**Basics**

Procedures (understanding the data structure might help understand the following. This in turn explains why the data structure was used. To go to data structure, click here.) :

The object started out as a simply dodecahedron-20 vertices with specific coordinates that make up the 12 pentagonal faces. To make the math slightly easier, we conceptually rotate the 12 faces such that the faces lie on the x-y plane. We then have to engrave symbols to each of the 12 pentagonal faces.

To engrave any symbol onto a face/surface, it must have some kind of “resolution”, just like a picture would need square pixels. However, we decided to cut the pentagons into equilateral triangles instead of squares.

\*INSERT GIF FILE HERE

Each symbol is about to be engraved into the pentagons. They are, as of now in JPEG form. Each pixel in the square picture can be represented by a number (only one number is needed because the symbols are black and white, so there are no RBG colors).

However, you might be able to see a problem: the JPEG files are squared, while the faces are pentagons. Furthermore, the pentagons are “cut-up” into triangles, while the symbols are “cut-up” into squares. Therefore, the pentagon and the square would need to first be lined up, with the center of the pentagon and the square lined up to the origin of an universal coordinate system, which would also help determine which square correspond to which triangle.

With this, everything is prepared: the whole pentagon is first engraved, then each triangle which corresponds to a white pixel is left untouched, each triangle which corresponds to a black pixel is popped back up, and each triangle which corresponds to a grayish color is partially pushed up, the darker the pixel, the higher the triangle gets lifted. Furthermore, the sides of the pentagon is also lifted back up.

The pentagons are now ready to be reassemble back into the dodecahedron. First, we must refer back to the original dodecahedron. In the original dodecahedron, which we know the coordinates of the 5 corners of each pentagon (in fact, only one corner is needed). We now find the corner from the pentagon which was engraved which corresponds to the corner of the pentagon from the original dodecahedron. With the two corners and their coordinates, a rotation matrix was made and using the rotation matrix the vertices on the pentagon which had been engraved could be rotated using the rotation matrix. After rotation, the pentagons are translated (shifted) into its appropriate position.

**Prep to Print**

The dodecahedron is basically finished…if you only plan on looking at it. However, the edges of the dodecahedron is not “zipped” and the dodecahedron is technically not enclosed because the vertices on the edges are not connected, thus the pentagons are not truly connected In order to truly connect the pentagons the duplicated vertices must be “unified”. This should have been an easy task…except there is a huge amount of vertices and the number of calculations for calculating everything against everything increase in the scale of factorials. However, many of the vertices you should not need to check because they are in the middle. Using a specific nature of now the data was stored (namely all the vertices on the edges are grouped together), the amount of vertices that needed to be tested drastically decreased. The last problem lies with the fact that the coordinates have floating point errors, thus making it harder to determine which vertices are supposed to be the same vertices and which are actually two different vertices. Fortunately, there is a sweet spot around 10^-6 to 10^-13 to round to where no two different vertices will be close enough to accidentally be merged, while all floating point errors are minor enough to be rounded…and with that the dodecahedron is finished and ready to be printed!

**Beatify**

The final part of the process, using a process known as taubin smoothing, smooths the dodecahedron and helps with aliasing problems. The resolution of the dodecahedron was also raised serval times before the aliasing problem neutralized to an acceptable level. Furthermore, to make the images fit better, the images were shifted and cropped serval times. The code was also changed to make an edges slightly thinner and the images slightly smaller compared to the dodecahedron such that the images do not collide into the edges.

The dodecahedron is now essentially done. To see how the dodecahedron is put onto this website, please visit the credits page.