THE MATH BEHIND OPTICAL ILLUSIONS

MARISOL ALBA, ADILENE OROZCO, DANNY SIMPSON
CSUB FACULTY ADVISOR: DAVID GOVE

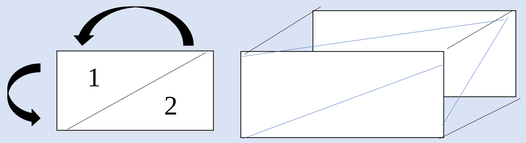
ABSTRACT

Optical illusions are a result of our brain attempting to visualize three dimensional objects from a two-dimensional plane, allowing us to see the “impossible.” Using Kokichi Sugihara’s research on optical illusions and our knowledge of matrices and vectors, we were able to recreate optical illusions into 3-D prints. We used the software program Sage to calculate new vertices of the 3-D structure that would create an optical illusion from a certain perspective(s).

VERTICES IN 3D

As with solids when creating in a three-dimensional space, quadrilaterals and triangles come to mind when creating a STL file for such shapes. For such polyhedral surfaces, those shapes can help form a solid. In 3D, vertices exist where three or more edges meet to form the faces of a solid. With various vertices, it can help with the formation of shapes and ultimately be visible in two and three dimensions.

For polyhedral surfaces, we divide each face into triangles. Multiple sets of interconnecting triangle can create various shaped faces. Typically, the order of the points that connect the vertices depends on the viewer and in what direction the surface faces. Looking toward the surfaces, the vertices of a triangle are listed counter-clockwise. The outward normal direction vector can be found using the cross product of two associated vectors; this vector is perpendicular to the vectors forming the triangle. When you divide by the magnitude you obtain the facet normal vector required in stl format.



POINTS AND PLANES

All sets of three noncollinear points in three-dimensional space determine a plane. To determine a formula for the plane, we use the three points to determine two vectors and take the cross product of theses to find the vector perpendicular to the plane. Using the fact that the dot product of perpendicular vectors is 0, leads to the following equation of the plane.

$$a(x-x_0)+b(y-y_0)+c(z-z_0)=0$$

We gratefully acknowledge the support of Chevron and the CSUB FAB lab!

METHODS

We first created a "normal/ non-illusion" version of our models using vectors to visualize how the structure would look like from any viewing angle.

In order to get the illusion version of our 3D models we had to create what we called an "eye vector." This connects to where we assume the position of the viewer's eye will be, allowing for us to transform the original placement of the vertices on some of the surfaces on our models. We did this by programming a formula into sage that transforms the original vertices into new ones.

EXAMPLE:
EYE = vector([-30, -40, 15])
P1E=vector([3.0,3.0,9.0])
P1ET= EYE + (51/43) * (P1E -EYE)
=<-30, -40, 15> + (51/43) (3+30, 3+40, 9 - 15)
=<-30, -40, 15> + (33(51/43), 43(51/43), -6(51/43))
= <-30, -40, 15> + (1683/43, 51, -306/43)
= <393/43, 11, 339/43 >
The fraction in the equation is from the formula:

$$-40+43t=11$$
$$43t=51$$
$$t=51/43$$

where 11 is the desired y-coordinate of the transformed vertex.

SAGE AND STL CODE EXAMPLES

```
def facpar(A,B,C):
    u=B-A
    v=C-A
    cr=u.cross_product(v)
    l=abs(cr)
    n=1/l * cr
    out="facet normal " + str(n[0]) + " " + str(n[1]) + " " + str(n[2])
    out=out + "\n    outer loop"
    out=out + "\n    vertex " + str(A[0])+ " " + str(A[1]) + " " + str(A[2])
    out=out + "\n    vertex " + str(B[0])+ " " + str(B[1]) + " " + str(B[2])
    out=out + "\n    vertex " + str(C[0])+ " " + str(C[1]) + " "+ str(C[2])
    out=out + "\n    end loop" + "\nendfacet"
    return(out)
```

```
;top base
facet normal 0.0 -0.0469 -0.9989
    outer loop
        vertex 2.1333 2.8 3.9556
        vertex 4.6667 2.8 3.9556
        vertex 4.6 1.0 4.04
    endloop
endfacet

facet normal 0.0 -0.0469 -0.9989
    outer loop
        vertex 2.1333 2.8 3.9556
        vertex 4.6 1.0 4.04
        vertex 2.08 1.0 4.04
    endloop
endfacet
```

"(A,B,C)" are defined as the three vertices that connect to create a triangle. We assign names to these vertices, for example:

```
BA=vector([4.0,5.0,1.0])
BB=vector([7.0,5.0,1.0])
BC=vector([7.0,7.0,1.0])
```

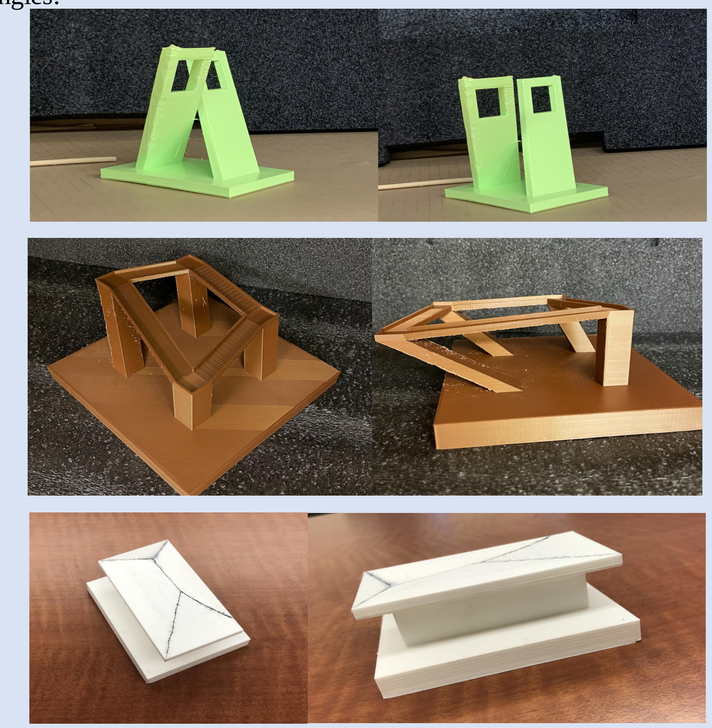
Once vertices are defined we insert the command:

```
print(facpar(BA,BB,BF))
```

To create a "paragraph" of stl code that instructs the 3-D printer to create a facet of the surface to be printed.

OUR 3-D MODELS

The following images are of the same models at two different viewing angles:



DISCUSSION

Some of the smaller details in the STL code, such as spelling and spacing, we found to be troublesome during the process. We made sure to double check our code before running it in the 3-D Viewer to avoid any mishaps. We also learned that the order in which the vertices are listed was very crucial to the overall structure of the illusion. If the vertices were listed in a clockwise manner, the surfaces would distort and create unfavorable results. Having to visualize where the vertices went on the structure also turned out to be difficult during the process, but we soon found out that creating a “blueprint” of the structures made the process smoother. Overall, we learned that the smallest of details shouldn’t be overlooked.

REFERENCES

Sugihara, K. [Kokichi Sugihara]. (2015, June 13). *Ambiguous Cylinders* [Video]. YouTube. https://www.youtube.com/watch?v=SzC-pV_moFc

Sugihara, K. (2016). Anomalous Mirror Symmetry Generated by Optical Illusion. *Symmetry*, *8*(4), 21. doi: 10.3390/sym8040021

Tech Insider. (2016, October 22). *Japanese professor creates amazing 3D optical illusions* [Video]. YouTube. https://www.youtube.com/watch?v=PVRyMvYVHGQ

zero20vt. (2012, October 1). *3D Paper Illusion - Impossible Gravity Illusion - 1* [Video]. YouTube. https://www.youtube.com/watch?v=tdp0sRUeXt4