Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Mini Project Report on:

"MY Paint"

[Code No: COMP 342]

(For partial fulfillment of 3rd Year/1st Semester in Computer Science)

Submitted By

Mukul Aryal (03)
Yunidh Rawal (48)

Submitted To:

Dhiraj Shrestha

Department of Computer Science and Engineering

Submission Date:2024/06/15

1: Introduction

This program is a simple 2D drawing application implemented using:

- Python
- GLFW (an OpenGL library for window management and input handling)
- OpenGL (for rendering graphics)

It was inspired by the classic windows application MS Paint, and we have aimed to create a simple application to experiment how OpenGL and graphics rendering can be used to draw shapes in real-time.

2: Project Description

Source Code:

```
import glfw
from OpenGL.GL import *
import numpy as np
import imageio

# Initialize window height
WIDTH = 1600
HEIGHT = 900

# Constants for brush colors and sizes
BRUSH_COLORS = {
    "black": (0.0, 0.0, 0.0),
    "red": (1.0, 0.0, 0.0),
    "green": (0.0, 1.0, 0.0),
    "blue": (0.0, 0.0, 1.0),
    "eraser": (1.0, 1.0, 1.0),
}

BRUSH_SIZES = {
    "small": 5,
    "medium": 10,
    "large": 15,
}

# Initial brush color
current_color = BRUSH_COLORS["black"]
current_size = BRUSH_SIZES["small"]

# Store drawn points and their colors
drawn_points = []
```

```
# Callback functions
def key_callback(window, key, scancode, action, mods):
    global current_color, current_size
    if action == glfw.PRESS:
        if key == glfw.KEY_B:
            current_color = BRUSH_COLORS["black"]
        elif key == glfw.KEY_R:
            current_color = BRUSH_COLORS["red"]
        elif key == glfw.KEY_G:
            current_color = BRUSH_COLORS["green"]
        elif key == glfw.KEY_U:
            current_color = BRUSH_COLORS["blue"]
        elif key == glfw.KEY_E:
            current_color = BRUSH_COLORS["eraser"]
        elif key == glfw.KEY_S:
            save_image()
        elif key == glfw.KEY_1:
            current_size = BRUSH_SIZES["small"]
        elif key == glfw.KEY_2:
            current_size = BRUSH_SIZES["medium"]
        elif key == glfw.KEY_3:
            current_size = BRUSH_SIZES["large"]

def mouse_button_callback(window, button, action, mods):
    global drawing
    if button == glfw.MOUSE_BUTTON_LEFT:
        if action == glfw.PRESS:
            drawing = True
        elif action == glfw.RELEASE:
            drawing = False
# Save the drawing as an image
def save_image(filename="drawing.png"):
    widht, height = glfw.get_framebuffer_size(window)
    glReadBuffer(GL_FRONT)
    pixels = glReadPixels(0, 0, widht, height, GL_RGB, GL_UNSIGNED_BYTE)
    image = np.frombuffer(pixels, dtype=np.uint8).reshape(height, widht, 3)
    image = np.flipud(image)  # Flip the image vertically
    imageio.imwrite(filename, image)
```

```
# Initialization steps for GLFW window
if not glfw.init():
    raise Exception("GLFW initialization failed")
window = glfw.create_window(WIDTH, HEIGHT, "MY Paint Program", None, None)
if not window:
    glfw.terminate()
    raise Exception("GLFW window creation failed")
glfw.make_context_current(window)
glfw.set_key_callback(window, key_callback)
glfw.set_mouse_button_callback(window, mouse_button_callback)

# OpenGL settings
glOrtho(0, WIDTH, 0, HEIGHT, -1, 1)
glClearColor(1.0, 1.0, 1.0, 1.0)
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
glfw.swap_buffers(window)

# Main loop:
drawing = False
while not glfw.window_should_close(window):
    glfw.wait_events()
    if drawing:
        x, y = glfw.get_cursor_pos(window)
        y = HEIGHT - y  # Convert to OpenGL coordinates
        drawn_points.append((x, y, current_color, current_size))
    # Draw the canvas
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    for point in drawn_points:
        glColor3f(*point[2])
        glPointSize(point[3])
        glBegin(GL_POINTS)
        glVertex2f(point[0], point[1])
        glEnd()
    glfw.swap_buffers(window)
glfw.terminate()
```

Here's a description of what the code is doing:

Imports and Initialization:

- The code starts by importing the necessary libraries: `glfw` for creating and managing the window, `OpenGL.GL` for rendering graphics, `numpy` for array operations, `imageio` for saving the drawing as an image.
- It initializes GLFW and checks if the initialization was successful.
- It sets the window dimensions (`WIDTH` and `HEIGHT`) to 1600x900 pixels.
- It defines dictionaries for brush colors (`BRUSH_COLORS`) and brush sizes (`BRUSH_SIZES`).
- It initializes the current brush color and size to black and small, respectively.
- It creates an empty list (`drawn_points`) to store the coordinates, colors, and sizes of the drawn points.

Callback Functions:

- `key_callback` is a function that handles keyboard input. It changes the current brush color or size based on the key pressed.

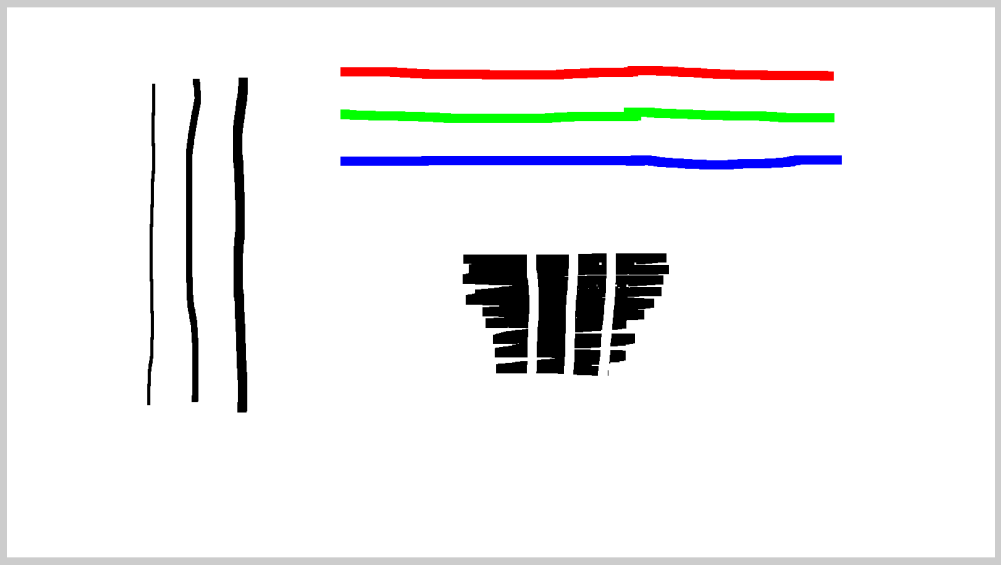
  List of commands:

  'B' : Black brush
  'R' : Red brush
  'G' : Green brush
  'U' : Blue brush
  'E' : Eraser
  'S' : Save image
  '1' : Small brush size
  '2' : Medium brush size
  '3' : Large brush size

- `mouse_button_callback` is a function that handles mouse input. It sets the `drawing` flag to `True` when the left mouse button is pressed and `False` when it's released.

Image Saving Function:

- `save_image` is a function that saves the current drawing as an image file. It reads the pixel data from the front buffer of the window, converts it to a NumPy array, flips it vertically, and saves it as a PNG file using `imageio.imwrite`.



Example of different brushes, picture taken with the application's saving feature.

GLFW Window Initialization:

- The code creates a GLFW window with the specified dimensions and title.
- It sets the current OpenGL context for the window.
- It registers the `key_callback` and `mouse_button_callback` functions to handle keyboard and mouse input, respectively.
- It sets up the OpenGL viewport and clears color .

Main Loop:

- The `drawing` flag is initialized to `False`.
- The main loop runs until the user closes the window.
- Inside the loop:
    - It waits for events (e.g., keyboard, mouse) using `glfw.wait_events()`.
    - If the `drawing` flag is `True`, it gets the current cursor position, converts it to OpenGL coordinates, and appends the coordinates, current color, and current size to the `drawn_points` list.
    - It clears the color and depth buffers.
    - It iterates over the `drawn_points` list and draws each point as a single OpenGL point primitive with the specified color and size.
    - It swaps the front and back buffers to display the rendered frame.

Termination:

- After the main loop ends (when the user closes the window), the code terminates GLFW.

3: Conclusion

In conclusion, this program demonstrates a basic 2D drawing application inspired by MS Paint, implemented using Python with GLFW and OpenGL for real-time graphics rendering. The application allows users to draw with various brush colors and sizes, and also allows them to save the drawings as an image file. Through this project, we explored the integration of OpenGL for rendering and GLFW for input handling, resulting in a simple yet functional drawing tool. This project highlights the potential of combining these technologies to create interactive graphical applications.

4: References

- **GLFW Documentation**: https://www.glfw.org/docs/latest/
- **OpenGL Documentation**: OpenGL 4.6 (Core Profile) - May 5, 2022