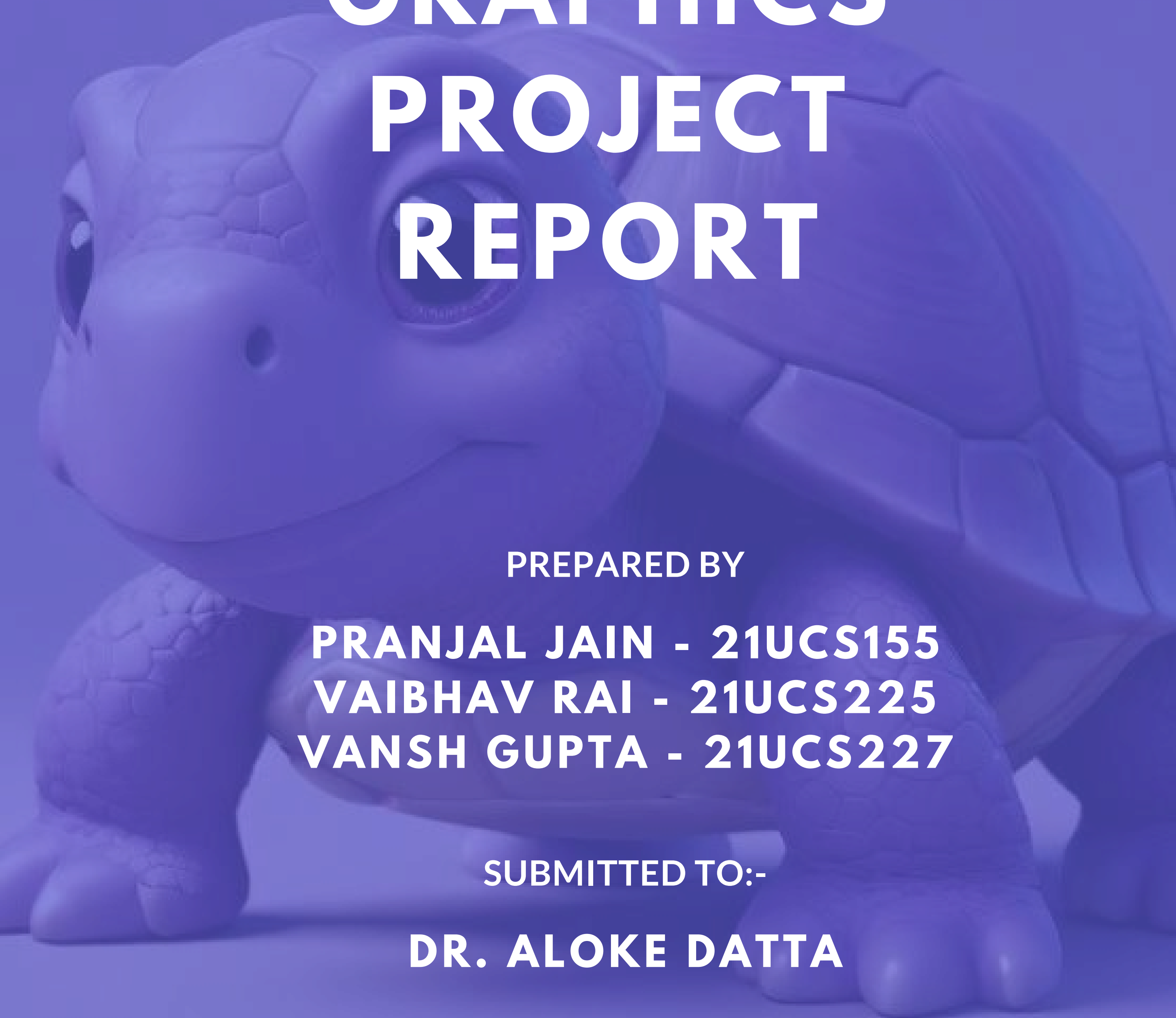
COMPUTER GRAPHICS PROJECT REPORT



PREPARED BY

PRANJAL JAIN - 21UCS155
VAIBHAV RAI - 21UCS225
VANSH GUPTA - 21UCS227

SUBMITTED TO:-

DR. ALOKE DATTA

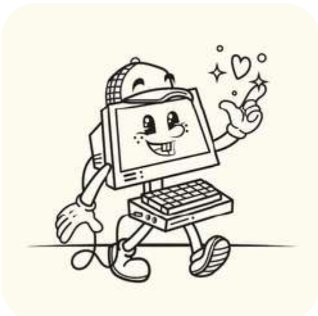
Github Link:-
https://github.com/Pranjal-231003/Computer-Graphics-Spongobob-Squarepaints-Through-Turtle

CONTENTS

INTRODUCTION

Objective:- Make your favorite cartoon character using the knowledge of Computer Graphics

In the realm of Computer Graphics, line and polygon drawing algorithms like Bresenham's Line Drawing Algorithm are utilized to render figures. These algorithms serve to approximate line segments on discrete graphical platforms such as pixel-based screens and printers. The process of line sketching on such media requires approximation, particularly in complex cases. Basic methods are employed to rasterize lines onto the screen or printing surface, typically in a single color.



TURTLE LIBRARY

The turtle library in Python provides a simple and intuitive way to create graphics and drawings using a turtle metaphor, allowing users to control a virtual turtle to draw on a canvas.

Method	Parameter	Description
Turtle()	None	Creates and returns a new turtle object
speed()	speed	Sets the drawing speed of the turtle
penup()	None	Picks up the turtle's Pen
pendown()	None	Puts down the turtle's Pen
pensize()	width	Sets the width of the turtle's pen
color()	Color name	Changes the color of the turtle's pen
begin_fill()	None	Marks the beginning of a filled area for the turtle to draw

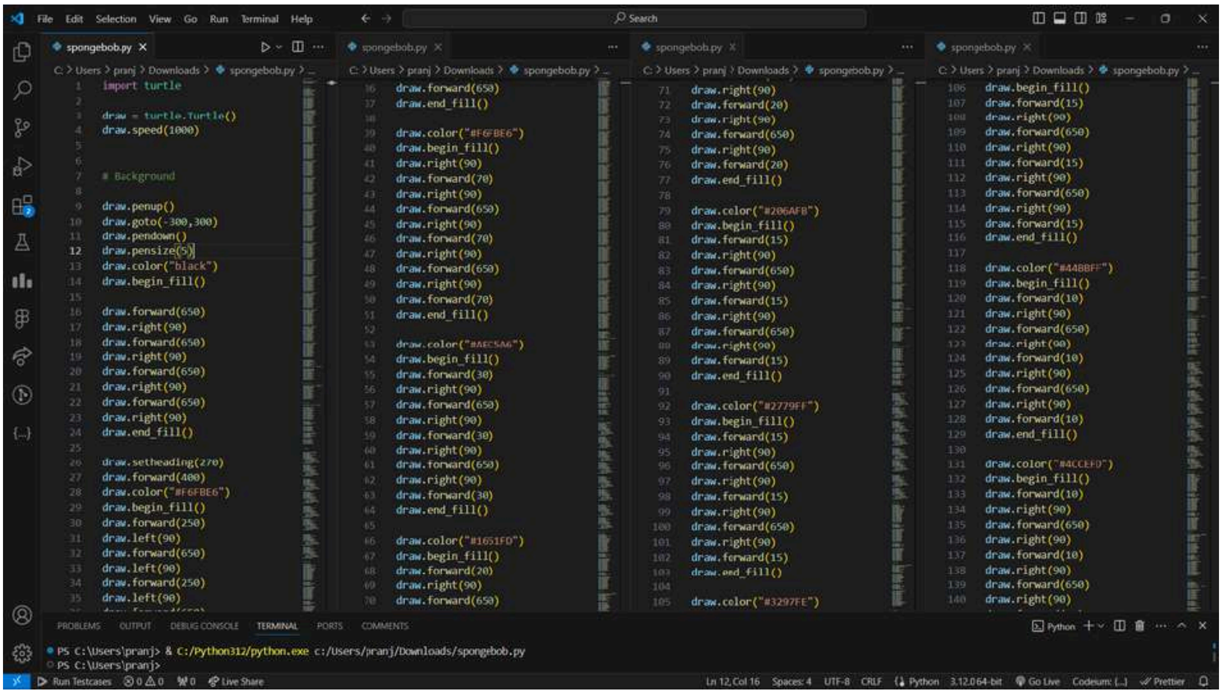
Method	Parameter	Description
end_fill()	None	Marks the end of a filled area and fills it with the current color
goto()	x, y	Moves the turtle to the specified coordinates
setheading()	angle	Sets the orientation of the turtle to the specified angle
forward()	distance	Moves the turtle forward by the specified distance
left()	angle	Turns the turtle counterclockwise by the specified angle
right()	angle	Turns the turtle clockwise by the specified angle
circle()	radius, extent	Draws a circle with the specified radius and extent
done()	None	Stops the turtle graphics window and waits for the user to close it

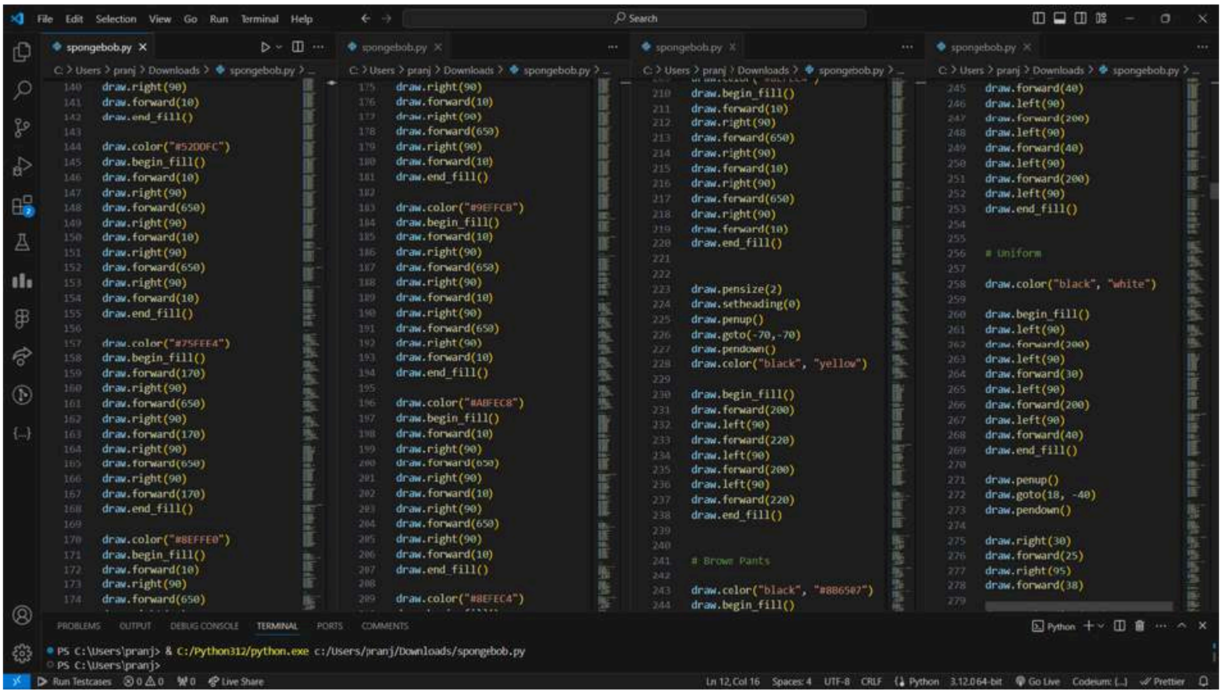
Thus, using turtle and its vast drawing methods, we created a drawing of the famous cartoon character **“Spongobob Squarepants”** for our project.
Turtle handled the drawing very precisely and in a nice manner. One can install the turtle library on a system by command:

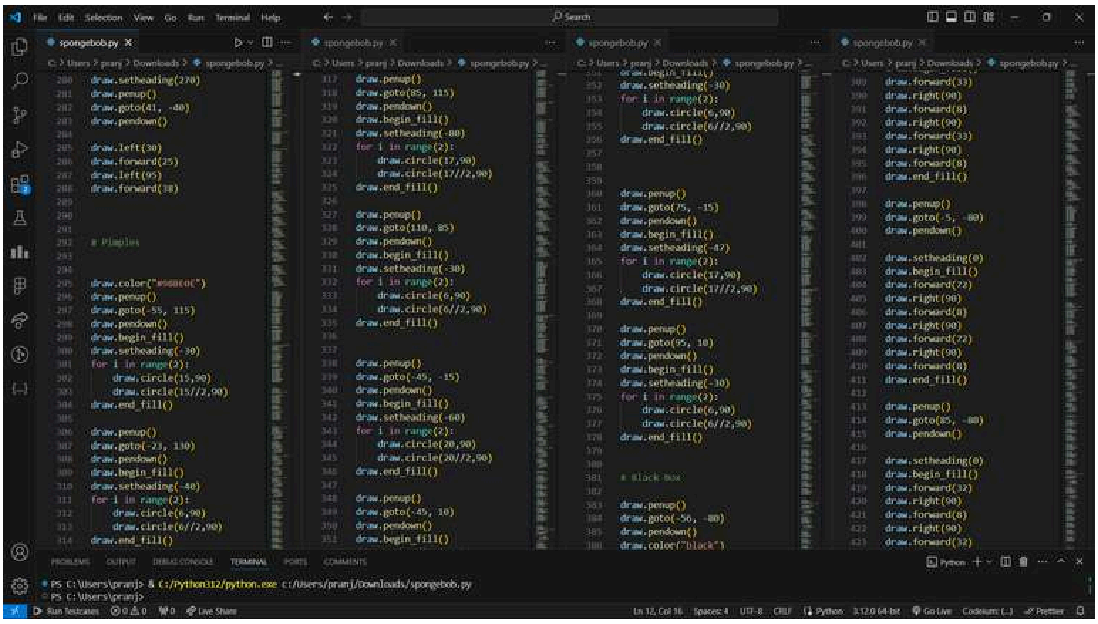
>>pip install turtle

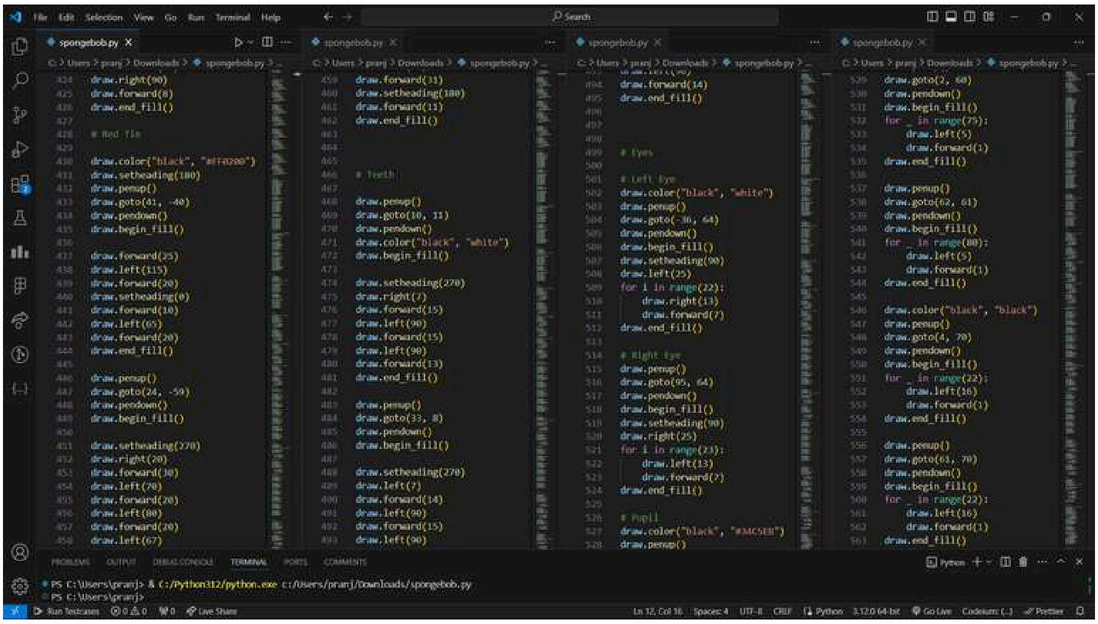
IMPLEMENTATION

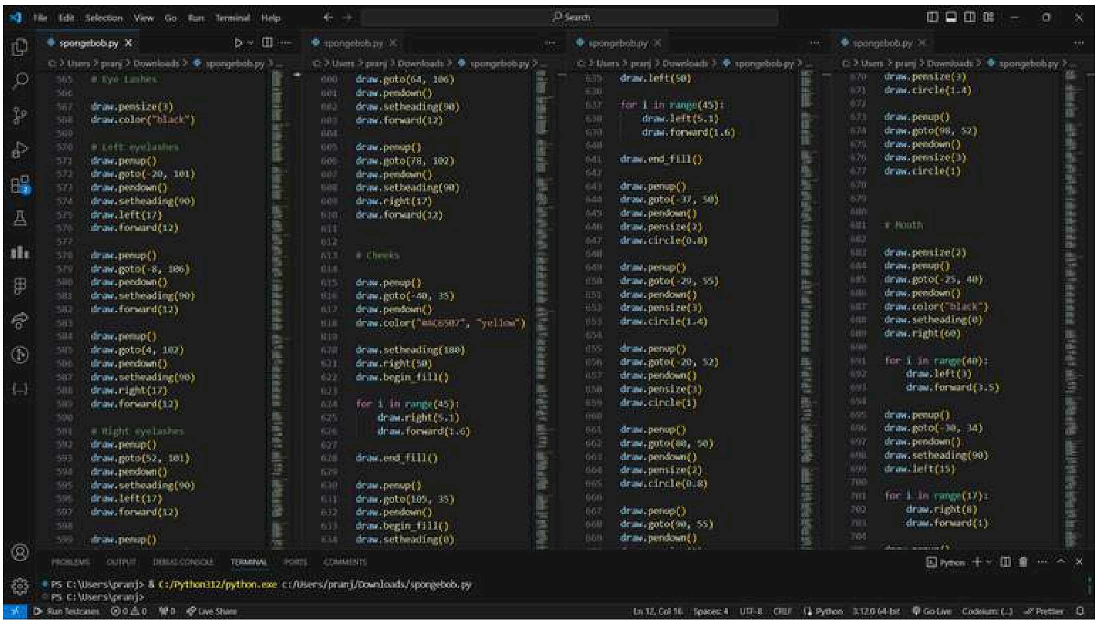
This is the code used to create “Spongebob Squarepants”.

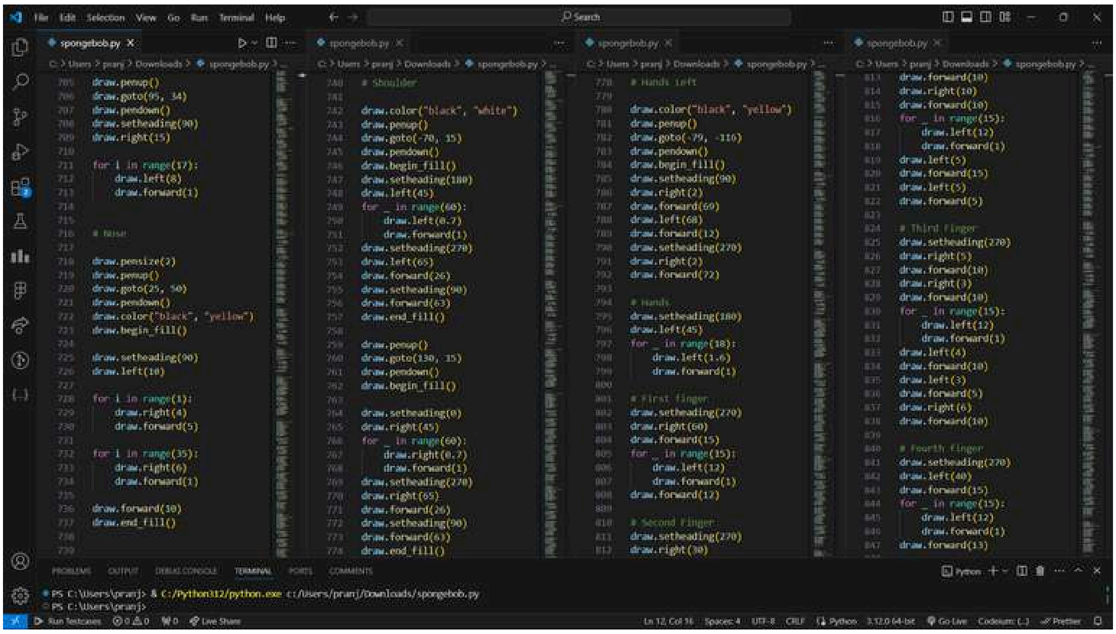


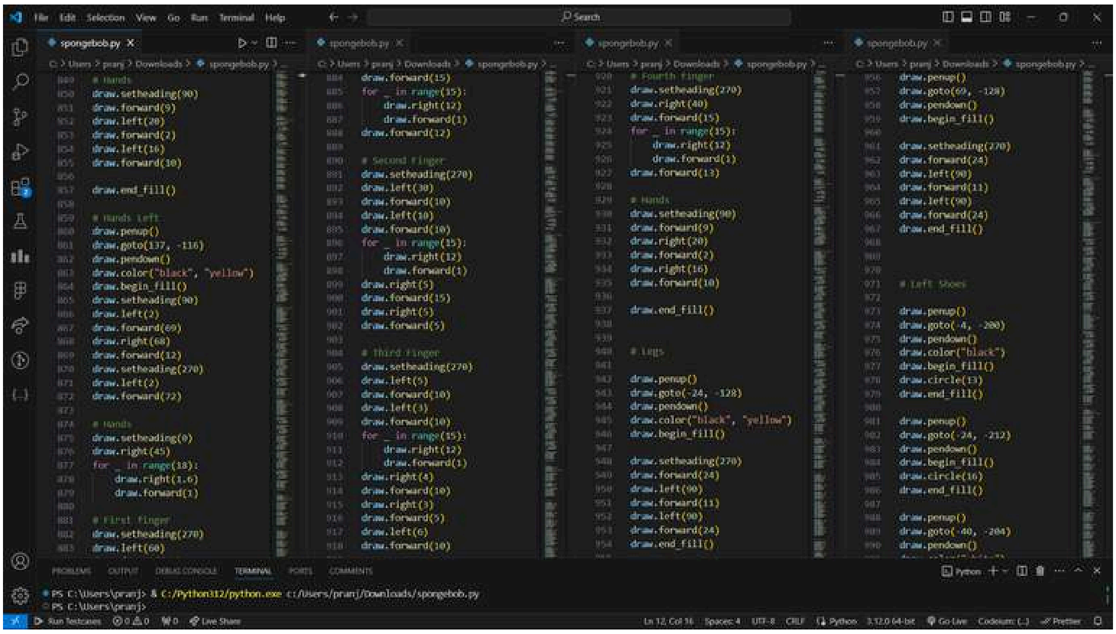


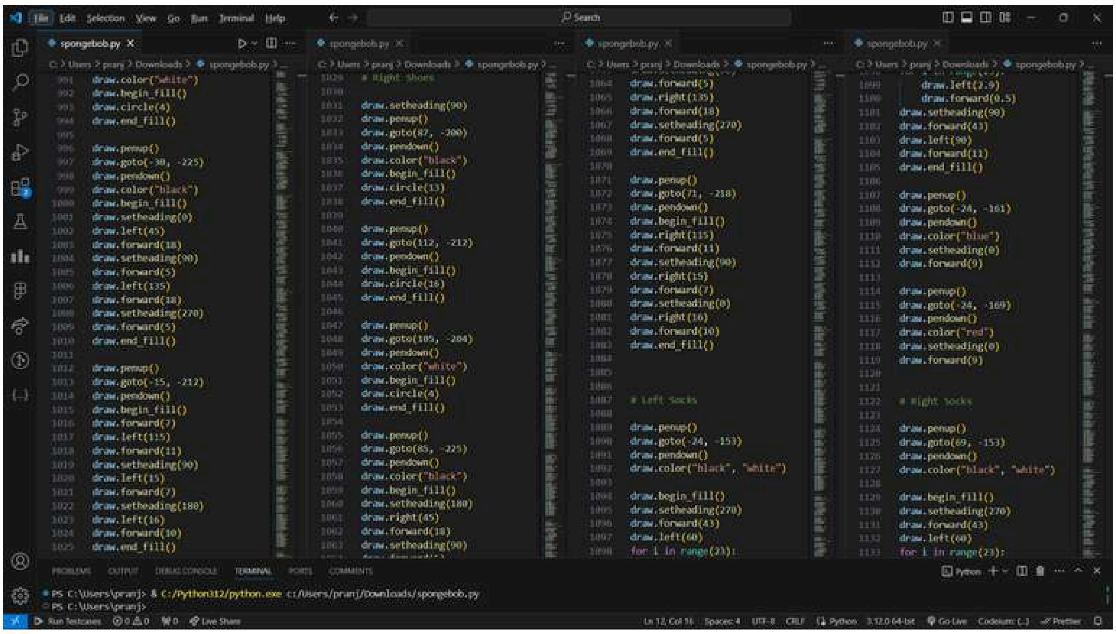


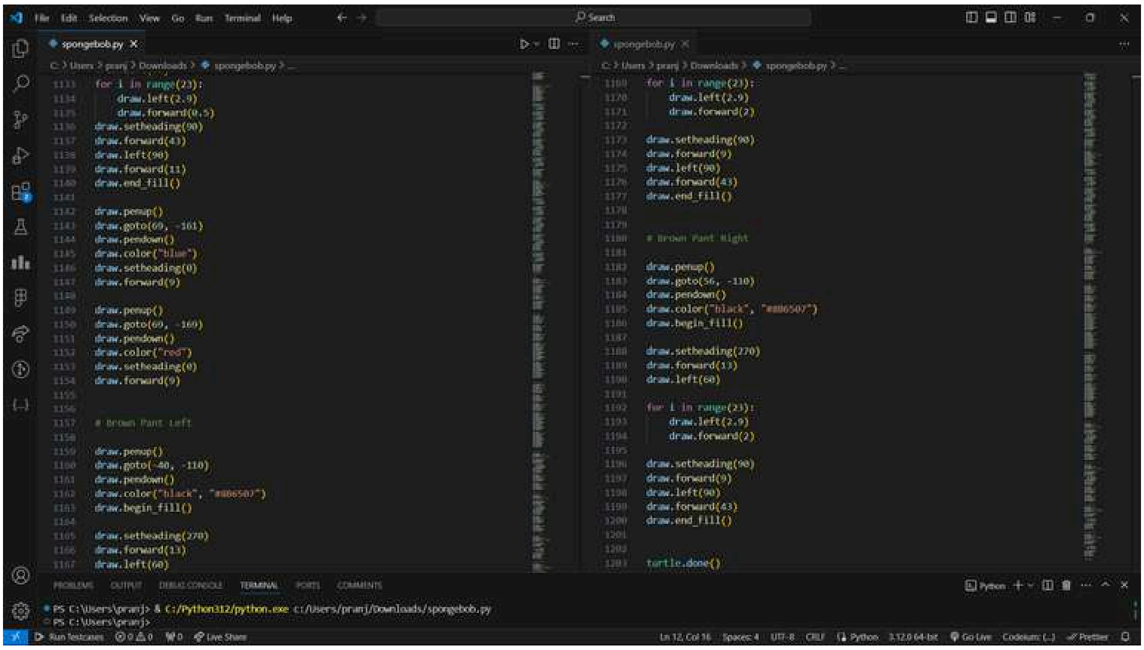










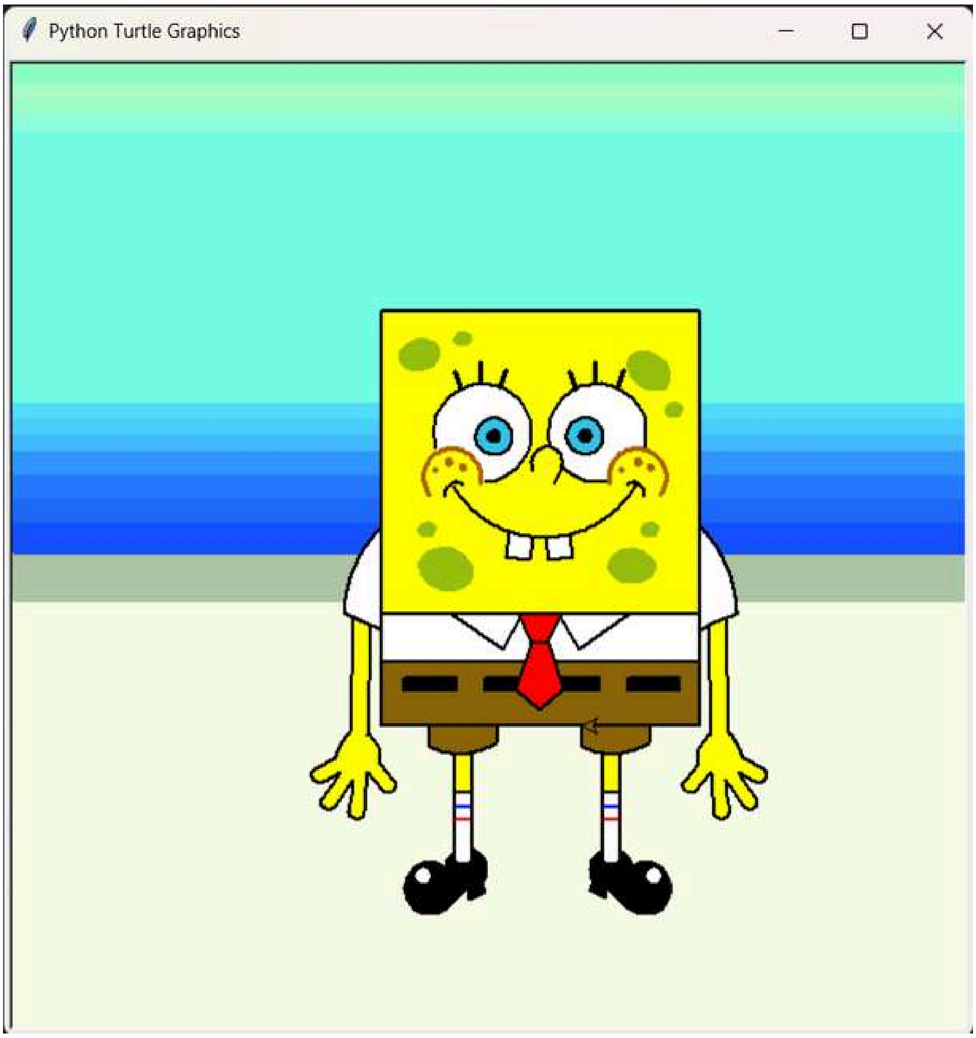


```python
for i in range(23):
    draw.left(2.9)
    draw.forward(0.5)
draw.setheading(90)
draw.forward(43)
draw.left(90)
draw.forward(11)
draw.end_fill()

draw.penup()
draw.goto(69, -161)
draw.pendown()
draw.color("blue")
draw.setheading(0)
draw.forward(9)

draw.penup()
draw.goto(69, -169)
draw.pendown()
draw.color("red")
draw.setheading(0)
draw.forward(9)


# Brown Pant Left

draw.penup()
draw.goto(-40, -110)
draw.pendown()
draw.color("black", "#8B6507")
draw.begin_fill()

draw.setheading(270)
draw.forward(13)
draw.left(60)
```

```python
for i in range(23):
    draw.left(2.9)
    draw.forward(2)

draw.setheading(90)
draw.forward(9)
draw.left(90)
draw.forward(43)
draw.end_fill()


# Brown Pant Right

draw.penup()
draw.goto(56, -110)
draw.pendown()
draw.color("black", "#8B6507")
draw.begin_fill()

draw.setheading(270)
draw.forward(13)
draw.left(60)

for i in range(23):
    draw.left(2.9)
    draw.forward(2)

draw.setheading(90)
draw.forward(9)
draw.left(90)
draw.forward(43)
draw.end_fill()


turtle.done()
```

```
PS C:\Users\pranj> & C:/Python312/python.exe c:/Users/pranj/Downloads/spongebob.py
PS C:\Users\pranj>
```

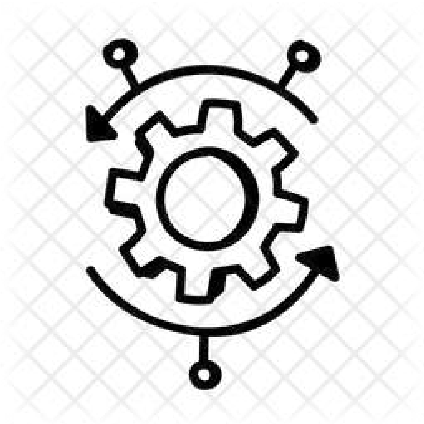
OUTPUT



This is the image of “Spongebob Squarepants” you get after running our code.

EXPLANANTION

The forward(x) function in Turtle draws a line of x pixels in the direction the turtle is facing. The concepts of Bresenham's line drawing and circle drawing algorithms are used in this project. It also accomodates Beizer curve algorithm, Scan line algorithm and Flood fill Algorithm.



1. BRESENHAM'S LINE DRAWING ALGORITHMS

This technique, pioneered by Bresenham, is utilized for scan converting a line.
It's notably efficient as it relies solely on integer addition, subtraction, and multiplication operations, which can be swiftly executed.
The algorithm selects the next pixel based on its proximity to the true line, ensuring a quick generation of lines.
Here's how it works:

- Starting from a pixel P1'(x1',y1'), subsequent pixels are chosen one at a time in the horizontal direction towards P2'(x2',y2').
- At each step, the next pixel is either the one to its right (lower-bound for the line) or the one diagonally up and to its right (upper-bound for the line).

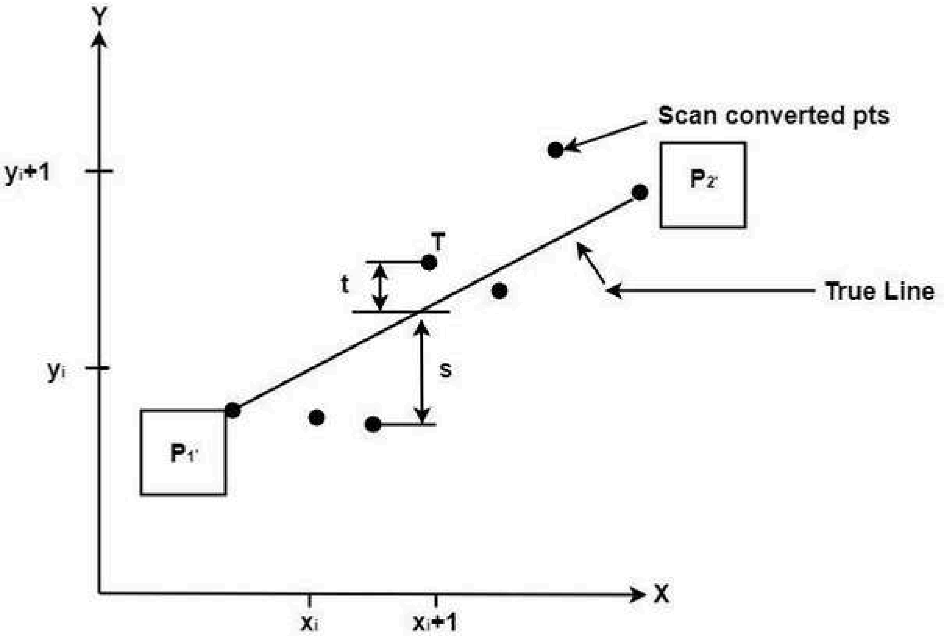
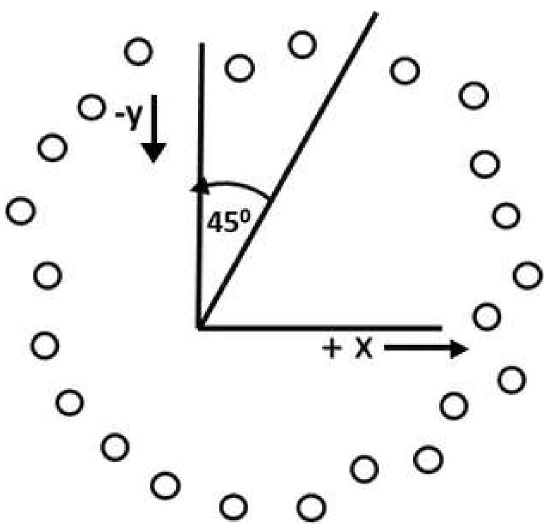


Fig: Scan Converting a line.

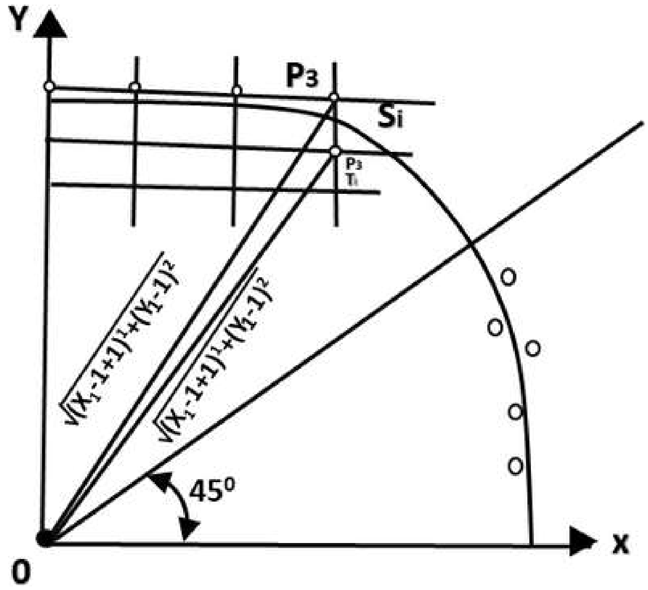
The line is accurately represented by pixels that minimize the distance from the path between P1' and P2'.

2.BRESENHAM’S CIRCLE DRAWING ALGORITHMS

Bresenham's algorithm for scan-converting a circle operates in the following manner: Points are generated from 90° to 45°, with movements restricted to the +x and -y directions, as depicted in the figure:-



The most accurate representation of the true circle is achieved by selecting pixels in the raster that are closest to the true circle. Our goal is to generate points from 90° to 45°. Let's assume that the last scan-converted pixel is labeled P1, as illustrated in the figure:-

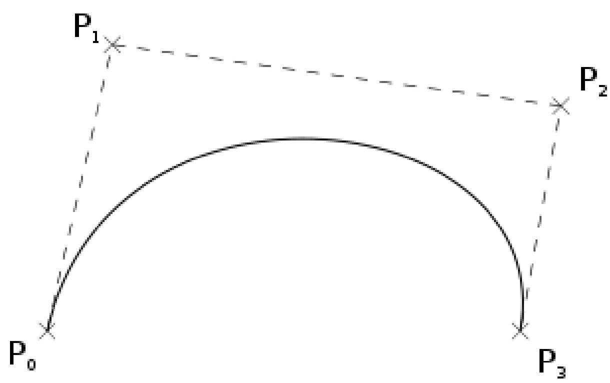


To identify each new point closest to the true circle, you have two options:

- Advance one unit in the x-direction.
- Advance one unit in the x-direction and simultaneously move one unit in the negative y-direction.

3.BEIZER CURVE ALGORITHM

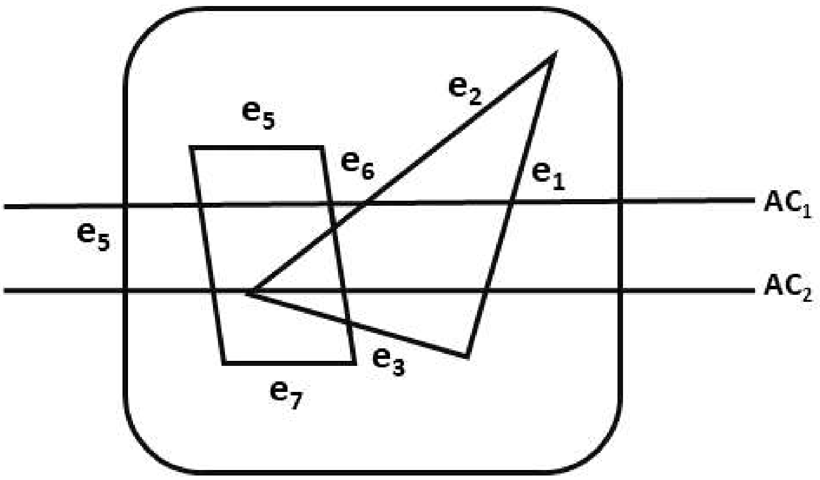
Bézier curves, a fundamental concept in computer graphics, find extensive application in various fields due to their versatility and smoothness. In the context of Turtle graphics, Bézier curves offer a powerful tool for creating complex and visually appealing shapes. Utilizing control points to define the curve's path, the algorithm efficiently interpolates between these points, providing local control over the curve's shape. This makes it ideal for generating smooth and precise curves in Turtle graphics, allowing for the creation of intricate designs and patterns with ease. Whether used for drawing intricate shapes or designing elaborate patterns, the Bézier curve algorithm, with its efficiency and accuracy, enhances the capabilities of Turtle graphics, enabling the creation of captivating visual compositions.



As shown in the diagram , the curve P0 to P3 is drawn by applying Beizer curve algorithm for points P0, P1, P2, P3.

4. SCAN LINE ALGORITHM

The Scan Line Algorithm operates on image space, processing one line at a time rather than pixel by pixel. It utilizes the concept of area coherence, managing edge and active edge lists for accurate rendering. The edge list records endpoint coordinates, while the Active Edge List (AEL) tracks edges intersecting a scan line, sorted by increasing x-values. This dynamic AEL grows and shrinks as needed during processing.
During each scan line iteration, the algorithm determines surface visibility based on intersecting edges, calculating depth for each surface. Visible surfaces receive intensity values stored in the refresh buffer.



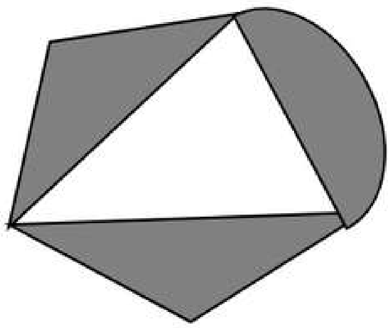
Following figures shown edges and active edge list. The active edge list for scan line AC1contain e1,e2,e5,e6 edges. The active edge list for scan line AC2contain e5,e6,e1.

Algorithm Steps:

1. Initialize data structures: Polygon table with color and edge pointers, Edge table with endpoint and polygon information, and sorted Active Edge List (AEL).
2. Begin scanning each line:
   - Populate the AEL sorted by y-values.
   - Scan until the flag F is active, using a background color.
   - Assign intensity values to visible surfaces based on depth.
   - Utilize coherence for remaining surfaces.
3. End the algorithm.

5.FLOOD FILL ALGORITHM

The flood fill algorithm begins by selecting a seed point inside the region to be filled. This point is then used as the starting point for filling the interior with a specified color. It employs either a 4-connected or 8-connected approach to traverse pixel positions, replacing existing colors with the desired color until the entire interior is filled. This method is particularly useful for filling regions with multiple boundary colors, providing a more efficient alternative to the boundary fill algorithm.



Algorithm:

```
Procedure floodfill (x, y,fill_ color, old_color: integer)
    If (getpixel (x, y)=old_color)
   {
    setpixel (x, y, fill_color);
    fill (x+1, y, fill_color, old_color);
     fill (x-1, y, fill_color, old_color);
    fill (x, y+1, fill_color, old_color);
    fill (x, y-1, fill_color, old_color);
     }
}
```

This algorithm is used for the flood fill algorithm to fill the colors in the figure.

CONCLUSION

Through the practical application of Computer Graphics principles in this project, we obtained a deeper understanding of how these concepts are utilized in real-world scenarios. Leveraging the Turtle library, a versatile Python tool, allowed us to create intricate and accurate drawings. By exploring various drawing techniques and algorithms, we were able to achieve detailed and precise visual representations. This hands-on experience provided valuable insights into the practical implementation of Computer Graphics concepts, enhancing our understanding of their significance in graphical applications.

REFERENCES

1. https://en.wikipedia.org/wiki/Bresenham's_line_algorithm
2. https://docs.python.org/3/library/turtle.html
3. https://www.geeksforgeeks.org/bresenhams-line- generation-algorithm/
4. https://www.geeksforgeeks.org/turtle-programming- python/