VISUALIZATION ON COORDINATE PLANE

A.Sunny Kumar
CED18I001

How to Run the code

To run the code: **python project.py**

Instructions to Use:

- Use right mouse button to get various drawing options:
  a. Plotting a point
  b. Drawing a line
  c. Circle
  d. To select points and draw a polygon connecting those points
  e. exit
- Press “**R**” to reset the drawing screen
- Press “**T**” and enter a point in the format of “**x y**” to shift the center of the window to point (x,y)
- Use “**Z**” to zoom IN and “**O**” to zoom OUT
- “**U**” key works as undo option

Press “**D**” to draw any shape:

Followed by the name of the shape you want to draw and the parameters required in the given format.

For **circle** - `Circle|cx cy| radius|[0, 0, 0]`

For **ellipse** - `Ellipse|cx cy| xL| yL|[0,0,0]|`

For **line** - `Line|p1|p2|color|size`

For **parabola** - `parabola|p1|p2|[0,0,0]|size`

For **hyperbola** - `Hyperbola|cx cy| a| b|limitX|[0, 0, 0]`

And then press “**enter**” key.

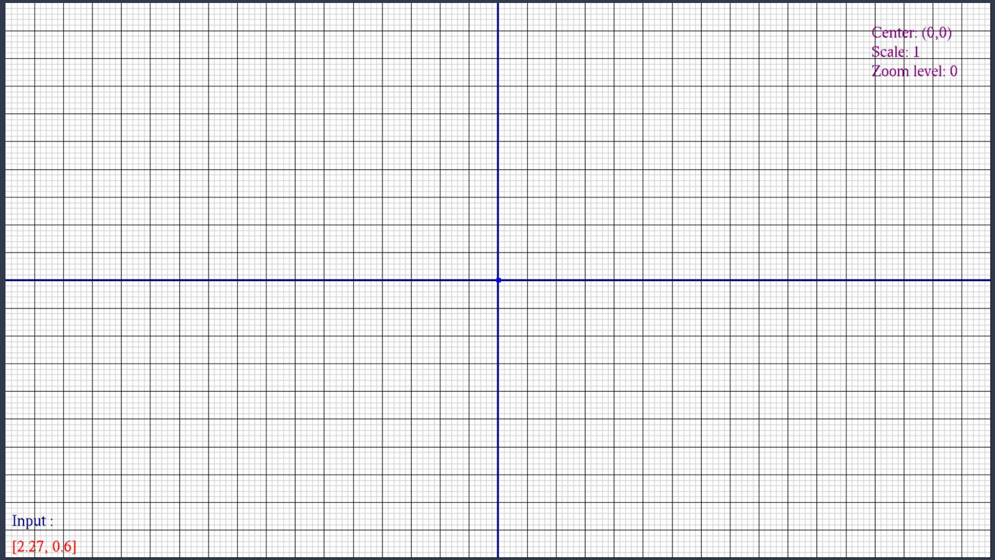
CONCEPT

The objective of this project to make a simple application which enables the user to draw different shapes from the user interface itself and helps them to visualize the drawing on a Coordinate plane using OpenGL for a better visualization of computer graphics.

It has user interaction using both keyboard and mouse.

IMPLEMENTATION

- Drawing coordinate plane
- The point on the left bottom of the screen dynamically shows the position of the mouse cursor.
- Zoom level, current scale of both x and y axis ,current center of the window were displayed on the top right of the screen



Center: (0,0)
Scale: 1
Zoom level: 0

Input :
[2.27, 0.6]

Functions used

- glTranslatef()
- glRotatef()
- glutBitmapCharacter()
- glutPostRedisplay()

Algorithms used

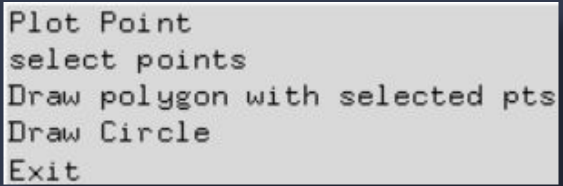
- Mid-point parabola
- Mid-point Hyperbola
- Circle drawing algorithm

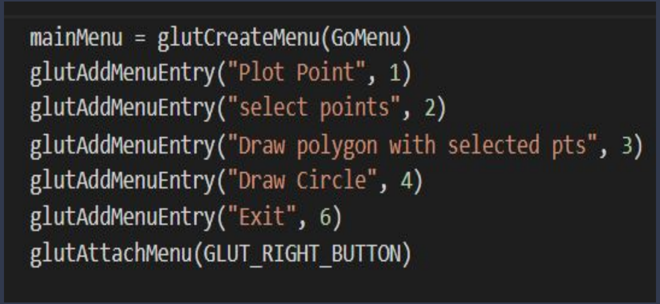
Mouse Function

```python
def mouse(*args):
    if len(args) == 2:
        mposX = round((c2[0] + ((-17 + args[0] * (34/1535))/sc)), 2)
        mposY = round((c2[1] + ((10 - args[1] * (20/840))/sc)), 2)
        mosX = args[0]
        mosY = args[1]
    else:
        mposX = round((c2[0] + ((-17 + args[2] * (34/1535))/sc)), 2)
        mposY = round((c2[1] + ((10 - args[3] * (20/840))/sc)), 2)
        mouseState = args[1]
        mosX = args[2]
        mosY = args[3]
    mousePosString = str([mposX, mposY])
    glutPostRedisplay()
    glutSwapBuffers()
    print("mouse args", args)
```

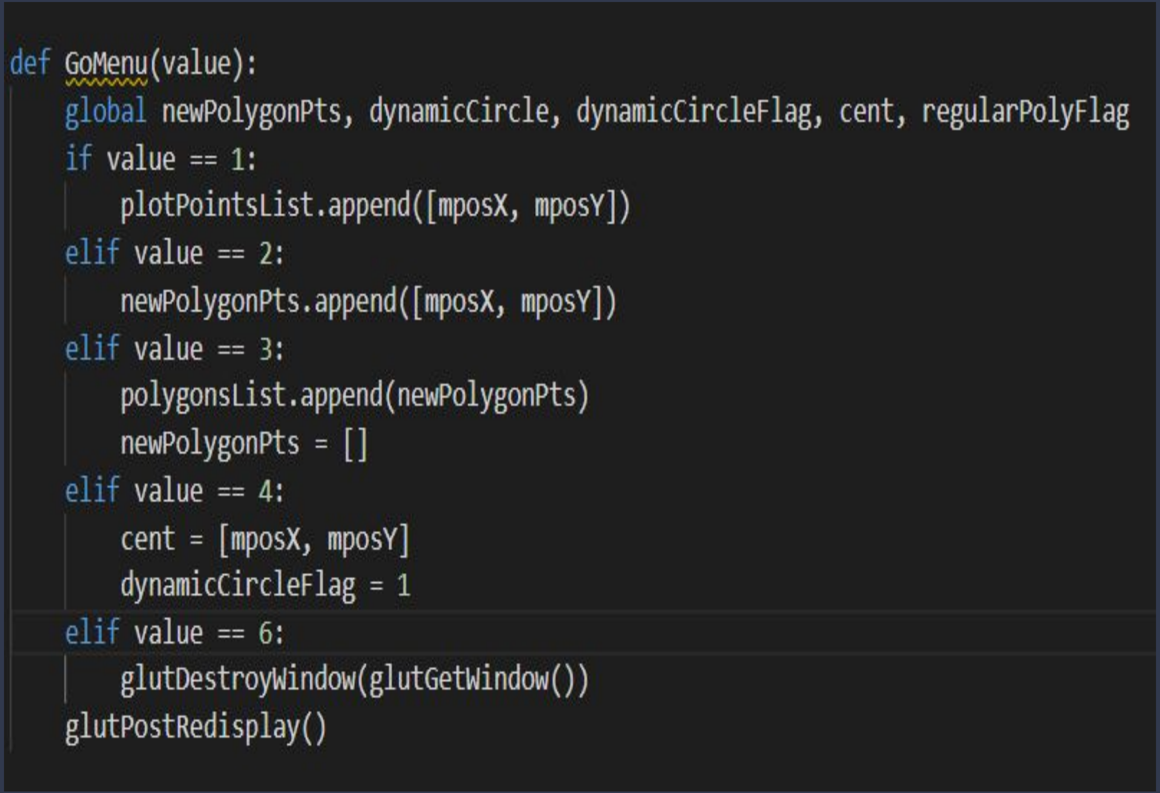
Mouse Menu Implementation

Implementation of menu using mouse for better user experience.





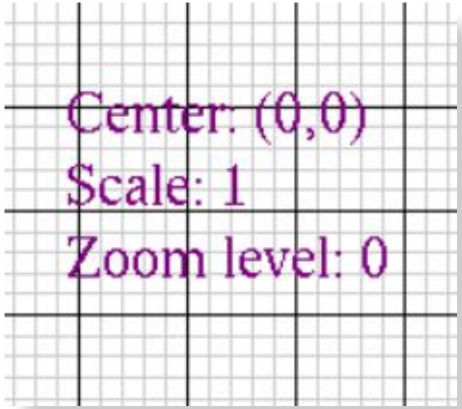
```
mainMenu = glutCreateMenu(GoMenu)
glutAddMenuEntry("Plot Point", 1)
glutAddMenuEntry("select points", 2)
glutAddMenuEntry("Draw polygon with selected pts", 3)
glutAddMenuEntry("Draw Circle", 4)
glutAddMenuEntry("Exit", 6)
glutAttachMenu(GLUT_RIGHT_BUTTON)
```



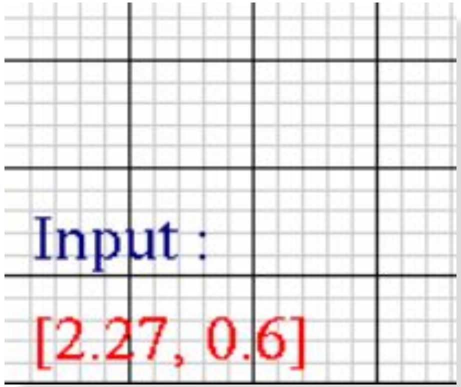
```
def GoMenu(value):
    global newPolygonPts, dynamicCircle, dynamicCircleFlag, cent, regularPolyFlag
    if value == 1:
        plotPointsList.append([mposX, mposY])
    elif value == 2:
        newPolygonPts.append([mposX, mposY])
    elif value == 3:
        polygonsList.append(newPolygonPts)
        newPolygonPts = []
    elif value == 4:
        cent = [mposX, mposY]
        dynamicCircleFlag = 1
    elif value == 6:
        glutDestroyWindow(glutGetWindow())
    glutPostRedisplay()
```

Glut print

```
def glut_print(x, y, font, text,
c=[1, 0, 0]):
    blending = False
    if glIsEnabled(GL_BLEND):
        blending = True
    glColor3f(c[0], c[1], c[2])
    glWindowPos2f(x, y)
    for ch in text:
        glutBitmapCharacter(font,
ctypes.c_int(ord(ch)))
    if not blending:
        glDisable(GL_BLEND)
```

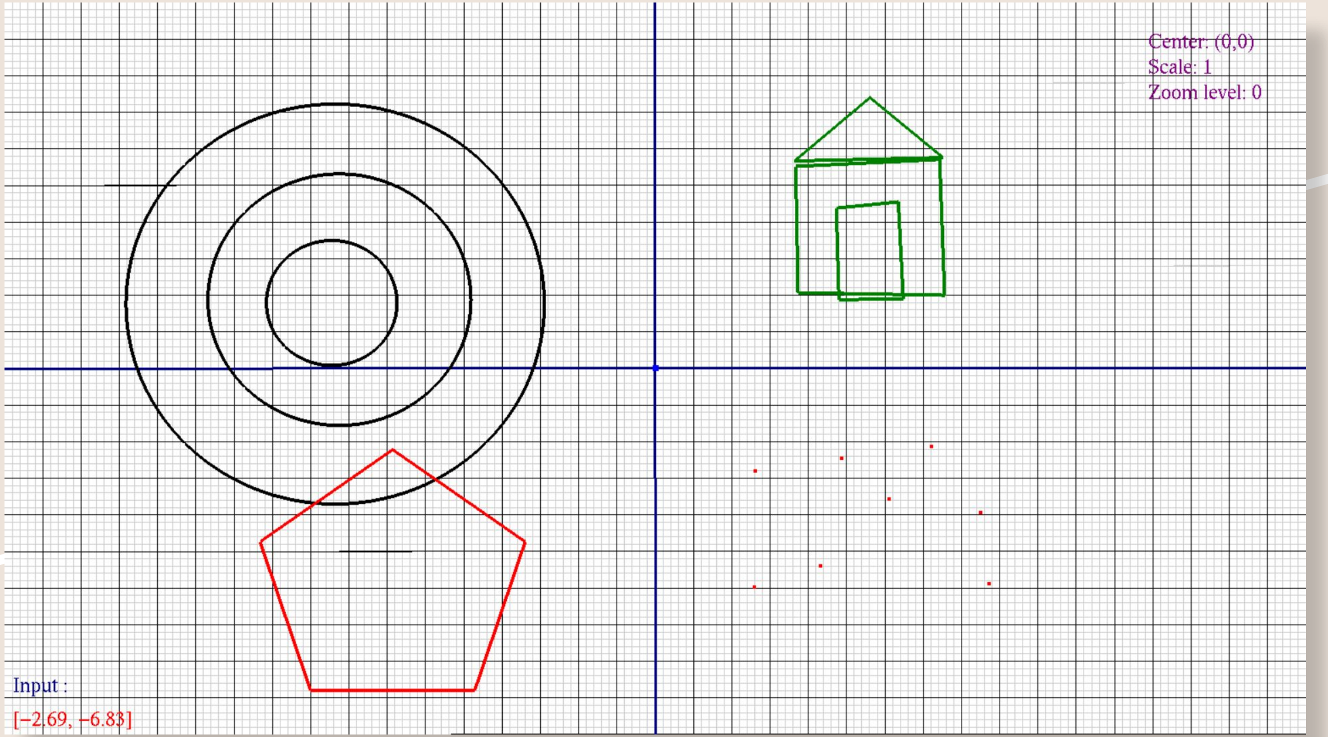


```
Center: (0,0)
Scale: 1
Zoom level: 0
```



```
Input :
[2.27, 0.6]
```

OUTPUT



THANK YOU