# COSC 4370 Homework 2

**Name**: Jordan Foster | **PSID:** 1669900
March, 2022

## 1. Assignment

The goal of this assignment is to recreate several 3D that were provided within the instructions, based on various OpenGL functions an objects, as well as a free-range scene that is to be created without reference.

## 2.1 Method and Implementation for Problem 1

For the initial problem, it was quite apparent that the implementation of a solution would be rather straightforward. However, seeing as I have not had experience working with 3D spaces before, an informative piece was referenced to assist with this particular problem[1]. The image at hand was one of 10 teapots arranged in a circular fashion.

By utilizing the *glPushMatrix* function, each teapot within the scene can be treated and arranged in a rigid and separate manner, preventing them from being haphazardly lumped together and allowing them to be placed in the circular fashion that was needed. The *glRotatef* function was utilized to rotate the teapots along the z-axis, creating the desired tilted effect when paired with rotating the object by increments of 36 degrees. The angle of 36 degrees is based on the fact that there are 10 teapots and 360 degrees in a circle, therefore *360/10 = 36* angles per teapot. Additionally, *glTranslatef* was used to add appropriate space between teapots—to avoid crowded objects—and created the final desired result.

## 2.2 Method and Implementation for Problem 2

The second problem involved recreating an image of a series of steps, much like a staircase. From the first glance, the idea of how to recreate the scene was very apparent: use *glutSolidCubes*, place them in a linear fashion and fill in the space between the cubes to create the solid staircase.

To do this, a simple for loop was used to create the linear incline of the steps, which I counted to be 15 in total. From this point, the job was already half-way done (at least visually). To fill in the space beneath the steps, a nested for loop was used to fill in the rows starting from each step on the incline and ending at the last cube at the apex of the steps. The incline is linear, but uses the y value of 0.025+i/240, with i being the iteration variable within the for loop and

---

[1] Reference: https://community.khronos.org/t/rotating-objects-in-a-double-orbit/70704/2

implemented within the *glTranslatef* function. Containing the rows produced in the for loop between a *glPushMatrix* and *glPopMatrix* call allowed more orderly arrangement in line with the cube placements. This successfully created the solid object displayed with very few hardships.

## 2.3 Method and Implementation for Problem 3

The image displayed for this problem shows a pyramid composed of solid teapots. Initially, the implementation seemed simple: I attempted to create the pyramid row-by-row with a for loop, filling in the teapots as it iterated. However, this method did not scale as I had intended, so I slightly altered it.

By using *glTranslatef* within a for loop, I created a right-side diagonal of teapots. From there, using additional for loops (and changing the initial position with additional *glTranslatef* calls), filling in the rows based on the needed amount of solid teapots was more straightforward. The method used was similar to a typewriter mechanic, filling in the teapots was right to left, followed by left to right, until the end of an incoming diagonal. While not the initial approach I had aspired to use, this still proved to be effective in reproducing the desired image.

## 2.4 Method and Implementation for Problem 4

For this final problem, there was no visual reference to illustrate the finished product; however, there were guidelines provided that made clear that nested *glPushMatrix* calls, as well as a triangle rendered via coordinates were to be utilized in some fashion. With free range to design a scene, I decided to create a sine wave of cubes, each topped with a teapot.

To start, a for loop was created, iterating 16 times just to allow more room to display the wave. There the first *glPushMatrix* call was placed—within it, the sine wave was created with small gaps along with x-axis (with x being set to *i*/3.0f, which I being the iteration variable) and a simple declaration of y being equal to sin(*i*), with *i* being the iteration variable. Beneath the placement of the *glutSolidCube*, the nested *glPushMatrix* call was placed. Within this nested call, the *glTranslatef* function was used to set the origin to a point just above the previously placed cubes. There the teapot would be placed. A call to *glScalef* was used to scale the teapots along the x- and y-axes per iteration, multiplying the iteration variable by 0.075f and 0.25f respectively, elongating the teapot by incrementally increasing amounts. With the teapots placed, the matrix is completed and we use *glPopMatrix* to signal this.

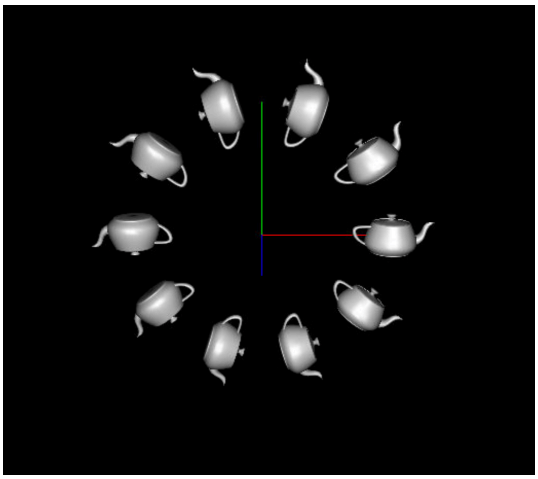
Now, with those calls completed, the only thing left is to render the triangle. To start, a call to *glBegin(GL_TRIANGLES)* is used to signify and declare that the vertices used as input are indeed being used to create a triangle—the *glBegin* function is used to draw such 2D shapes with given parameters. The vertices were added using *glVertex2f* and finally the function was ended

with *glEnd*, which signals that the *glBegin* call has completed. With that, the 2D triangle was rendered behind our teapot-cube sine waves and our scene is complete.

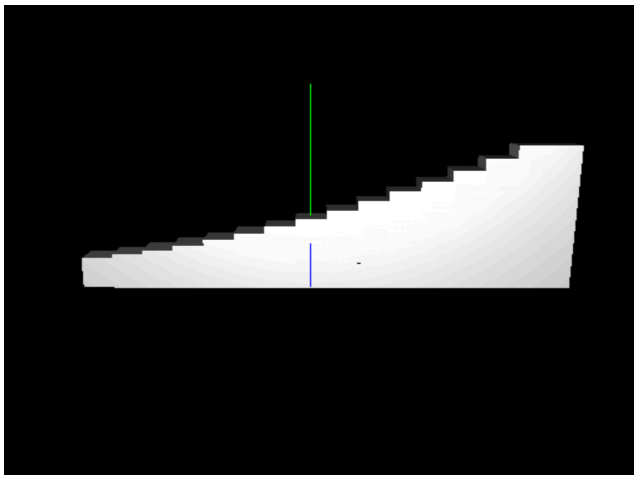
## 3. Result

With all scenes produced and reproduced to a satisfactory level, the program ran smoothly and displayed the following outputs:

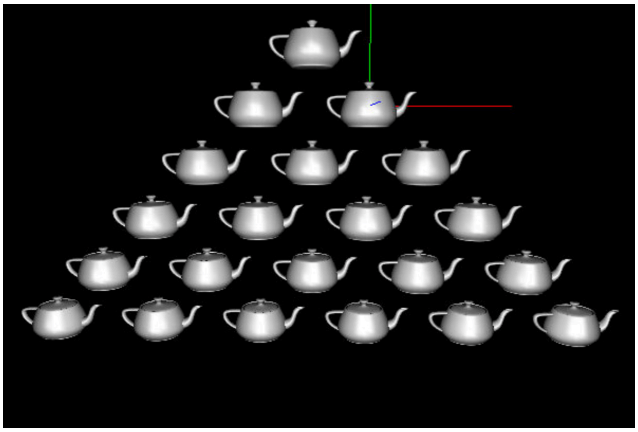
### Problem 1



### Problem 2



### Problem 3



### Problem 4