CS 112 Section H-Assignment-3

Due Date: 29 February 2024
Group Size Maximum 2 Student

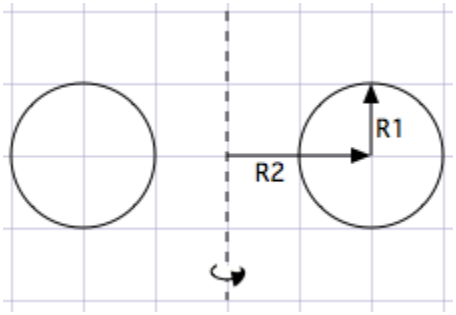
Faculty of Computer Science and Engineering
Ghulam Ishaq Khan Institute of Engineering Sciences and Technology

Spinny Donut Assignment

Write a C++ program to create a spinning donut, similar to this (NiceInsidiousGrub-size_restricted.gif (384×326) (gfycat.com). You need to **define a Donut class** which calculates all the pixel points and then draws the donut onto the console.

The following steps would guide you towards figuring out the solution for this.

1. Create a circle of radius R1 centered at R2.



Before thinking about how to spin the donut, we need to figure out how to draw a static donut first. So we have a circle of radius **R1** centered at point **(R2,0,0)**, drawn on the xy-plane. We can draw this by sweeping an angle — θ — from 0 to 2π:

$$(x, y, z) = (R_2, 0, 0) + (R_1 \cos\theta, R_1 \sin\theta, 0)$$

2. Create a donut by rotating around the Y axis.

Now we take that circle and rotate it around the y-axis by another angle — let's call it φ, using the standard technique i.e. multiply by a rotation matrix. So if we take the previous points and rotate about the y-axis we get:

$$(R_2 + R_1\cos\theta, \quad R_1\sin\theta, \quad 0) \cdot \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix}$$

$$= ((R_2 + R_1\cos\theta)\cos\phi, \quad R_1\sin\theta, \quad -(R_2 + R_1\cos\theta)\sin\phi)$$

3. Spin the donut around the X and Z axis.

We also want the whole donut to spin around on at least two more axes for the animation. Let's denote the rotation about the x-axis by A and the rotation about the z-axis by B.

This can be achieved using:

$$(R_2 + R_1\cos\theta, \quad R_1\sin\theta, \quad 0) \cdot \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos A & \sin A \\ 0 & -\sin A & \cos A \end{pmatrix} \cdot \begin{pmatrix} \cos B & \sin B & 0 \\ -\sin B & \cos B & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

All of the above equations can be boiled down to a simple calculations through the power of Algebra, so instead of calculating the above values using matrix multiplications (which you can still do if you feel like it), you can use the below equations to get x, y, zpoints for a rotating donut. Try to reuse variables wherever possible to reduce computation and memory overhead.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (R_2 + R_1\cos\theta)(\cos B\cos\phi + \sin A\sin B\sin\phi) - R_1\cos A\sin B\sin\theta \\ (R_2 + R_1\cos\theta)(\cos\phi\sin B - \cos B\sin A\sin\phi) + R_1\cos A\cos B\sin\theta \\ \cos A(R_2 + R_1\cos\theta)\sin\phi + R_1\sin A\sin\theta \end{pmatrix}$$

To achieve the rotating effect, you can clear the screen and draw the donut again and again using different values of A and B, where A and B are very slightly incremented (by a few decimal points).

4. Project donut onto 2D screen.

Up to this point, we have been calculating the values of the donut in 3D space, but to show the donut on a screen (which as we know is a 2D environment), we need to project those 3D points into 2D space.

We can use the following equation for 3D to 2D projection:

$$(x', y') = \left(\frac{K_1 x}{K_2 + z}, \frac{K_1 y}{K_2 + z}\right)$$

Where K_1controls the scale (which depends on your pixel resolution) and K_2controls the distance from the viewer to the donut. You can try out different values for these two to see what works best for your configurations. For a screen_size of around 25, I'd recommend $K_1 = (10 \ldots 15)$ and $K_2 = 5$, but it will change depending on your implementation and screen_size.

5. Determine illumination by calculating surface normal.

Now that we know where to put the pixel, we still haven't considered which shade to plot. To calculate illumination, we need to know the surface normal — the direction perpendicular to the surface at each point. If we have that, then we can take the dot product of the surface normal with the light direction, which we can choose arbitrarily. That gives us the cosine of the angle between the light direction and the surface direction: If the dot product is >0, the surface is facing the light and if it's <0, it faces away from the light. The higher the value, the more light falls on the surface.

Luminance value could be calculated using:

$$\cos\phi\cos\theta\sin B - \cos A\cos\theta\sin\phi - \sin A\sin\theta + \cos B(\cos A\sin\theta - \cos\theta\sin A\sin\phi)$$

which ranges from -sqrt(2) to +sqrt(2). If it's < 0, the surface is pointing away from us, so we won't bother trying to plot it. Otherwise, we will plot it according to the luminance intensity, using the following characters

```
".,-~:;=!*#$@"
```

such that the higher the luminance, the higher index character will be used. Multiplying the luminance value by 8 would give us a value between 0 and 11, which can be used to index into the output characters.

Note: Make sure to read the task description carefully and do as required.