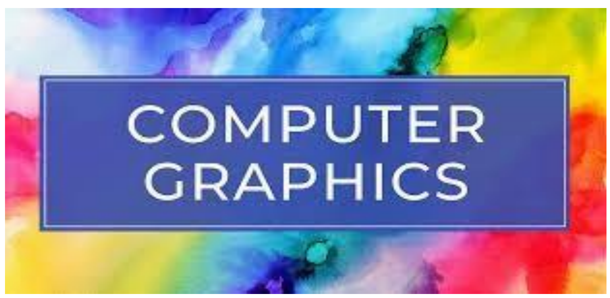
LNMIIT
The LNM Institute of Information Technology

Computer Graphics

Project Report



Submitted To: Dr. Aloke Datta

Submitted By:

Krishan Kumar Agrawal	20ucs100
Joshal Joshi	20ucs092
Yash Jain	20ucs237

Contents

OBJECTIVE

The aim of the project is to create a digital cartoon character using the knowledge of computer graphics.

INTRODUCTION

In this project, we are going to create an image of “Homura Akemi”, a character from thepopular anime Magic Girl “Madoka”. For this project, we will be using python and its Turtle module.

The Turtle module in Python provides a simple interface for drawing shapes on the screen. It has a robotic turtle which starts at (0, 0) on the screen. This turtle can be manipulated using trivial commands such as forward, backward, rotate, goto and many more. Hence, it can be used to create complex shapes and drawings quite easily.

Code: https://github.com/jYash2309/CG-Project

TURTLE LIBRARY

`Turtle` is a Python feature like a drawing board, which lets us command a turtle to draw all over it!

We can use functions like **turtle.forward(…)** and **turtle.right(…)** which can move the turtle around.

Method	**Parameter**	**Description**
Turtle()	None	Creates and returns a new turtle object
forward()	amount	Moves the turtle forward by the specified amount
backward()	amount	Moves the turtle backward by the specified amount
right()	angle	Turns the turtle clockwise
left()	angle	Turns the turtle counterclockwise
penup()	None	Picks up the turtle's Pen
pendown()	None	Puts down the turtle's Pen
up()	None	Picks up the turtle's Pen
down()	None	Puts down the turtle's Pen
color()	Color name	Changes the color of the turtle's pen

fillcolor()	Color name	Changes the color of the turtle will use to fill a polygon
heading()	None	Returns the current heading
position()	None	Returns the current position
goto()	x, y	Move the turtle to position x,y
begin_fill()	None	Remember the starting point for a filled polygon
end_fill()	None	Close the polygon and fill with the current fill color
dot()	None	Leave the dot at the current position
stamp()	None	Leaves an impression of a turtle shape at the current location
shape()	shapename	Should be 'arrow', 'classic', 'turtle' or 'circle'

Thus, using turtle and its vast drawing methods, we created a drawing of the famous cartoon(anime) character for our project.

Turtle handled the drawing very precisely and in a nice manner.One can install the turtle library on a system by command:

```
pip install turtle
```

SETUP

Imports

First up, we will be importing the necessary modules.

```
import turtle as te
from typing import Tuple
```

Global Declarations

Now, we need to declare a few global variables that we will require as well as configure the turtle drawing environment

```
Step_write = 500       # Sampling times of Bezier function
Speed = 1
W = 600            # Interface Width
H = 500            # Interface Height
HX, YH = 0, 0         # Record the handle of the previous Bessel function
```

- Step_write is the number of time steps for Bezier curve
- Speed is the speed of turtle
- W (Width) and H (Height) are the dimensions of the turtle window
- HX and YH are used for smooth Bezier curve

```
te.tracer(10)
te.setup(W, H, 0, 0)
te.setworldcoordinates(0, H, W, 0)
te.pensize(1)
te.speed(Speed)
te.penup()
```

- te.tracer is used to set the turtle to draw on every 10th refresh of screen
- te.setup set the turtle window
- te.setworldcoordinates sets the actual drawing area. It is set such that top-left is the origin and right & down are positive x & y axes, respectively.
- te.pensize sets the thickness of stroke

DRAWING PRIMITIVES

For drawing our actual figure we first need a few primitive methods to draw portions of the image.

m_t

This function simply moves the turtle from current position to the given position without drawing the path

```
def m_t(point):
    te.penup()
    te.goto(point)
```

Bzp(bezier point)

This function takes a list of control points and the time step and returns the point on the curve at that time step. It uses recursion to continuously reduce the list of points until only one point remains.

```
def bzp(pts, time, dim=2):  # Bezier Point
    deg = len(pts) - 1
    if(deg == 0):
        return pts[0]
    else:
        n_p = []
        for idx in range(deg):
            temp = []
            for d in range(dim):
                temp.append(pts[idx][d] * (1.0 - time) +
                            pts[idx + 1][d] * time)
            n_p.append(tuple(temp))
        return bzp(tuple(n_p), time, dim)
```

bzc(Bezier curve)

This function gets all the control points for the curve and then draws the curve. It iterates over the time steps and draws the point for each one.

```
def bzc(pts, dim=None):     # Bezier Curve
    if(dim is None):
        dim = len(pts[0])

    m_t(pts[0])
    te.pendown()
    for time in range(0, Step_write + 1):
        p = bzp(pts, time / Step_write, dim)
        te.goto(p)
    te.penup()
```

Bzc_through(bezier_curve_through)

The bezier_curve_through method draws a bezier curve through the given points along with the current position as the first point.

It also takes an optional parameter **rel(relative)** (default: false). If it is true, the points are taken as relative to the current position and, therefore, are first converted to absolutepoints by adding the current position to them. Otherwise, they remain the same.

Then, it inserts the current position to the list of points and calls the method for drawing the Bezier curve.

```python
def bzc_through(pts, rel: bool = False):  # Bezier Through
    global HX
    global YH
    curr = te.position()
    pts = list(pts)
    if(rel):
        for i in range(len(pts)):
            pts[i] = tuple([c1 + c2 for c1, c2 in zip(pts[i], curr)])
    pts.insert(0, curr)
    bzc(tuple(pts))
    HX = pts[-1][0] - pts[-2][0]
    YH = pts[-1][1] - pts[-2][1]
```

Smooth_bzc(smooth_bezier_curve)

This function creates a bezier curve between the given points while smoothly connecting it to the last curve drawn. Similar to the previous method, it also takes an optional parameter **rel** which performs the same action as seen before.

```python
def smooth_bzc(pts, rel: bool = False):   # Smooth Bezier curve
    global HX
    global YH
    pts = list(pts)
    pts.insert(0, (HX + te.position()[0], YH + te.position()[1]))
    if(rel):
        pts[0] = (pts[0][0] - te.position()[0],
                  pts[0][1] - te.position()[1])
    bzc_through(tuple(pts), rel)
```

l_b(line between)

Given two points; source and destination, draw a line connecting source to destination.

```
def l_b(src, dst):
    m_t(src)
    te.pendown()
    te.goto(dst)
```

l_d(line displacement)

Given certain displacement, draw a line from the current position to the displaced position.

```
def l_d(disp):
    l_b(te.position(), te.position() + disp)
```

l_t(lineto)

Given the destination point, draw a line from the current position to that destination.

```
def l_t(dst):
    l_b(te.position(), dst)
```

hz_t(horizontal_to)

Takes a destination x coordinate and draws a horizontal line from current position to destination point.

```
def hz_t(dst_x):
    l_b(te.position(), (dst_x, te.ycor()))
```

hz_d(horizontal_displace)

Takes a displacement in the x direction and draws the line.

```
def hz_d(dx):
    l_b(te.position(), te.position() + (dx, 0))
```

Vrt_d(vertical_displace)

Takes a displacement in the y direction and draws the line.

```
def vrt_d(dy):
    l_b(te.position(), te.position() + (0, dy))
```

polyline(poly_l)

This method takes some points p_1, p_2, p_3,..., p_n as input and draws the lines p_1 to p_2, p_2 to p_3,..., p_{n-1} to p_n.

```
def poly_l(pts):
    total = len(pts)
    for idx in range(total - 1):
        l_b(pts[idx], pts[idx + 1])
```

IMPLEMENTATION

Now we can use the built up functions to draw the final image.

Below code consist of listed functions :

1. bzc_through
2. smooth_bzc
3. m_t
4. l_d
5. l_t

```
# Coat
te.color("black", "#F2F2F2")
m_t((61, 462))
te.begin_fill()
smooth_bzc(((12, -41), (27, -58)), rel=True)
bzc_through(((-6, -36), (6, -118), (9, -132)), rel=True)
bzc_through(((-15, -27), (-23, -51), (-26, -74)), rel=True)
bzc_through(((4, -66), (38, -105), (65, -149)), rel=True)
hz_t(486)
bzc_through(((12, 24), (40, 99), (33, 114)), rel=True)
bzc_through(((39, 82), (55, 129), (39, 144)), rel=True)
smooth_bzc(((-31, 23), (-39, 28)), rel=True)
smooth_bzc(((-12, 37), (-12, 37)), rel=True)
l_d((50, 92))
hz_t(445)
smooth_bzc(((-29, -38), (-31, -46)), rel=True)
smooth_bzc(((78, -107), (72, -119)), rel=True)
smooth_bzc(((355, 178), (340, 176)))
smooth_bzc(((272, 63), (264, 64)))
smooth_bzc(((-29, 67), (-27, 73)), rel=True)
smooth_bzc(((99, 292), (174, 428), (173, 439)))
smooth_bzc(((-8, 23), (-8, 23)), rel=True)
l_t((61, 462))
```

Below code consist of listed functions :

1. bzc_through
2. smooth_bzc
3. Pencolour
4. Polyline
5. Pencolor
6. Begin_fill
7. end_fill

```
m_t((60.5, 461.5))
te.color("black", "#D3DFF0")
te.begin_fill()
bzc_through(((0, 0), (17, -42), (27, -59)), rel=True)
bzc_through(((-6, -33), (6, -128), (10, -133)), rel=True)
bzc_through(
    ((-15, -10), (-27, -66), (-27.285, -75)), rel=True)
te.pencolor("#D3DFF0")
bzc_through(((12.285, 11), (82.963, 156),
                         (82.963, 156)), rel=True)
te.pencolor("black")
smooth_bzc(((12.322, 75), (19.322, 86)), rel=True)
bzc_through(((-1, 11), (-8, 25), (-8, 25)), rel=True)
hz_t(60.5)
te.end_fill()

m_t((444.5, 464))
te.begin_fill()
bzc_through(((0, 0), (-29, -36), (-31, -46)), rel=True)
smooth_bzc(((53.59, -82.337), (53.59, -82.337)), rel=True)
te.pencolor("#D3DFF0")
smooth_bzc(((86.41, -47.663), (96.072, -54.85)), rel=True)
smooth_bzc(((563.5, 297.5), (570.5, 299.5), (518.5, 334)))
te.pencolor("black")
bzc_through(((-2, 16), (-12, 33), (-12, 37)), rel=True)
smooth_bzc(((50, 92), (50, 93)), rel=True)
hz_t(444.5)
te.end_fill()

m_t((195, 49))
te.begin_fill()
te.pencolor("#D3DFF0")
poly_l(((195, 49), (175.5, 106.5), (202.522, 49)))
te.pencolor("black")
hz_t(195)
```

Below code consist of listed functions :

1. l_b
2. Polyline
3. m_t
4. bzc_through

```
# Wrinkles
te.pencolor("black")
l_b((94.5, 397.5), (107.5, 373.5))
l_b((122.5, 317.5), (95.875, 274.699))
l_b((122.5, 341.5), (141.5, 402.5))
l_b((141.5, 409.5), (153.5, 431.5))
l_b((340.023, 49), (360.5, 144))
l_b((478.5, 95.5), (518.5, 161.5))
l_b((518.5, 332.5), (460.5, 359.5))
poly_l(((506.5, 369.5), (493.5, 402.5), (502.5, 443.5)))
m_t((530, 429))
bzc_through(((4, 16), (-5, 33), (-5, 33)), rel=True)
```

Below code consist of listed functions :

1. smooth_bzc
2. bzc_through
3. Begin_fill
4. l_t
5. h_t
6. m_t

```
# Inside of jacket
te.color("black", "#2b1d2a")
m_t((225, 462))
te.begin_fill()
hz_t(165)
smooth_bzc(((9, -15), (8, -25)), rel=True)
bzc_through(((-47, -126), (6, -212), (12, -225)), rel=True)
smooth_bzc(((185, 305), (202, 428), (225, 462)))
l_t((225, 462))
te.end_fill()

m_t((390, 462))
te.begin_fill()
bzc_through(
    ((10, -23), (34, -180), (35, -222)), rel=True)
bzc_through(((7, 4), (54, 45), (61, 61)), rel=True)
smooth_bzc(((-73, 101), (-72, 118)), rel=True)
bzc_through(((5, 15), (31, 46), (31, 45)), rel=True)
l_t((390, 462))
te.end_fill()

"""
Layer 3
"""

# Inside of jacket
te.color("black", "#2b1d29")
m_t((225, 462))
te.begin_fill()
bzc_through(((-28, -50), (-40, -166), (-40, -250)), rel=True)
bzc_through(((6, 51), (-6, 87), (45, 106)), rel=True)
smooth_bzc(((64, 27), (89, 24)), rel=True)
smooth_bzc(((49, -18), (56, -20)), rel=True)
smooth_bzc(((50, -10), (51, -85)), rel=True)
bzc_through(((0, 29), (-25, 201), (-36, 225)), rel=True)
l_t((225, 462))
te.end_fill()
```

Below code consist of listed functions :

1. bzc_through
2. smooth_bzc
3. Vrt_d
4. L_d
5. L_t
6. end_fill

```
# Clothes
te.color("black", "#3D3D3D")
m_t((225, 462))
te.begin_fill()
bzc_through(((-5, -5), (-22, -53), (-23, -70)), rel=True)
l_d((32, -13))
bzc_through(((3, -25), (6, -28), (12, -36)), rel=True)
smooth_bzc(((13, -12), (16, -12)), rel=True)
vrt_d(-2)
bzc_through(((45, 20), (64, 14), (94, 1)), rel=True)
vrt_d(2)
bzc_through(((8, -2), (15, 2), (17, 4)), rel=True)
smooth_bzc(((0, 6), (-2, 9)), rel=True)
bzc_through(((10, 10), (10, 29), (11, 33)), rel=True)
smooth_bzc(((23, 4), (25, 6)), rel=True)
smooth_bzc(((-17, 83), (-17, 78)), rel=True)
l_t((225, 462))
te.end_fill()
```

Below code consist of listed functions :

1. Vrt_d
2. Bzc_through
3. Smooth_bzc
4. h_d
5. Vrt_d
6. l_d
7. begin_fill

```
te.color("black", "#968281")
m_t((262, 329))
te.begin_fill()
vrt_d(17)
bzc_through(((1, 2), (44, 14), (45, 15)), rel=True)
smooth_bzc(((3, 12), (3, 12)), rel=True)
hz_d(3)
vrt_d(-5)
bzc_through(((1, -3), (4, -6), (5, -7)), rel=True)
l_d((36, -14))
bzc_through(((1, -1), (3, -16), (2, -17)), rel=True)
smooth_bzc(((318, 348), (296, 344), (262, 329)))
te.end_fill()
```

Below code consist of listed functions :

1. bzc_through
2. Smooth_bzc
3. Line_displace
4. End_fill
5. l_t
6. m_t

```
te.color("black", "#E7F1FF")
m_t((225, 462))
te.begin_fill()
l_d((-3, - 5))
bzc_through(((0, -2), (4, -4), (5, -6)), rel=True)
smooth_bzc(((16, 3), (19, -8)), rel=True)
smooth_bzc(((0, -7), (0, -11)), rel=True)
smooth_bzc(((5, -8), (9, -5)), rel=True)
smooth_bzc(((19, -8), (19, -11)), rel=True)
smooth_bzc(((6, -7), (6, -7)), rel=True)
smooth_bzc(((7, -2), (9, -4)), rel=True)
l_d((41, -2))
l_d((12, 9))
smooth_bzc(((3, 15), (7, 18)), rel=True)
smooth_bzc(((15, 4), (17, 4)), rel=True)
smooth_bzc(((4, -4), (6, -4)), rel=True)
smooth_bzc(((6, 4), (5, 9)), rel=True)
smooth_bzc(((0, 9), (0, 9)), rel=True)
smooth_bzc(((1, 7), (7, 6)), rel=True)
smooth_bzc(((8, 0), (8, 0)), rel=True)
l_d((-2, 8))
l_t((225, 462))
te.end_fill()

te.pensize(2)
m_t((240, 450))
smooth_bzc(((0, 9), (3, 12)), rel=True)
m_t((372, 462))
bzc_through(((-2, -4), (-5, -29), (-7, -28)), rel=True)
te.pensize(1)
```

Below code consist of listed functions :

1. Smooth_bcz
2. L_d
3. Bzc_through
4. Begin_fill
5. M_t
6. End_fill
7. h_d

```
    te.color("black", "#A2B8D6")
    m_t((262, 331))
    te.begin_fill()
    bzc_through(((0, 8), (-1, 13), (0, 15)), rel=True)
    smooth_bzc(((43, 14), (45, 15)), rel=True)
    l_d((3, 12))
    hz_d(3)
    smooth_bzc(((-1, -3), (0, -5)), rel=True)
    l_d((5, -7))
    l_d((36, -14))
    bzc_through(((1, -1), (2, -12), (2, -15)), rel=True)
    smooth_bzc(((25, -2), (15, 13)), rel=True)
    bzc_through(((-2, 4), (-7, 29), (-7, 32)), rel=True)
    smooth_bzc(((-35, 19), (-41, 22)), rel=True)
    smooth_bzc(((-9, 14), (-12, 14)), rel=True)
    smooth_bzc(((-7, -12), (-14, -15)), rel=True)
    bzc_through(((-19, -2), (-41, -25), (-41, -25)), rel=True)
    smooth_bzc(((-10, -26), (-10, -30)), rel=True)
    smooth_bzc(((255, 332), (262, 331)))
    te.end_fill()

    m_t((262, 346))
    l_d((-12, -6))
    m_t((369, 333))
    bzc_through(((2, 4), (-6, 10), (-15, 14)), rel=True)
```

Below code consist of listed functions :

1. bzc_through
2. l_d
3. H_d
4. Smooth_bzc
5. begin_fill

```
te.color("black", "#151515")
m_t((247, 358))
te.begin_fill()
bzc_through(((-5, 3), (-8, 20), (-6, 23)), rel=True)
bzc_through(((25, 21), (50, 17), (50, 17)), rel=True)
l_d((-23, 64))
hz_d(22)
smooth_bzc(((1, -13), (2, -16)), rel=True)
l_d((13, -50))
bzc_through(((2, 2), (7, 3), (10, 1)), rel=True)
smooth_bzc(((18, 65), (18, 65)), rel=True)
hz_d(19)
l_d((-24, -65))
bzc_through(((21, 5), (39, -10), (44, -13)), rel=True)
bzc_through(((5, -20), (1, -21), (0, -24)), rel=True)
bzc_through(((-18, -2), (-49, 15), (-52, 17)), rel=True)
smooth_bzc(((-11, -3), (-15, -1)), rel=True)
smooth_bzc(((252, 356), (247, 358)))
te.end_fill()
```

Below code consist of listed functions :

1. bzc_through
2. Smooth_bzc
3. End_fill
4. M_t
5. Begin_fill
6. l_d

```
te.color("black", "#A2B8D6")
m_t((297, 387))
te.begin_fill()
l_d((-11, 6))
bzc_through(((-1, 0), (-20, -7), (-30, -19)), rel=True)
smooth_bzc(((259, 373), (297, 385), (297, 387)))
te.end_fill()

m_t((323, 384))
te.begin_fill()
l_d((8, 7))
l_d((30, -14))
bzc_through(((1, -1), (5, -6), (4, -7)), rel=True)
smooth_bzc(((329, 379), (323, 384)))
te.end_fill()
```

Below code consist of listed functions :

1. bzc_through
2. Smooth_bzc
3. End_fill
4. Begin_fill
5. m_t

```
te.color("black", "#F3EEEB")
m_t((185, 212))
te.begin_fill()
bzc_through(((4, -9), (46, -77), (52, -75)), rel=True)
bzc_through(((-2, -17), (19, -68), (27, -73)), rel=True)
bzc_through(((16, 15), (71, 108), (76, 112)), rel=True)
smooth_bzc(((76, 53), (86, 60)), rel=True)
bzc_through(((0, 65), (-27, 75), (-31, 76)), rel=True)
bzc_through(((-50, 28), (-70, 30), (-85, 30)), rel=True)
smooth_bzc(((-77, -22), (-86, -26)), rel=True)
smooth_bzc(((180, 302), (186, 228), (185, 212)))
te.end_fill()
```

Below code consist of listed functions :

1. Bzc_through
2. Smooth_bzc
3. Line_to
4. Begin_fill
5. End_fill
6. m_t

```
te.color("black", "#2B1D29")
m_t((189, 202))
te.begin_fill()
bzc_through(((-1, 22), (19, 51), (19, 51)), rel=True)
smooth_bzc(((-10, -42), (7, -92)), rel=True)
smooth_bzc(((212, 168), (196, 189), (189, 202)))
te.end_fill()

m_t((221, 155))
te.begin_fill()
bzc_through(((-2, 6), (5, 48), (5, 48)), rel=True)
smooth_bzc(((18, -28), (20, -48)), rel=True)
bzc_through(((-5, 24), (4, 43), (7, 50)), rel=True)
bzc_through(((-10, -49), (3, -72), (13, -106)), rel=True)
bzc_through(((-2, -7), (-3, -32), (-3, -35)), rel=True)
bzc_through(((-17, 18), (-27, 71), (-27, 71)), rel=True)
l_t((221, 155))
te.end_fill()

m_t((264, 64))
te.begin_fill()
bzc_through(((-4, 5), (14, 100), (14, 100)), rel=True)
smooth_bzc(((-6, -79), (-5, -85)), rel=True)
bzc_through(((0, 98), (49, 139), (49, 139)), rel=True)
smooth_bzc(((8, -50), (3, -65)), rel=True)
smooth_bzc(((272, 64), (264, 64)))
te.end_fill()

m_t((342, 176))
te.begin_fill()
bzc_through(((-1, 27), (-10, 57), (-10, 57)), rel=True)
smooth_bzc(((20, -33), (17, -54)), rel=True)
l_t((342, 176))
te.end_fill()
```

Below code consist of listed functions :

1. l_d
2. Bzc_through
3. Smooth_bzc
4. End_fill
5. M_t
6. Pencolor
7. Pensize

```
te.color("black", "#D1D1D1")
te.pensize(2)
m_t((206, 212))
te.begin_fill()
l_d((15, -7))
bzc_through(((4, -1), (26, -2), (30, 0)), rel=True)
smooth_bzc(((10, 3), (12, 7)), rel=True)
te.pencolor("#D1D1D1")
te.pensize(1)
smooth_bzc(((2, 27), (-1, 30)), rel=True)
smooth_bzc(((-39, 5), (-44, 1)), rel=True)
smooth_bzc(((206, 212), (206, 212)))
te.end_fill()

m_t((384, 204))
te.begin_fill()
te.pencolor("black")
te.pensize(2)
bzc_through(((-3, -1), (-18, -1), (-28, 1)), rel=True)
smooth_bzc(((-9, 6), (-10, 9)), rel=True)
te.pencolor("#D1D1D1")
te.pensize(1)
smooth_bzc(((3, 18), (6, 23)), rel=True)
smooth_bzc(((38, 6), (40, 4)), rel=True)
smooth_bzc(((10, -9), (13, -22)), rel=True)
te.pencolor("black")
te.pensize(2)
l_t((384, 204))
te.end_fill()
```

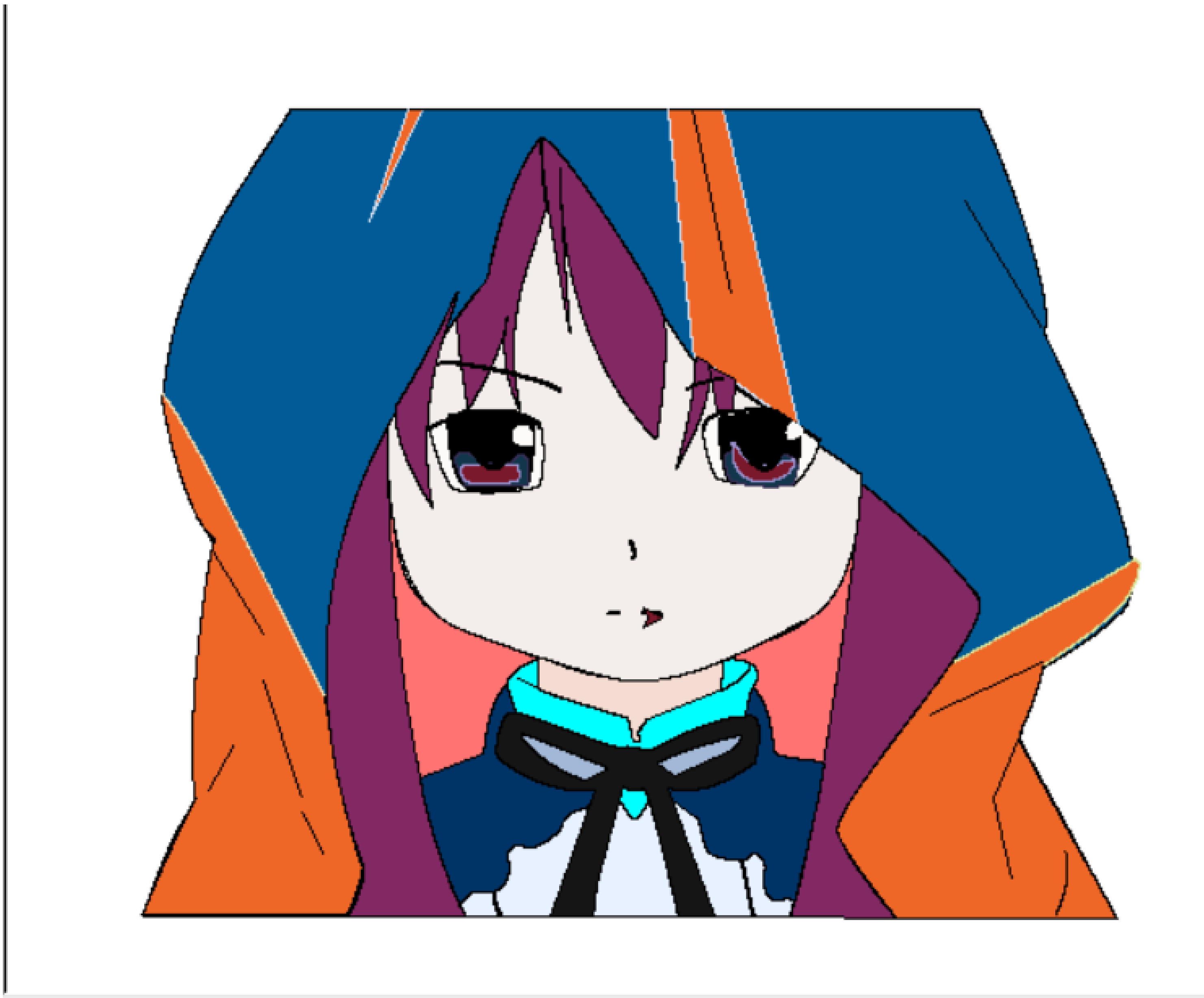
Below code consist of listed functions :

1. M_t
2. l_b
3. bzc_through
4. Smooth_bzc
5. Pencolor

```
te.pencolor("black")
m_t((309, 270))
bzc_through(((0, 0), (4, 7), (1, 9)), rel=True)
l_b((296.5, 307.5), (303.5, 307.5))
m_t((315, 307))
smooth_bzc(((10, -1), (10, 2)), rel=True)

# Wait for the user to click to exit
te.exitonclick()
```

FINAL RESULT



CONCLUSION

Through this project, we learned about various computer graphics concepts and how to implement them. We learned about Bezier curves & line drawing algorithms as well as about drawing in Python using the Turtle module.

REFERENCES

1. https://en.wikipedia.org/wiki/Bresenham's_line_algorithm
2. https://en.wikipedia.org/wiki/B%C3%A9zier_curve
3. https://www.geeksforgeeks.org/bresenhams-line-generation-algorithm/
4. https://www.pythonsandbox.com/docs/turtle
5. https://docs.python.org/3/library/turtle.html