Md. Rafat Hossain Reyal
Roll No: 1910576122
Session:2018-19
Year: 4th year
Semester:Even Semester
Department:Computer Science
Engineering

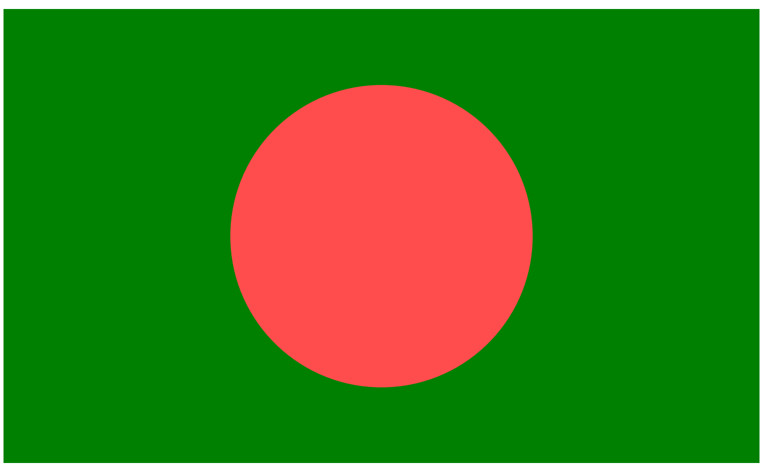
Contents

1 Draw National Flag

Python Code

```
import matplotlib.pyplot as plt

# Create figure and axis
fig, ax = plt.subplots()

# Draw green rectangle (flag background)
ax.add_patch(plt.Rectangle((0, 0), 10, 6, color='#006a4e'))
    # green

# Draw red circle (centered slightly to the left)
#ax.add_patch(plt.Circle((4.5, 3), 2, color='#f42a41'))  #
    red
ax.add_patch(plt.Circle((5, 3), 2, color='#f42a41'))

# Set limits and turn off axes
ax.set_xlim(0, 10)
ax.set_ylim(0, 6)
ax.set_aspect('equal')
plt.axis('off')

# Show flag
plt.show()
```

Flag Output



2 Your Name moving on screen Animation

Python Code

```python
import turtle
import time

t = turtle.Turtle()
t.hideturtle()
t.penup()
t.color("blue")

screen = turtle.Screen()
screen.bgcolor("white")
screen.tracer(0)

x = -300

while True:
    t.clear()
    t.goto(x, 0)
    t.write("Reyal", font=("Arial", 24, "bold"))
    x += 2
    if x > 300:
        x = -300
    screen.update()
    time.sleep(0.01)
```

Name animation Output:

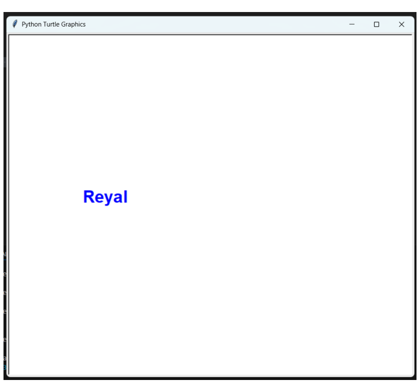


Figure 1: Initial Frame

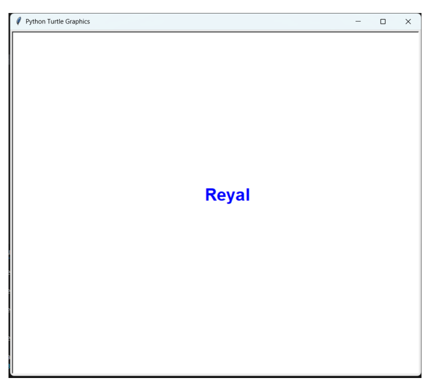


Figure 2: Moving Frame

3 Illustrate the Sutherland–Hodgman polygon clipping algorithm

Python Code

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# --- Clipping constants ---
LEFT, RIGHT, BOTTOM, TOP = 0, 1, 2, 3

# --- Clipping window ---
clip_window = (1.5, 3.5, 2.5, 4.5)  # xmin, xmax, ymin, ymax

# --- Star polygon points (closed) ---
star = [
    (2.5, 5.0), (2.8, 3.6), (4.2, 3.6),
    (3.0, 2.6), (3.5, 1.2), (2.5, 2.0),
    (1.5, 1.2), (2.0, 2.6), (0.8, 3.6),
    (2.2, 3.6), (2.5, 5.0)
]

# --- Inside test ---
def inside(p, edge, bounds):
    x, y = p
    xmin, xmax, ymin, ymax = bounds
    if edge == LEFT:
        return x >= xmin
    elif edge == RIGHT:
        return x <= xmax
    elif edge == BOTTOM:
        return y >= ymin
    elif edge == TOP:
        return y <= ymax

# --- Find intersection point ---
def intersection(p1, p2, edge, bounds):
    x1, y1 = p1
    x2, y2 = p2
    xmin, xmax, ymin, ymax = bounds

    if x1 == x2:
        m = float('inf')
    else:
        m = (y2 - y1) / (x2 - x1)

    if edge == LEFT:
        x = xmin
        y = y1 + m * (x - x1)
```

```python
    elif edge == RIGHT:
        x = xmax
        y = y1 + m * (x - x1)
    elif edge == BOTTOM:
        y = ymin
        x = x1 + (y - y1) / m if m != 0 else x1
    elif edge == TOP:
        y = ymax
        x = x1 + (y - y1) / m if m != 0 else x1

    return (x, y)

# --- Clip polygon against one edge ---
def clip_polygon(polygon, edge, bounds):
    output = []
    prev = polygon[-1]
    for curr in polygon:
        if inside(curr, edge, bounds):
            if inside(prev, edge, bounds):
                output.append(curr)
            else:
                output.append(intersection(prev, curr, edge, bounds))
                output.append(curr)
        elif inside(prev, edge, bounds):
            output.append(intersection(prev, curr, edge, bounds))
        prev = curr
    return output

# --- Perform all four edge clipping steps ---
clip_steps = [star]
for edge in [LEFT, RIGHT, BOTTOM, TOP]:
    clipped = clip_polygon(clip_steps[-1], edge, clip_window)
    clip_steps.append(clipped)

# --- Titles for each subplot ---
titles = ["Original Polygon", "Clip Left", "Clip Right", "Clip Bottom", "Clip Top"]

# --- Plotting (compact layout) ---
fig, axes = plt.subplots(1, 5, figsize=(14, 3), gridspec_kw={'wspace': 0.05})
xmin, xmax, ymin, ymax = clip_window

for i in range(5):
    ax = axes[i]
    polygon = clip_steps[i]
    if len(polygon) > 1:
```

```
        x, y = zip(*polygon)
        ax.fill(x, y, 'black', alpha=0.95)

    # Draw clipping window as dashed rectangle
    rect = patches.Rectangle((xmin, ymin), xmax - xmin, ymax - ymin,
                             linewidth=1, edgecolor='gray', facecolor='none', linestyle='--')
    ax.add_patch(rect)

    ax.set_xlim(0, 5)
    ax.set_ylim(0, 5.5)
    ax.set_aspect('equal')
    ax.set_title(titles[i], fontsize=9, pad=5)
    ax.axis('off')

plt.subplots_adjust(left=0.02, right=0.98, top=0.88, bottom=0.12)
plt.show()
```

Output:

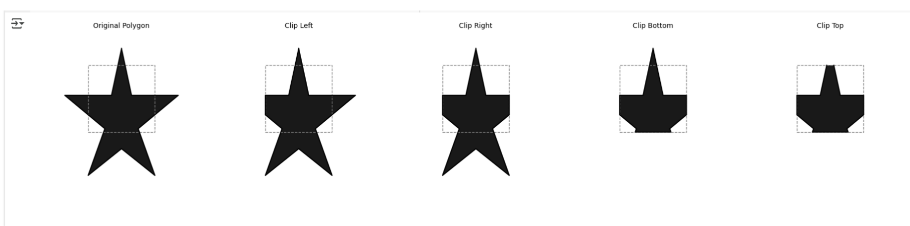


Figure 3: Stages of polygon clipping: Original Polygon, Clip Left, Clip Right, Clip Bottom, Clip Top.

4 Realistic Starfield Animation

Python Code

```
import turtle
import random
import math

# Setup screen
screen = turtle.Screen()
screen.bgcolor("black")
screen.title("Realistic Starfield View")
screen.setup(width=800, height=600)
screen.tracer(0)

# Star class for 3D movement
class Star:
    def __init__(self):
        self.reset()

    def reset(self):
        self.x = random.uniform(-400, 400)
        self.y = random.uniform(-300, 300)
        self.z = random.uniform(1, 800)
        self.pz = self.z
        self.color = random.choice(["white", "lightblue", "yellow", "lightgray"])

    def update(self, speed):
        self.pz = self.z
        self.z -= speed
        if self.z <= 1:
            self.reset()

    def draw(self, t):
        # Convert 3D to 2D perspective
        sx = int(self.x / self.z * 800)
        sy = int(self.y / self.z * 800)
        px = int(self.x / self.pz * 800)
        py = int(self.y / self.pz * 800)

        # Calculate star size
        size = max(1, int((800 - self.z) / 100))

        # Draw star trail
        t.pencolor(self.color)
        t.pensize(size / 2)
        t.goto(px, py)
        t.pendown()
```

```python
        t.goto(sx, sy)
        t.penup()

# Turtle setup
t = turtle.Turtle()
t.hideturtle()
t.penup()
t.speed(0)

# Create starfield
stars = [Star() for _ in range(150)]

# Animation loop
while True:
    t.clear()
    for star in stars:
        star.update(speed=10)
        star.draw(t)
    screen.update()
```

Output:

The following image depicts a realistic starfield view, simulating a cosmic scene with stars, galaxies, and nebulae in a deep space environment.

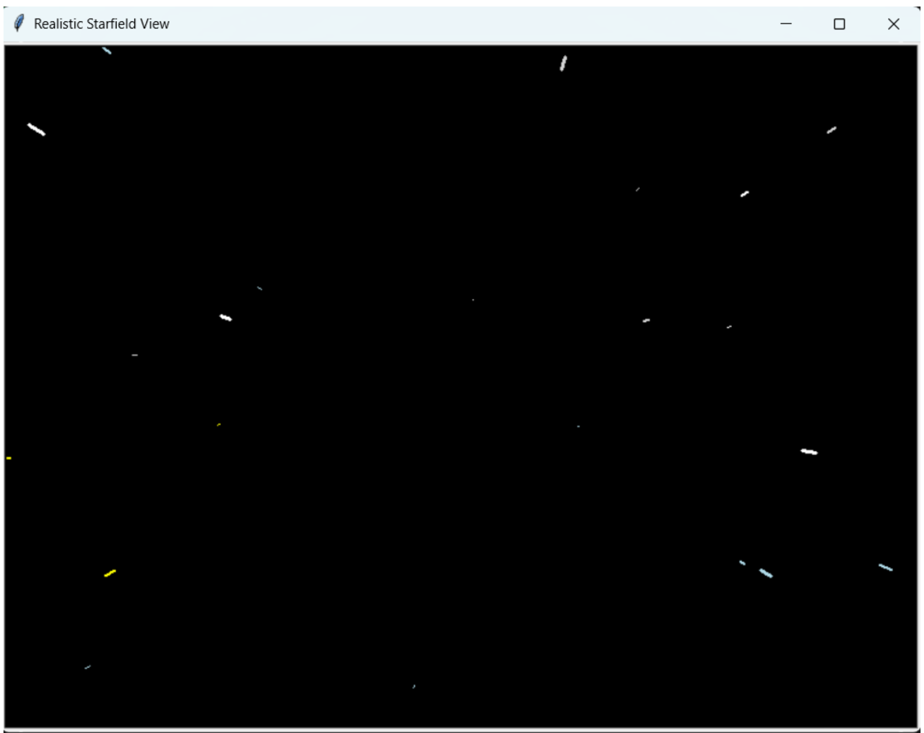


Figure 4: Realistic Starfield View: A simulation of the cosmos with stars, galaxies, and nebulae.