CIS 515: COMPUTER GRAPHICS

LAB – 2

UNIVERSITY OF MICHIGAN – DEARBORN

Fall 2024

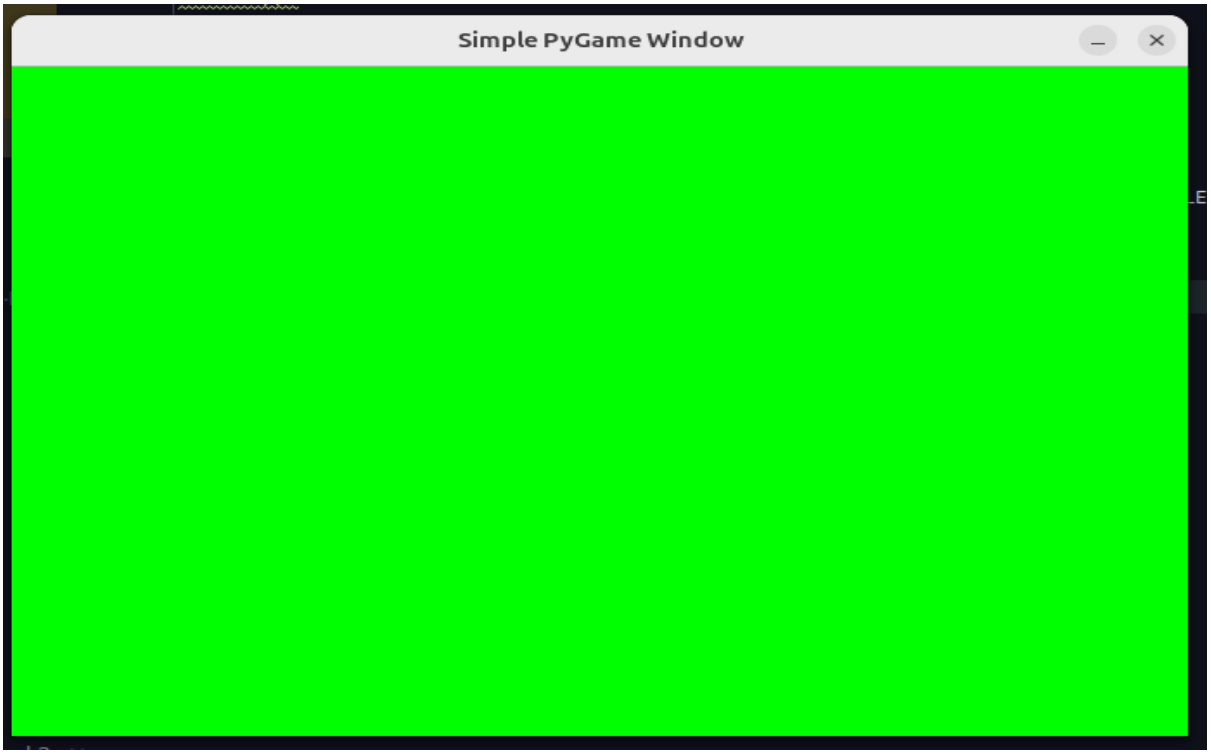
BY,

VISHVENDRA REDDY BHOOMIDI

bhoomidi@umich.edu

Question 1 – Simple PyGame Window:

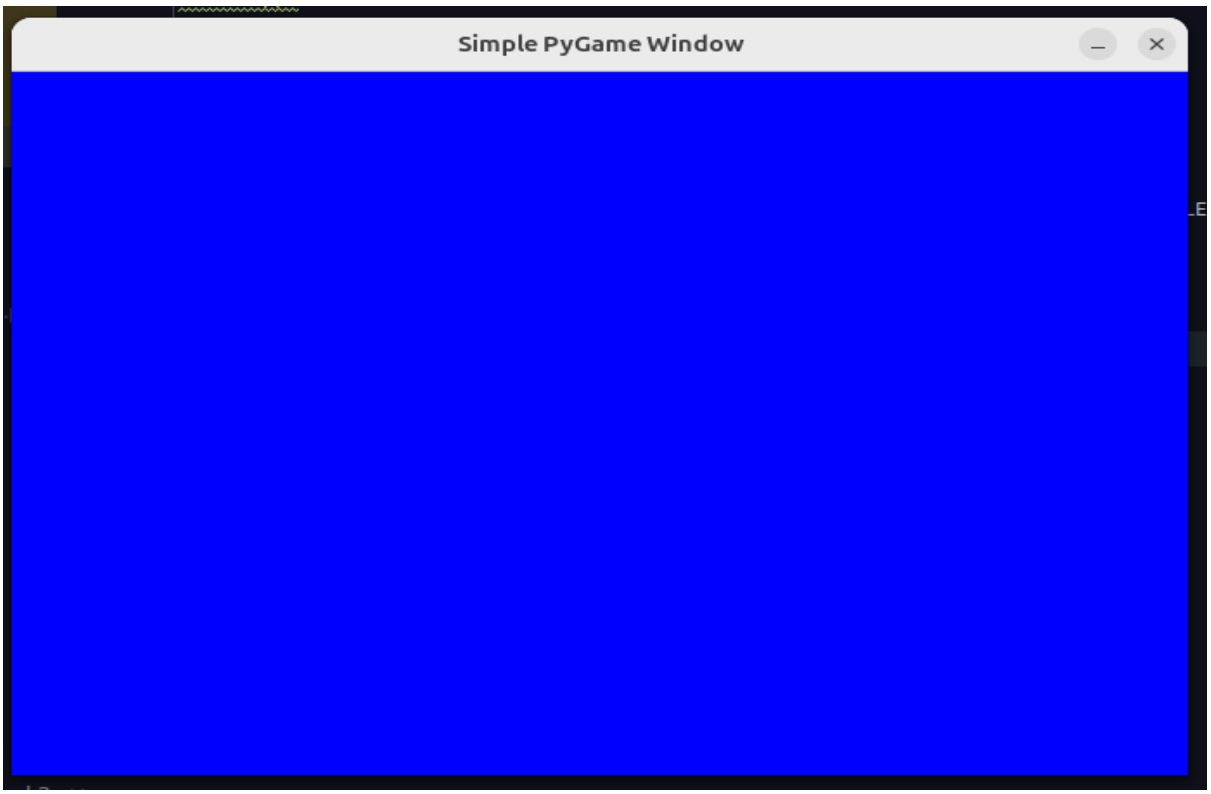
```
import pygame as pg
from OpenGL.GL import *


class App:
    def __init__(self):
        pg.init()

        pg.display.gl_set_attribute(pg.GL_CONTEXT_MAJOR_VERSION, 3)
        pg.display.gl_set_attribute(pg.GL_CONTEXT_MINOR_VERSION, 3)
        pg.display.gl_set_attribute(pg.GL_CONTEXT_PROFILE_MASK,
pg.GL_CONTEXT_PROFILE_CORE)

        pg.display.set_mode((680, 480), pg.OPENGL | pg.DOUBLEBUF)
        pg.display.set_caption('Simple PyGame Window')

        glClearColor(0.0, 1.0, 0.0, 1.0)
        self.clock = pg.time.Clock()

        self.mainloop()

    def mainloop(self):
        running = True
        while running:
            for event in pg.event.get():
                if event.type == pg.QUIT:
                    running = False

            glClear(GL_COLOR_BUFFER_BIT)
            pg.display.flip()

            self.clock.tick(60)

        pg.quit()


if __name__ == "__main__":
    App()
```



Question 2 – Modification of Background color:

```
import pygame as pg
from OpenGL.GL import *


class App:
    def __init__(self):
        pg.init()

        pg.display.gl_set_attribute(pg.GL_CONTEXT_MAJOR_VERSION, 3)
        pg.display.gl_set_attribute(pg.GL_CONTEXT_MINOR_VERSION, 3)
        pg.display.gl_set_attribute(pg.GL_CONTEXT_PROFILE_MASK,
pg.GL_CONTEXT_PROFILE_CORE)

        pg.display.set_mode((680, 480), pg.OPENGL | pg.DOUBLEBUF)
        pg.display.set_caption('Simple PyGame Window')

        glClearColor(0.0, 0.0, 1.0, 1.0)
        self.clock = pg.time.Clock()

        self.mainloop()

    def mainloop(self):
        running = True
        while running:
            for event in pg.event.get():
```

```
            if event.type == pg.QUIT:
                running = False

        glClear(GL_COLOR_BUFFER_BIT)
        pg.display.flip()

        self.clock.tick(60)

    pg.quit()


if __name__ == "__main__":
    App()
```

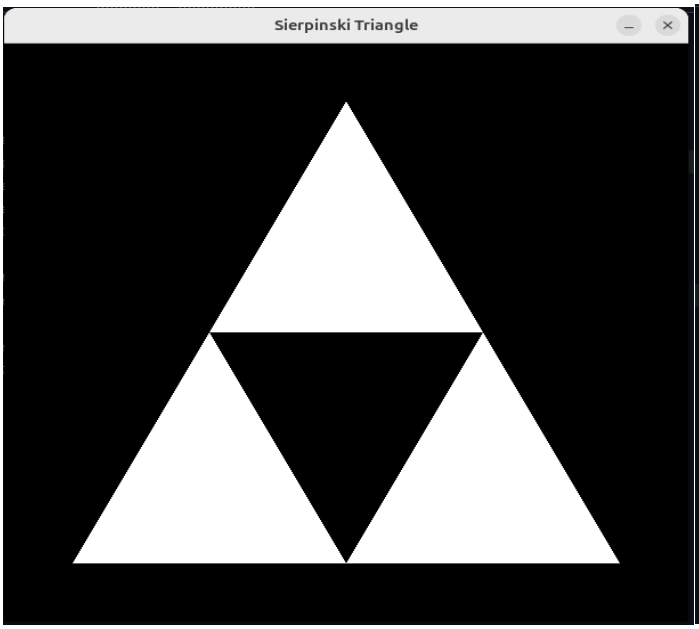


Question 3 – Sierpinski Triangle:

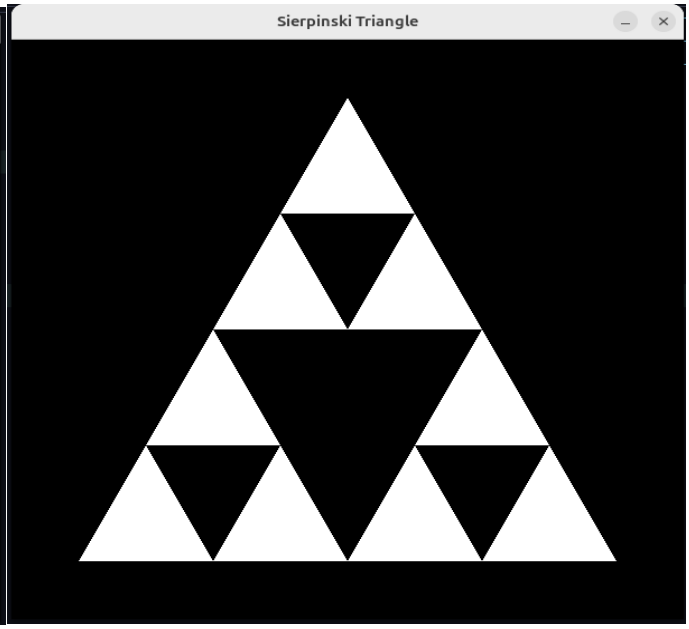
```
import pygame
from OpenGL.GL import *
from OpenGL.GLU import *

def midpoint(p1, p2):
    return [(p1[0] + p2[0]) / 2.0, (p1[1] + p2[1]) / 2.0]

def draw_sierpinski(vertices, iterations):
    if iterations == 0:
        glBegin(GL_TRIANGLES)
        glVertex2f(vertices[0][0], vertices[0][1])
        glVertex2f(vertices[1][0], vertices[1][1])
        glVertex2f(vertices[2][0], vertices[2][1])
        glEnd()
    else:
        midpoints = [
            midpoint(vertices[0], vertices[1]),
            midpoint(vertices[1], vertices[2]),
            midpoint(vertices[2], vertices[0])
        ]
        draw_sierpinski([vertices[0], midpoints[0], midpoints[2]], iterations - 1)
        draw_sierpinski([midpoints[0], vertices[1], midpoints[1]], iterations - 1)
        draw_sierpinski([midpoints[2], midpoints[1], vertices[2]], iterations - 1)

def init_window():
    pygame.init()
    display = (600, 600)
    pygame.display.set_mode(display, pygame.DOUBLEBUF | pygame.OPENGL)
    pygame.display.set_caption('Sierpinski Triangle')
    glClearColor(1, 1, 1, 1)
    gluOrtho2D(-1, 1, -1, 1)

def main():
    init_window()
    vertices = [[-0.8, -0.8], [0.8, -0.8], [0.0, 0.8]]
    iterations = 4

    running = True
    while running:
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glColor3f(0, 0, 0)
        draw_sierpinski(vertices, iterations)

        pygame.display.flip()
```

```
        pygame.time.wait(10)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

    pygame.quit()

if __name__ == "__main__":
    main()
```

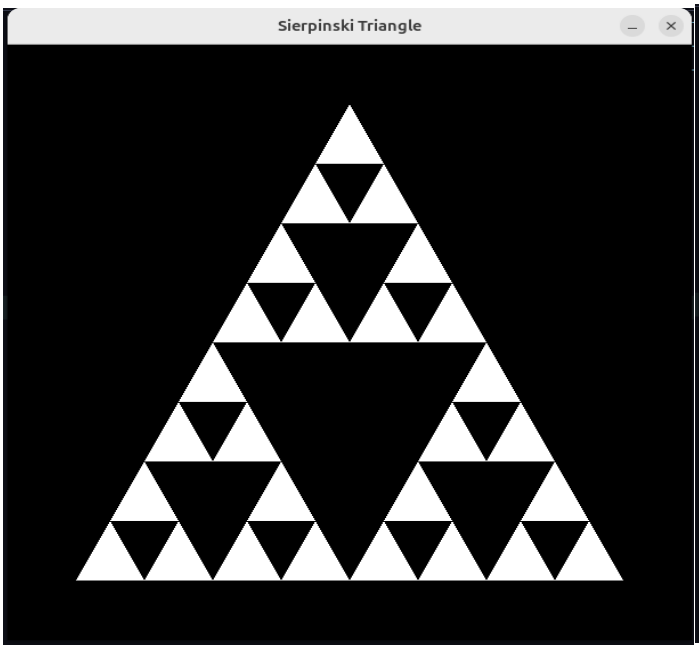
Iteration = 1



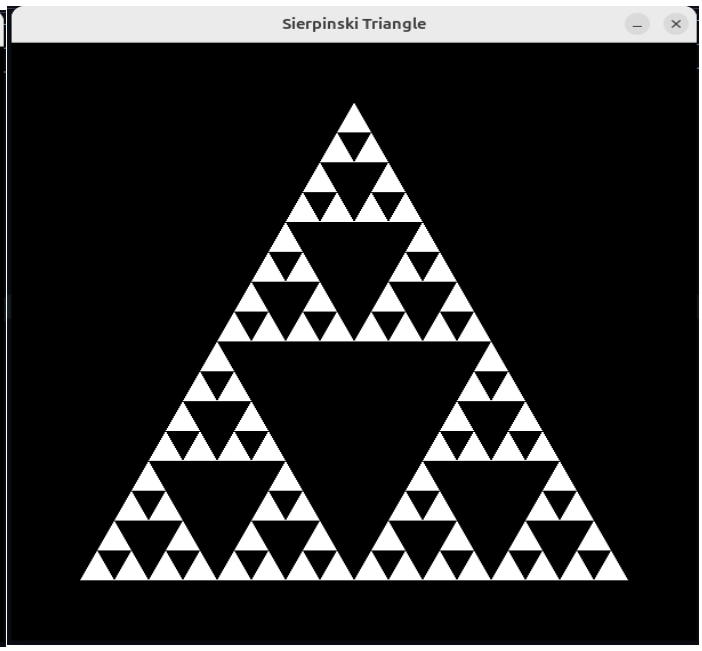
Iteration = 2



Iteration = 3



Iteration = 4

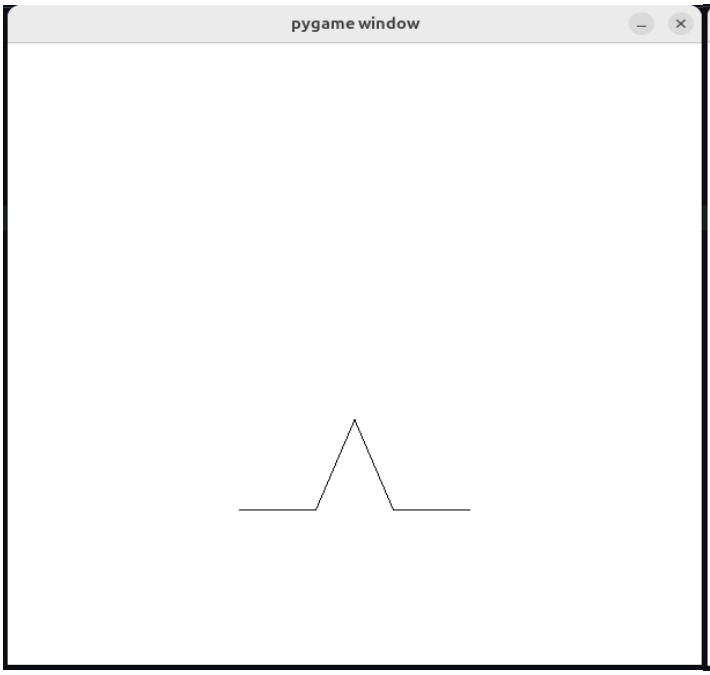


Question 4 – Koch Snowflake:

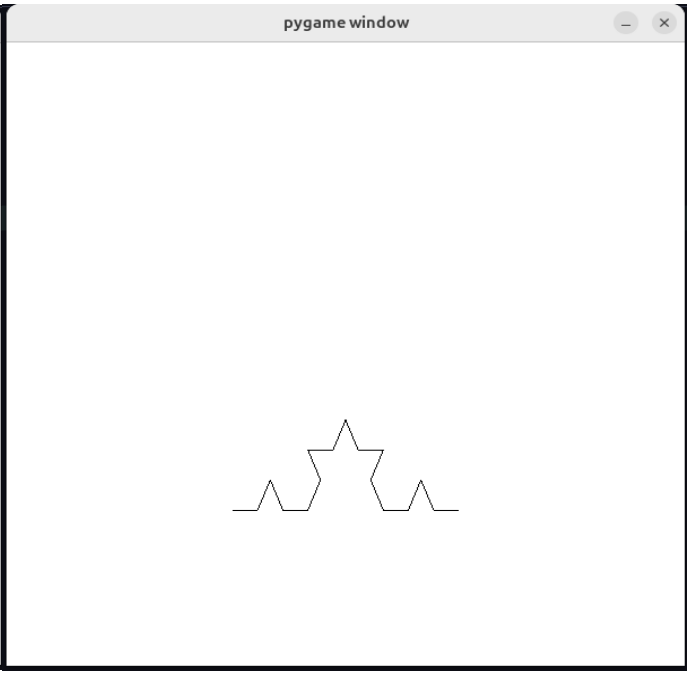
```
import pygame
from OpenGL.GL import *
from OpenGL.GLU import *
from math import *


def draw_koch_curve(p1, p2, iterations):
    if iterations == 0:
        glBegin(GL_LINES)
        glVertex2f(p1[0], p1[1])
        glVertex2f(p2[0], p2[1])
        glEnd()
    else:
        one_third = [(2 * p1[0] + p2[0]) / 3, (2 * p1[1] + p2[1]) / 3]
        two_third = [(p1[0] + 2 * p2[0]) / 3, (p1[1] + 2 * p2[1]) / 3]

        dx = p2[0] - p1[0]
        dy = p2[1] - p1[1]
        length = sqrt(dx ** 2 + dy ** 2) / 3
        angle = atan2(dy, dx) + pi / 3
        peak = [one_third[0] + length * cos(angle), one_third[1] + length * sin(angle)]

        draw_koch_curve(p1, one_third, iterations - 1)
        draw_koch_curve(one_third, peak, iterations - 1)
        draw_koch_curve(peak, two_third, iterations - 1)
        draw_koch_curve(two_third, p2, iterations - 1)


def init_window():
    pygame.init()
    display = (600, 600)
    pygame.display.set_mode(display, pygame.DOUBLEBUF | pygame.OPENGL)
    glClearColor(1, 1, 1, 1)
    gluOrtho2D(-1.5, 1.5, -1, 1)


def main():
    init_window()

    iterations = 4
    vertices = [
        [-0.5, -0.5],
        [0.5, -0.5]
    ]
```

```
    running = True
    while running:
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glColor3f(0, 0, 0)

        draw_koch_curve(vertices[0], vertices[1], iterations)

        pygame.display.flip()
        pygame.time.wait(10)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

    pygame.quit()


if __name__ == "__main__":
    main()
```

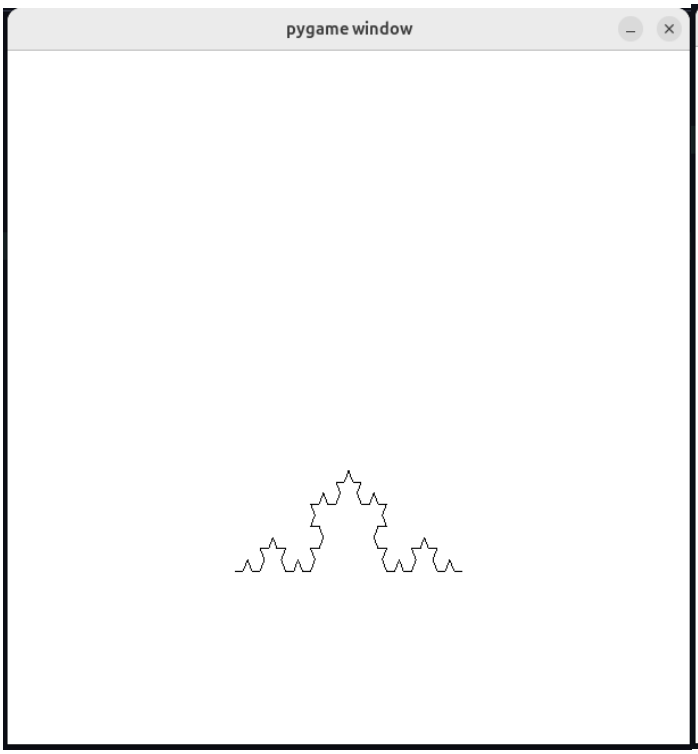
Iteration = 1



Iteration = 2



Iteration = 3



Iteration = 4