Answer Sheet Session 4

To be given 1 week before session 4. Students try out the problems first. Then they see this solution and compare it with theirs. We discuss the problems in session 4.

Reminder to TRY IT YOURSELF FIRST

Note that this answer sheet is just a guide to show how things can turn out. The real submission would be the pictures of the results viewed in CURA.

Also note that you should only see this sheet if you CANNOT come up with a code that works at all. Otherwise, your CURA images are enough for a submission.

In fact, the point is to make improvements to the code and make something better than the given code. Put your own spin on it.

This course made to show real work. We start with something relatively simple with minimal instruction and self-learn with online resources. Then we try and keep doing more and more complex things until we have something substantial!

The only difference in real life work is that you don’t get the answer sheet. You make your own!

By the end of this internship, you should add something of your own above and beyond what is mentioned in the answer sheets. That is when you graduate and get ready for the real world!

# Session 4: Polygons, curves and fills

## Housekeeping functions

Make global functions to take the name of the file from the user. This way, we can quickly and easily change file names for different actions being done.

Note: put these functions in a ses4headers.h file and have the main function in a separate file.

### Header file Ses4headers.h

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include <math.h>

using namespace std;

string gcodeSes4; // Filename to write the gcode in

string line; // read and write lines

fstream fileRead; // File to read top and bottom

fstream fileMain; // File to write into

const double pi = 3.14159;

const double er = 0.013161;

void read\_top\_file()

{

fileRead.open("top\_filler.txt", ios::in);

fileMain.open(gcodeSes4, ios::out); // truncate for the top fill

cout<<"Reading from top filler file\n";

while (getline(fileRead,line))

{

fileMain<<line<<endl;

cout<<line<<endl;

}

fileRead.close();

cout<<"\nTop filler file finished\n";

fileMain.close();

}

void read\_bottom\_file()

{

fileRead.open("bottom\_filler.txt", ios::in);

fileMain.open(gcodeSes4, ios::app); // append to add bottom

cout<<"\nReading from bottom filler file\n";

fileMain<<endl; // give some extra space

while (getline(fileRead,line))

{

fileMain<<line<<endl;

cout<<line<<endl;

}

fileRead.close();

cout<<"\nBottom filler file finished";

fileMain.close();

}

### Function to check if the print is within boundaries

int check\_bounds(double xcen, double ycen, double rad\_eq) // checks if it is within the bounds of the print bed

{ // rad\_eq is the equivalent radius regardless of the kind of shape.

// note: the boundaries for Ultimaker S5 printer are:

// x (10,320) since bed size is 330 AND y (10,230) since bed size is 240

double xup, yup, xd, yd;

int is\_ok = 0;

xup = xcen + rad\_eq;

yup = ycen + rad\_eq;

xd = xcen - rad\_eq;

yd = ycen - rad\_eq;

if (xup>320)

cout<<"\nThe x value exceeds boundary\n";

else if (yup>230)

cout<<"\nThe y value exceeds boundary\n";

else if (xd<10)

cout<<"\nThe x value gets too low\n";

else if (yd<10)

cout<<"\nThe y value gets too low\n";

else

{

cout<<"\nThe print is within boundaries\n";

is\_ok = 1;

}

return is\_ok;

}

### Function to make Polygons

double make\_polygon(int numsides, double xcen, double ycen, double side, double e)

{

// function to make a regular polygon at given center xcen,ycen number of sides and side length

// need the initial extrusion position and the final extrusion position needs to be returned

double r, theta, ang, x, y, zraise; // r = circum radius along which all points are. xy coordinates are for writing it on

// zraise is the z height we raise to when doing the moves

theta = pi/numsides; // theta = (360/numsides)/2 = 180/numsides or pi/numsides in radians

r = side/(2\*sin(theta));

cout<<"\nMaking Polygon\n";

fileMain.open(gcodeSes4, ios::app); // append to add the polygon

x = xcen + r;

y = ycen;

zraise = z + 2; // raise 2mm up from the current level

fileMain<<"\nG1 F1000 Z"<<zraise; // raise for move

fileMain<<"\nG1 F1500 X"<<x<<" Y"<<y<<" E"<<e; // go to the starting point with fast move

fileMain<<"\nG1 F1200 Z"<<z; // back to current level and ready for printing speed

// cout<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e;

for(int i = 1; i<=numsides; i++)

{

ang = (i\*2\*pi)/numsides; // angle for current point. ends at 2\*pi (0 again)

x = xcen + r\*cos(ang);

y = ycen + r\*sin(ang);

e = e + side\*er; // add to extrusion value

fileMain<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e; // writing all the points

cout<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e;

}

fileMain<<endl<<endl; // leave gap after completion

fileMain.close();

return e; // return the e value

}

### Main file ses4main.cpp

#include "ses4headers.h"

int main ()

{

double x0,y0,xc,yc,s,e0,radeq; // xy coordinates and extrusion value and x and y increment for array

int nsides, checkbed;

cout<<"Enter the name of the file: ";

cin>>line;

gcodeSes4 = line + ".gcode";

// Reading Top File

read\_top\_file();

// write middle section of code

e0 = 0;

x0 = 10;

y0 = 10;

z = 0.2; // current z value

fileMain.open(gcodeSes4, ios::app);

fileMain<<"\nG1 F1200 X"<<x0<<" Y"<<y0<<" E"<<e0<<" \n"; // setting original position and speed

fileMain.close();

checkbed = 0;

while(checkbed == 0)

{

cout<<"\nEnter number of sides of the polygon: ";

cin>>nsides;

cout<<"\nEnter the x and y coordinates of the polygon center: ";

cin>>xc>>yc;

cout<<"\nEnter side length of the polygon: ";

cin>>s;

radeq = s/(2\*sin(pi/nsides));

cout<<"\n equivalent radius is: "<<radeq<<endl;

checkbed = check\_bounds(xc,yc,radeq);

cout<<"\n checkbed is: "<<checkbed<<endl;

}

e0 = make\_polygon(nsides,xc,yc,s,e0);

checkbed = 0;

while(checkbed == 0)

{

cout<<"\nEnter the x and y coordinates of the center of the circle: ";

cin>>xc>>yc;

cout<<"\nEnter the radius of the circle: ";

cin>>s;

checkbed = check\_bounds(xc,yc,s);

cout<<"\n checkbed is: "<<checkbed<<endl;

}

e0 = make\_circle(xc,yc,s,e0);

cout<<"\n\nThe final e0 value is: "<<e0<<endl;

//Reading Bottom File

read\_bottom\_file();

}

## Making curves Circle and Ellipse

Starting with a distance calculation function:

### Distance function

double dist\_calc(double oldp[2], double newp[2]) // points old x1,y1 and new x2,y2

{

double dist, xgap, ygap;

xgap = newp[0] - oldp[0];

ygap = newp[1] - oldp[1];

dist = sqrt((xgap\*xgap) + (ygap\*ygap)); // dist = sqrt(dx^2 + dy^2)

return dist;

}

### Circle Function

double make\_circle(double xcen, double ycen, double radius, double e)

{

// function to make a circle of given radius

double theta, ang, x, y, op[2], np[2], d, zraise; // old n new points to calc distance

int num = 100;

theta = pi/num; // we use a 100 points (100 sided polygon) to make a circle

cout<<"\nMaking Circle\n";

fileMain.open(gcodeSes4, ios::app); // append to add the circle

x = xcen + radius;

y = ycen;

op[0] = x;

op[1] = y;

zraise = z + 2; // raise 2mm up from the current level

fileMain<<"\nG1 F1000 Z"<<zraise; // raise for move

fileMain<<"\nG1 F1500 X"<<x<<" Y"<<y<<" E"<<e; // go to the starting point with fast move

fileMain<<"\nG1 F1200 Z"<<z; // back to current level and ready for printing speed

// cout<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e;

for(int i = 1; i<=num; i++)

{

ang = (i\*2\*pi)/num; // angle for current point. ends at 2\*pi (0 again)

x = xcen + radius\*cos(ang);

y = ycen + radius\*sin(ang);

np[0] = x;

np[1] = y;

d = dist\_calc(op,np);

e = e + d\*er; // add to extrusion value

op[0] = x;

op[1] = y; // rewrite the values for next loop

fileMain<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e; // writing all the points

cout<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e;

}

fileMain<<endl<<endl; // leave gap after completion

fileMain.close();

return e; // return the e value

}

### Ellipse Function

double make\_ellipse(double xcen, double ycen, double a, double b, double e)

{

// function to make an ellipse with given major and minor axes a and b

double theta, ang, x, y, op[2], np[2], d, zraise; // old n new points to calc distance

int num = 100;

theta = pi/num; // we use a 100 points (100 sided polygon) to make a circle

cout<<"\nMaking Ellipse\n";

fileMain.open(gcodeSes4, ios::app); // append to add the circle

x = xcen + a;

y = ycen;

op[0] = x;

op[1] = y;

zraise = z + 2; // raise 2mm up from the current level

fileMain<<"\nG1 F1000 Z"<<zraise; // raise for move

fileMain<<"\nG1 F1500 X"<<x<<" Y"<<y<<" E"<<e; // go to the starting point with fast move

fileMain<<"\nG1 F1200 Z"<<z; // back to current level and ready for printing speed

// cout<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e;

for(int i = 1; i<=num; i++)

{

ang = (i\*2\*pi)/num; // angle for current point. ends at 2\*pi (0 again)

x = xcen + a\*cos(ang);

y = ycen + b\*sin(ang);

np[0] = x;

np[1] = y;

d = dist\_calc(op,np);

e = e + d\*er; // add to extrusion value

op[0] = x;

op[1] = y; // rewrite the values for next loop

fileMain<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e; // writing all the points

//cout<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e;

}

fileMain<<endl; // leave gap after completion

fileMain.close();

return e; // return the e value

}

### Function call in main function for ellipse

checkbed = 0;

while(checkbed == 0)

{

cout<<"\nEnter the x and y coordinates of the center of the ellipse: ";

cin>>xc>>yc;

cout<<"\nEnter the major and minor axes of the ellipse: ";

cin>>a\_ax>>b\_ax;

radeq = a\_ax;

if(b\_ax>a\_ax)

radeq = b\_ax;

checkbed = check\_bounds(xc,yc,radeq); // just use the longer axis

cout<<"\n checkbed is: "<<checkbed<<endl;

}

## Filling functions

### Filling of a circle

double fill\_circ(double e)

{ // w is the extrusion line width or gap between 2 extusions that we use to make a filled shape

double d, radius, r, xc, yc, w;

int checkbed = 0,i = 1;

while(checkbed == 0)

{

cout<<"\nEnter the x and y coordinates of the center of the circle: ";

cin>>xc>>yc;

cout<<"\nEnter the radius of the circle: ";

cin>>radius;

cout<<"\nEnter the width of the printing: ";

cin>>w;

checkbed = check\_bounds(xc,yc,radius);

cout<<"\n checkbed is: "<<checkbed<<endl;

}

r = radius;

while(r>w)

{

e = make\_circle(xc,yc,r,e);

cout<<"\n\nE value after "<<i<<"th circle: "<<e<<endl<<endl;

r = r-w;

cout<<"\n\nRadius value after "<<i<<"th circle: "<<r<<endl<<endl;

i++;

}

cout<<"\nMaking the inner dot\n";

if(r>0)

{

d = (pi\*r\*r)/w; // effective length = area/extrusion width

cout<<"\nEffective length is: "<<d<<endl;

e = e + d\*er; // amount to purge in the center

fileMain.open(gcodeSes4, ios::app); // append to add the center point

fileMain<<"\nG1 X"<<xc<<" Y"<<yc; // reach center first

fileMain<<"G1 E"<<e; // purging in the center only

fileMain<<endl; // leave gap after completion

fileMain.close();

}

return e; // return the e value

}

### Filling of an Ellipse

double fill\_ellipse(double e)

{

double d, a\_ax, b\_ax, r , a, b, radeq, xc, yc, w, x, y;

int checkbed = 0,i = 1, flip = 0;

while(checkbed == 0)

{

cout<<"\nEnter the x and y coordinates of the center of the ellipse: ";

cin>>xc>>yc;

cout<<"\nEnter the major and minor axes of the ellipse: ";

cin>>a\_ax>>b\_ax;

cout<<"\nEnter the width of the printing: ";

cin>>w;

radeq = a\_ax;

if(b\_ax>a\_ax)

radeq = b\_ax;

checkbed = check\_bounds(xc,yc,radeq); // just use the longer axis

cout<<"\n checkbed is: "<<checkbed<<endl;

}

a = a\_ax;

b = b\_ax;

r = b;

if (a<b)

{

r = a;

flip = 1;

}

while(r>w)

{

e = make\_ellipse(xc,yc,a,b,e);

cout<<"\n\nE value after "<<i<<"th ellipse: "<<e<<endl<<endl;

a = a-w;

b = b-w;

r = b;

if(flip)

r = a;

cout<<"\n\nShorter axis value after "<<i<<"th ellipse: "<<r<<endl<<endl;

i++;

}

cout<<"\nMaking the inner line\n";

if(r>(w/2))

{

d = (pi\*a\*b)/w; // effective length = area/extrusion width. Area of ellipse = pi\*ab

cout<<"\nEffective length is: "<<d<<endl;

e = e + d\*er; // amount to purge in the center

// point to use

d = a\_ax - b\_ax; // effective point of contact

if (flip)

d = b\_ax - a\_ax;

d = d-w; //give little more room

x = xc + d;

y = yc;

if(flip)

{

y = yc + d;

x = xc;

}

fileMain.open(gcodeSes4, ios::app); // append to add the center point

fileMain<<"\nG1 X"<<x<<" Y"<<y; // reach one focus point

x = xc - d;

if (flip)

{

y = yc - d;

x = xc;

}

fileMain<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e; // move to the other focus point while spreading this

fileMain<<endl; // leave gap after completion

fileMain.close();

}

return e; // return the e value

}

### Filling of Polygon

double fill\_polygon(double e)

{

double radeq,d, xc, yc, w, x, y, side, r;

int checkbed = 0,i = 1, nsides;

while(checkbed == 0)

{

cout<<"\nEnter the x and y coordinates of the center of the polygon: ";

cin>>xc>>yc;

cout<<"\nEnter the number of sides of the polygon: ";

cin>>nsides;

cout<<"\nEnter the side length of the polygon: ";

cin>>side;

cout<<"\nEnter the width of the printing: ";

cin>>w;

radeq = side/(2\*sin(pi/nsides));

cout<<"\n equivalent radius is: "<<radeq<<endl;

checkbed = check\_bounds(xc,yc,radeq);

cout<<"\n checkbed is: "<<checkbed<<endl;

}

r = side/(2\*tan(pi/nsides)); // radius of the incircle

while(r>w)

{

e = make\_polygon(nsides,xc,yc,side,e);

cout<<"\n\nE value after "<<i<<"th polygon: "<<e<<" and incircle radius is: "<<r;

r = r-w; // reducing incircle

side = r\*2\*tan(pi/nsides); // defining side using incircle radius

cout<<"\n\nSide lenght after "<<i<<"th polygon: "<<side<<" and incircle radius is: "<<r<<endl;

i++;

}

cout<<"\nMaking the inner point\n";

if(r>0)

{

d = (r\*side\*nsides)/(2\*w); // effective length = area/extrusion width. Area of polygon = 1/2\*r\*side\*nsides

cout<<"\nEffective length is: "<<d<<endl;

e = e + d\*er; // amount to purge in the center

fileMain.open(gcodeSes4, ios::app); // append to add the center point

fileMain<<"\nG1 X"<<xc<<" Y"<<yc<<" E"<<e; // go to center and extrude that on the way

fileMain<<endl; // leave gap after completion

fileMain.close();

}

return e; // return the e value

}

### Main Function Call for All

// write middle section of code

e0 = 0;

x0 = 10;

y0 = 10;

z = 0.2; // current z value

fileMain.open(gcodeSes4, ios::app);

fileMain<<"\nG1 F1200 X"<<x0<<" Y"<<y0<<" E"<<e0<<" \n"; // setting original position and speed

fileMain.close();

checkbed = 0;

cont = 'y';

while(cont == 'y')

{

cout<<"\nWhat shape would you like to fill?";

cout<<"\n1. Circle \n2. Ellipse \n3. Polygon \nEnter the choice number: ";

cin>>ch;

if (ch == 1)

e0 = fill\_circ(e0);

if (ch == 2)

e0 = fill\_ellipse(e0);

if (ch == 3)

e0 = fill\_polygon(e0);

cout<<"\nWould you like to continue? (y/n) ";

cin>>cont;

}

cout<<"\n\nThe final e0 value is: "<<e0<<endl;

//Reading Bottom File