```cpp
/*Terminal Donut
                ~Nitin Rohit
This is just the clone project which was first made by Andy Salon and I tried to replicate it after reading the article which was made
by the original maker. I have made use of the same concept of rotation and simple intensity formula.*/

//Include the required header files
#include <iostream>
#include <unistd.h>
#include <math.h>

using namespace std;

//make an array for as an alternative of intensity level from low to high density using ASCII characters
char value[20]=".,-~:;=!*#$@";

//Variable to control the rate of rotation of the toroid(aka donut)
const float r=0.05;

//Variables to get different points in the donut
const float theta_spc=0.07;
const float phi_spc=0.05;

//Variables for the tow radius of the donut
const float R1=1;
const float R2=2;

//Value of distance between viewer and the donut
const float K2=5;

//Dimensions of the screen which the viewer will see the donut
const int scrW=30;
const int scrH=30;

//Value of pi and getting the value of distance where the screen will be placed(here it is such that the donut takes 3/4th the width)
const float pi=M_PI;
const float K1=scrW*K2*3/(8*(R1+R2));

//Initilisating the output and inverseDistance array with default values
void initialise(char (*output)[scrH],float (*zbuffer)[scrH])
{
    for(int i=0;i<scrW;i++)
    for(int j=0;j<scrH;j++)
    {
        output[i][j]=' ';
        zbuffer[i][j]=0;
    }
}

//Function which displays the output array(the array which contains the donut virtually)
void show(char (*output)[scrH])
{
    for(int i=0;i<scrW;i++)
    {
        for(int j=0;j<scrH;j++)
```

```
            cout<<output[i][j];

        cout<<endl;
    }
}

//Function which computes the values of output array with the help of maths and zbuffer array
void display(float A,float B)
{
//Pre-computes the values of sines and cosines for less computational time
    float cA=cos(A),sA=sin(A);
    float cB=cos(B),sB=sin(B);

    char output[scrW][scrH];
    float zbuffer[scrW][scrH];
    initialise(output,zbuffer);

    for(float theta=0;theta<2*pi;theta+=theta_spc)
    {
        float ctheta=cos(theta),stheta=sin(theta);
        for(float phi=0;phi<2*pi;phi+=phi_spc)
        {
            float cphi=cos(phi),sphi=sin(phi);

            float cx=R2+R1*ctheta;
            float cy=R1*stheta;

            float x=cx*(cB*cphi+sA*sB*sphi)-cy*(cA*sB);//using rotation formula we can get the following equations
            float y=cx*(sB*cphi-sA*cB*sphi)+cy*(cA*cB);
            float z=K2+cA*cx*sphi+cy*sA;
            float zi=1/z;//calculating the z inverse as it will be used in the 3d to 2d conversion formula

            int xp=(scrW/2+K1*zi*x);//the screen size is halfed because it is the centre(origin in cartesian coordinate)
            int yp=(scrH/2-K1*zi*y);//the value is negated here because in terminal the y-axis is down-positive

            float L=cphi*ctheta*sB-cA*ctheta*sphi-sA*stheta+cB*(cA*stheta-ctheta*sA*sphi);//formula for value of intensity(how bright it is)

            if(L>0)
            {
                if(zi>zbuffer[xp][yp])
                {
                    zbuffer[xp][yp]=zi;
                    int lvalue=L*8;//multipling with 8, because it makes its range from 11 to 0(which is what we require)
                    output[yp][xp]=value[lvalue];
                }
            }

        }
    }
    show(output);
    cout<<"\x1b[H";//brings the pointer to the start of the terminal
}

```

```cpp
int main()
{
    for(float n=0;;n+=r)
    {
        display(2*n,n);//giving the input values using "for" loop with some increment everytime
        usleep(5);//this function delays the time in miliseconds(controls the frame rate or how fast the donut rotates)
    }
}
```