cU, n: $\vec{a} \times \vec{b}$ }

f_2 { V_1: cC, V_2: cR, V_3: cUR, n: $\vec{c} \times \vec{a}$ }

$$\begin{tikzcd}
\text{e[i][j] value at i,j index} \arrow[r, bend left] & \text{x, y, z coordinate in 1st octant}
\end{tikzcd}$$

First Cursor Position: (e[1][0]):

DX = 10, DY = 10, ROW = 4, COLUMN = 8;

$i = 1$

$j = 0$

$e[i][j] = e[1][0] = 8$

$$cC = [\,(j * DX),\,(((ROW-1)-i) * DY),\,(e[i][j])\,]$$

$$cC = [\,(0 * 10),\,(((4-1)-1) * 10),\,(8)\,]$$

$$cC = [\ \underset{\substack{\uparrow\\ \text{x}\\ \text{coordinate}}}{0},\ \underset{\substack{\uparrow\\ \text{y}\\ \text{coordinate}}}{20},\ \underset{\substack{\uparrow\\ \text{z}\\ \text{coordinate}}}{8}\ ]$$

$$cR = [\,((j+1) * DX),\,(((ROW-1)-i) * DY),\,(e[i][j+1])\,]$$

$$cR = [\,(1 * 10),\,(((4-1)-1) * 10),\,(e[1][0+1])\,]$$

$$cR = [\,10,\,20,\,1\,]$$

$$cUR = [\,((j+1) * DX),\,((ROW-i) * DY),\,(e[i-1][j+1])\,]$$

$$cUR = [\,((0+1) * 10),\,((4-1) * DY),\,(2)\,]$$

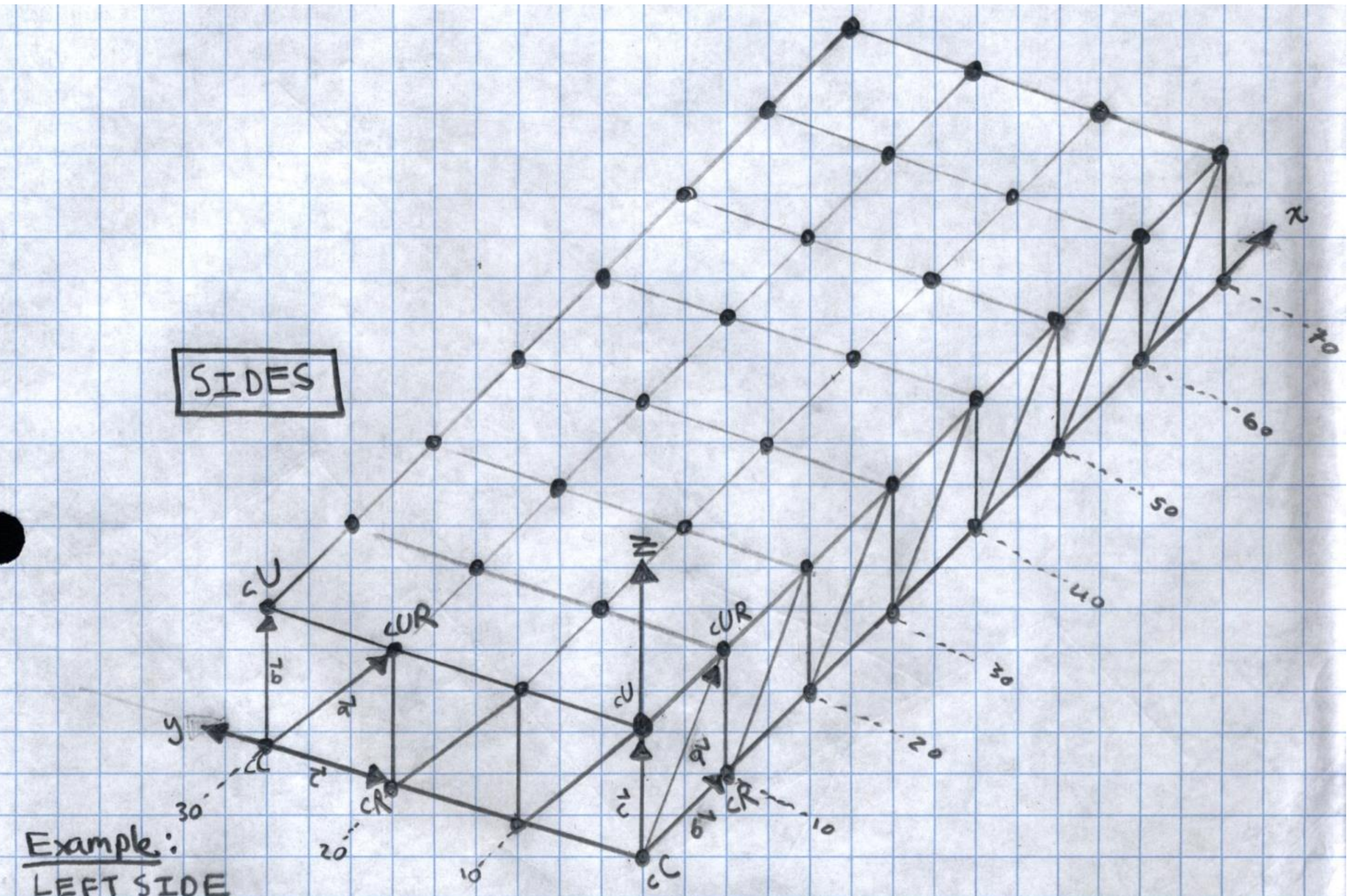
$$cUR = [\,10,\,30,\,2\,]$$

$$cU = [\,(j * DX),\,((ROW-i) * DY),\,(e[i-1][j])\,]$$

$$cU = [\,(0 * 10),\,((4-1) * 10),\,(e[0][0])\,]$$

$$cU = [\,0,\,30,\,1\,]$$

C2. Diagram of the Method for Writing the Top Surface



Example:

LEFT SIDE

{Iterate from $i=0$ to $i=(ROW-1)$}

First Cursor: $i=0$, $ROW=4$, $DY=10$

$cC=[0,(((ROW-1)-i)*DY),0]=[0,(((4-1)-0)*10),0]$

$cC=[0,30,0]$

$cR=[0,(((ROW-1)-(i+1))*DY),0]=[0,(((4-1)-(0+1))*10,0]$

$cR=[0,20,0]$