Universidade Federal de Ouro Preto - UFOP
Departamento de Ciência da Computação - DECOM

Relatório atividade 5

BCC327 - COMPUTACAO GRAFICA

Kayo Xavier Nascimento Cavalcante Leite - 21.2.4095

Professor: Rafael Alves Bonfim

Ouro Preto
30 de março de 2025

Sumário

Lista de Figuras

Lista de Códigos Fonte

1 Código e enuciado.

Na primeira atividade, foi requerido: Criação de um triângulo colorido com gradiente. Esse exercício visa à criação de um triângulo 2D com um gradiente de cores interpolado entre seus vértices. Além disto, foi pedido a implementação da movimentação de um quadrado 2D com o teclado. Este exercício, tendo como objetivo criar uma aplicação onde o usuário pode mover um quadrado na tela usando as teclas do teclado.

Para tal, decidi utilizar as bibliotecas pygame e pyopengl para gerar uma aplicação que satisfaça a demanda do gradiente. Como o segundo exercício já foi implementado de maneira semelhante na atividade dois, apenas reutilizei o código do mesmo para demonstração

O código da atividade se encontra anexado junto ao pdf com o relatório e, para executá-lo,o, necessário instalar o python e as bibliotecas Pygame e pyopengl, para tal pode-se utilizar o seguinte comando :

Instalando bibliotecas

no linux:
sudo pip3 install pygame
sudo pip3 install pyopengl
para o windows:
pip install pyopengl
pip install pygame

.

Após a instalação das bibliotecas, para rodar os códigos basta utilizar o terminal para entrar no diretório da atividade e usar o seguinte comando:

Rodando o exercício

python trianglegradient.py
python MovimentacaoQuadrado.py

O código se encontra todo comentado e formatado. A seguir a visualização do código completo da implementação do gradiente:

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

# Inicializa  o do Pygame
pygame.init()

# Configura  es da tela
WIDTH, HEIGHT = 800, 600
pygame.display.set_mode((WIDTH, HEIGHT), pygame.OPENGL | pygame.DOUBLEBUF)
pygame.display.set_caption("Tri ngulo Colorido com Gradiente")

def init_gl():
    # Define a cor de fundo
    glClearColor(1.0, 1.0, 1.0, 1.0)  # Fundo branco
    # Define o viewport
    glViewport(0, 0, WIDTH, HEIGHT)
    # Configura a proje  o para 2D com uma proje  o ortogr fica
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(0, WIDTH, 0, HEIGHT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    print("OpenGL inicializado com proje  o ortogr fica para 2D.")

```

```
def draw_gradient_triangle():
    glBegin(GL_TRIANGLES)
    # V rtice 1: Vermelho
    glColor3f(1.0, 0.0, 0.0)
    glVertex2f(WIDTH / 2, HEIGHT * 0.8)
    # V rtice 2: Verde
    glColor3f(0.0, 1.0, 0.0)
    glVertex2f(WIDTH * 0.2, HEIGHT * 0.2)
    # V rtice 3: Azul
    glColor3f(0.0, 0.0, 1.0)
    glVertex2f(WIDTH * 0.8, HEIGHT * 0.2)
    glEnd()

def main():
    init_gl()
    running = True

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

        glClear(GL_COLOR_BUFFER_BIT)
        draw_gradient_triangle()
        pygame.display.flip()


    pygame.quit()
    print("Programa finalizado.")

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

Código 1: Implementação do gradiente.

agora a implementação da movimentação do quadrado:

```
import pygame
import sys

# Kayo Xavier Nascimento Cavalcante Leite - 21.2.4095

# Inicializa  o do Pygame
pygame.init()

# Defini  es da tela
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Jogo do quadrado")

# Defini  es de cores
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
BLUE = (0, 0, 255)

#Defini  es do quadrado
object_x, object_y = 400, 300  # Initial position
object_size = 50
speed = 5

# Main game loop
running = True
```

```
while running:

    # Limpa a tela
    screen.fill(WHITE)

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

     # Movimenta  o da bolinha
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        object_x -= speed
    if keys[pygame.K_RIGHT]:
        object_x += speed
    if keys[pygame.K_UP]:
        object_y -= speed
    if keys[pygame.K_DOWN]:
        object_y += speed

    # Verifica colis es com as bordas
    object_x = max(0, min(object_x, WIDTH - object_size))
    object_y = max(0, min(object_y, HEIGHT - object_size))

    # Desenha quadrado:
    pygame.draw.rect(screen, BLACK, (object_x, object_y, object_size,
        object_size))

    # Atualiza o display
    pygame.display.flip()
    pygame.time.Clock().tick(60)  # Cap at 60 FPS

pygame.quit()
sys.exit()
```

Código 2: Implementação do MovimentacaoQuadrado.py

A seguir, a visualização da tela da aplicação com o gradiente e da movimentação do quadrado

Figura 1: Visualização do gradiente.

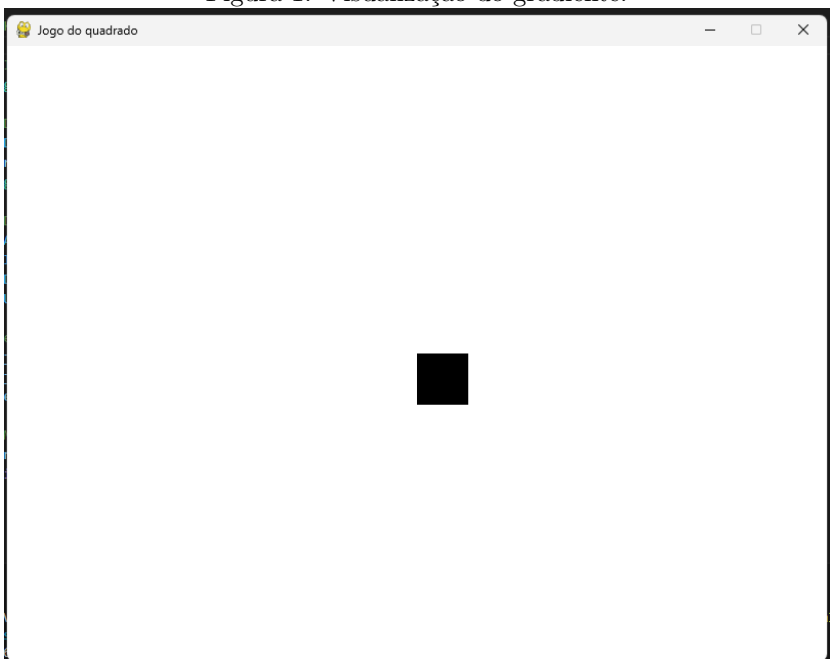

Figura 2: Visualização da movimentação do quadrado.