

Vv214 Final Project

Donut.c

Pingbang Hu, Xiaoyu Chen, Jinyi Wu

University of Michigan-Shanghai Jiao Tong University Joint Institute

April 28, 2021



JOINT INSTITUTE
交大密西根学院

1. Motivation
2. Introduction
3. Draw a donut
4. Rotate a donut
5. Bright and dark
6. Projection into terminal
7. Discrete Dynamic System
8. Extra
9. Source Code
10. Reference

Back in 2006, there's an interesting c project called Donut.c, which will print a rotating donut in the terminal.

Looking into the source code and a blog updated recently by the author, we find out that this program is highly depend on what we have learned in Vv214.

After some discussion, we decide to dive into the source code and analyze the whole mechanism behind it.

To start with, we first need to design a way to show the donut. Seems the terminal has a dark background with white character.

We let the **size** of the character to mimic the brightness of a single pixel. We show the character array as below, from the darkest to the brightest:

{. , - ; = ! * # \$ @}

With these characters, we can generate some black & white *ascii art* in the terminal!

Draw a donut

Now we have the way to show some diagram in the terminal, then we start to consider how to actually generate a rotating donut in the terminal.

In a mathematical way to produce a donut, we start with the concept of *parametric equation*. Since a donut basically is a circle rotating respected to the axis parallel to the fixed diameter, with some simply geometry, we have:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} R_2 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} R_1 \cos \theta \\ R_1 \sin \theta \\ 0 \end{pmatrix}, \quad \theta \in [0, 2\pi)$$

which produce two circles in the $x - y$ plane.

Draw a donut

Since we now have two circles, we now only need to rotate these circles for π degree by applying a *rotation matrix w.r.t. y-axis*.

With the parametric equation we have derived, we have:

$$\underbrace{\begin{pmatrix} \cos \phi & 0 & \sin \phi \\ 0 & 1 & 0 \\ -\sin \phi & 0 & \cos \phi \end{pmatrix}}_{Y : \text{rotation matrix for y-axis}} \cdot \underbrace{\begin{pmatrix} R_2 + R_1 \cos \theta \\ R_1 \sin \theta \\ 0 \end{pmatrix}}_{\text{two circles' parametric eqs.}}, \quad \begin{cases} \theta \in [0, 2\pi) \\ \phi \in [0, \pi) \end{cases}$$

This produce a full donut in the 3 dimensional space. And for simplicity, we will now omit the angles' ranges from now on.

Draw a donut



Rotate a donut

Now let say we want to rotate this donut respect to x and z axis, we just need to apply the rotation matrix as below:

$$\begin{array}{c} \text{X : rotation matrix for x-axis} \\ \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \psi_1 & \sin \psi_1 \\ 0 & -\sin \psi_1 & \cos \psi_1 \end{pmatrix}} \end{array} \cdot \begin{array}{c} \text{Z : rotation matrix for z-axis} \\ \underbrace{\begin{pmatrix} \cos \psi_2 & \sin \psi_2 & 0 \\ -\sin \psi_2 & \cos \psi_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}} \end{array} \cdot \underbrace{\begin{pmatrix} \cos \phi & 0 & \sin \phi \\ 0 & 1 & 0 \\ -\sin \phi & 0 & \cos \phi \end{pmatrix}}_Y \cdot \begin{pmatrix} R_2 + R_1 \cos \theta \\ R_1 \sin \theta \\ 0 \end{pmatrix}$$

Then now, we have a rotating donut in 3 dimensional space, which rotating speed is now fully controlled by the *rate of change* of angles ψ_1 and ψ_2 !

Now, let us think about how we actually get the brightness for an image. What makes the differences of bright and dark? The **light direction** and the **normal vector** for the surface!

Let us choose a light direction vector, say

$$L := \begin{pmatrix} L_x \\ L_y \\ L_z \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$$

And by simply geometry, the normal vector can be derived from the same way, we first consider a particular θ when creating the donut, we find out that for such a point, say

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} R_2 + R_1 \cos \theta \\ R_1 \sin \theta \\ 0 \end{pmatrix}$$

the normal vector is:

$$N := \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix} = \begin{pmatrix} \cos \theta \\ \sin \theta \\ 0 \end{pmatrix}$$

then we just need to apply the same rotation matrix X and Z , we can get every normal vector on the surface of a donut!

In order to show how much light is directly shot on the particular pixel on a surface, we use **inner product** between the *normal vector* and *light vector*, namely

$$B_{(x,y,z)} := \langle N, L \rangle = L^T N = (L_x, L_y, L_z) \cdot \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix}$$

where N_i and L_i is known on every point for $i = x, y, z$.

We have now derived the complete mathematical equation for the rotating donut! The only thing left is how to let the viewer to *see* the donut in the terminal.

There are two things we need to solve:

- ▶ Move the donut somewhere in front of the viewer(the viewer is at the origin).
- ▶ Project from 3 dimensional space onto our 2 dimensional terminal.

It's easy to obtain a simple relation for the projection, but there will have some problem we need to solve, namely *overlapping*.

We first derive the simple relation and then consider the problem of overlapping.

Discrete Dynamic System



- ▶ <https://www.a1k0n.net/2011/07/20/donut-math.html>
- ▶ https://en.wikipedia.org/wiki/3D_computer_graphics
- ▶ <https://www.javatpoint.com/computer-graphics-z-buffer-algorithm>