Assignment 1: Pencils

This document contains guidelines and instructions for both **Lab 1** and **Assignment 1**. Also, check out the **Grading** and **Submission Guideline** **at the bottom of this document.**

Introduction

In this assignment, you will take your first steps into computer graphics by understanding how digital images are represented and modified and implementing a "pencil" tool to draw on a canvas using a mouse interactively.

Objectives

1. Learn about how colors and digital images are represented digitally,
2. Create pixel art by programmatically modifying an image and
3. Implement a Photoshop-like pencil to draw on a canvas interactively.

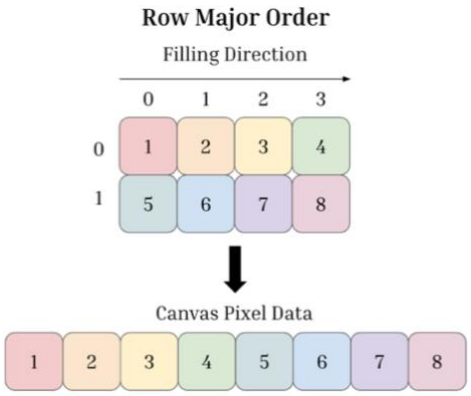
Code Structure and Program Running

There are two Python files:

- **assignment1_canvas.py**: canvas class and functions, where we modify pixel data on a given canvas.
- **assignment1_main.py**: main function, where we create the program window, mouse events to enable interactions with canvas data, and OpenGL drawing functions to render the pixel data on canvas.

Lab 1: Your First Canvas

In this course, we store pixel data in **row-major order**. Specifically, we use a 1D data structure to store 2D image data by "flattening" the image data in row-major order.



There are three types of canvas:

- **Grayscale**: drawing pixels in grayscale, i.e., black, white, or gray of different shades
- **Color**: drawing a color image with specific RGB colors for each pixel
- **RandomColor (default)**: drawing pixels of random colors

By running the *main* Python file (**assignment1_main.py**), you should see **Figure 1**:

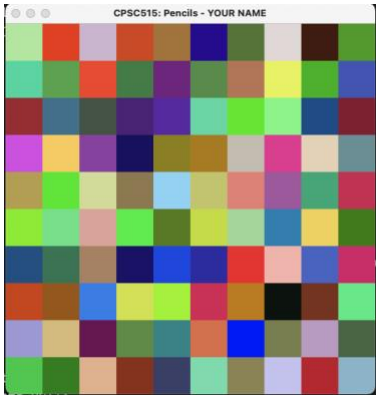


Figure 1: Draw canvas with *randomColor*.

Task 1: Create A Grayscale Canvas

In "A1_pencil_canvas.py":

Task 1.1: Implement the function `*floatToInt()*` to convert the input intensity in float into an integer (`*grayscale*`) within the range [0, 255].

Task 1.2: Implement the function `*initGrayCanvas*()`. This function should make self.data a 10 (Width) x 10 (Height) canvas and fill it with a dark gray of intensity 0.123. Note that `self.data` is a 1D NumPy array of RGBA values.

Hint: you will call the function `*floatToInt*()` to convert intensity into an integer of [0, 255]. **When drawing a grayscale image, each pixel's color channel shares the same value**, and the alpha value (A) is always 255 (opaque).

In "A1_pencil_main.py":

Task 1.3: Change the attribute `*canvas_type*` into "*grayscale*" when initializing the canvas at the beginning of the *main*() function. By running the *main* Python file, you should see **Figure 2**:

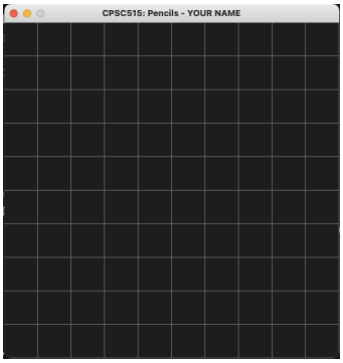


Figure 2: Task 1's result.

Task 1.4: Adjust the intensity value in the function `*initGrayCanvas*() to see the difference in the grayscale image. Verify your code in `*floatToInt*()` to see if higher intensity creates a brighter image and vice versa.

Task 2: Create A Heart

In this course, we choose the origin of the canvas' coordinate system at the **bottom-left.** The x-axis points to the right, and the y-axis points upward.

In "A1_pencil_canvas.py":

Task 2.1: Complete the function `*posToIndex*()` to convert a pixel's (row, column) coordinates in an image into its index in the canvas data (i.e., a 1D array) in row-majored order. To verify your code, see if the index of the pixel at (1, 2) is 12.

Task 2.2: Complete the function `*createHeart*()` to set the intensity of the corresponding pixels in Figure 3 to be 1.0. Use `*floatToInt*()` to convert intensity into an integer in the range [0, 255]. You must create a list of the pixels with (row, column) coordinates to represent the white pixels of the heart, then use `*posToIndex*()` to convert each (row, column) into an index of the canvas data to be updated.

Task 2.3: In the `*initGrayCanvas*()`, call your function `*createHeart()*`. Still, use the "grayscale" attribute for the "*canvas_type*" when initializing the canvas in the main function.

By running the *main* Python file, you should see **Figure 3**:

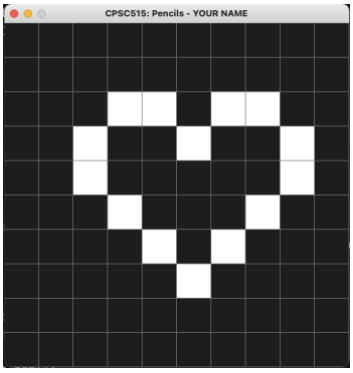


Figure 3: Task 2's result.

Task 3: Interacting with the Mouse

In "A1_pencil_main.py":

Task 3.1: Understand PyGame's mouse events and convert mouse coordinates into canvas coordinates. We have provided a stencil code for detecting and handling the mouse's left-button pressing and releasing events using events such as *'pygame.MOUSEBUTTONDOWN`* and `*event.button==1*`.

PyGame display's coordinate system is different from our canvas'. In PyGame, the origin of the display is at the **top-left corner**, and the x-axis points to the right, and the y-axis points downward. Also, the current PyGame's window's resolution is 500 x 500, which is also different from our canvas' 10 x 10. Click on the screen at random places, and read the mouse's (x, y) coordinates in the terminal window of the VS Code.

Write code to convert the mouse's (x, y) coordinates in the PyGame's display coordinate system into the canvas' (x, y) coordinates.

Task 3.2: Write code to "turn on" the pixels clicked by the mouse's left button by setting the pixel intensity to become white. By running the *main* Python file, you should see **Figure 4**:

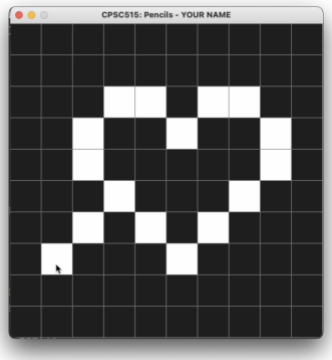


Figure 4: Task 3's result: **left-click** to "turn on" a pixel.

Supplement Materials:

- PyGame Event Handling: https://www.geeksforgeeks.org/pygame-event-handling/

Assignment 1: Pencils

Task 4: Create A Colorful Canvas

In "A1_pencil_canvas.py":

Task 4.1: Implement `*initializeColorCanvas*()` function. This function should make self.data a 10 x 10 canvas and fill it with RGBA color (0, 123, 123, 255).

In "A1_pencil_main.py":

Task 4.2: Change the attribute `*canvas_type*` into "*color*" when initializing the canvas at the beginning of the *main*() function.

By running the *main* Python file, you should see **Figure 5**:

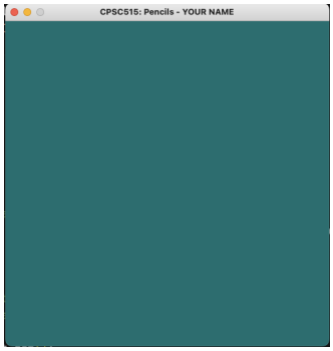


Figure 5: Task 4's result.

Task 5: Making A Stamp

When **right-clicking** on the canvas, instead of turning on one pixel, turn on the pixel being clicked and its neighboring pixels to form a specific pattern – a flower!

In "A1_pencil_canvas.py":

Task 5.1: Implement the `*drawFlower*()` function. This function should draw a flower centered on the mouse's input position, see **Figure 6**.

Things to keep in mind:

- Feel free to make your flower any color you like
- Make sure to use your `posToIndex` function to access pixel colors in canvas data.
- Handle edge cases as well!

In "A1_pencil_main.py":

Task 5.2: Call the `drawFlower()` function whenever the mouse's right button is clicked on the canvas. By running the *main* Python file, you should see **Figure 6**:

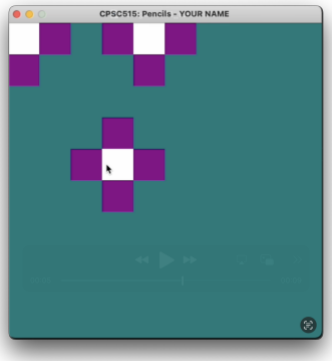


Figure 6: Task 5's result - stamp flowers by **right-clicking** on the canvas.

Task 6: Implement A Pencil

You will implement a pencil that draws wherever the **mouse's left button** is held down. By running the *main* Python file, you should see **Figure 7**:

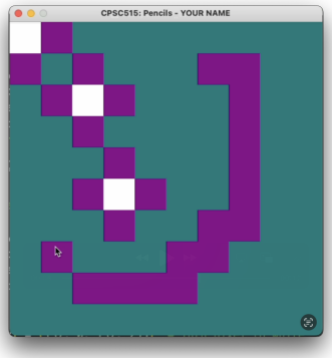


Figure 7: Task 6's result – draw on canvas by holding the left button.

Grading:

- General utility functionality (**10 pts**)
- Task 2: Create a Heart (**12 pts**)
- Task 3: Interacting with the Mouse (**13 pts**)
- Task 5: Making a Stamp (**15 pts**)
- Task 6: Implement a Pencil (**10 pts**)
- Software engineering, efficiency, & stability (**10 pts**)
- Written report: clarity, formatting, & comprehensiveness (**30 pts**)

Assignment 1 Submission Guideline

- **(70%) Two Python files**: "**A1_pencil_canvas.py**" and "**A1_pencil_main.py**", contain your complete code for **Tasks 2, 3, 5, 6** in this document; make sure your code is up and running.
    - Modify the window title to include your name. Search for "CPSC515: Pencils - YOUR NAME" in the main file.
- **(30%) A write-up (PDF)** detailing your approach, implementation, and results for **Tasks 2, 3, 5, 6**.
    - **For each task:**
        - Methods & Implementation:
            - Describe your approach to implementing the task; outline the key algorithms and techniques you use to address the task.
            - Explain any important design decisions and how they impact your implementation.

        - DO NOT copy your code here. Instead, use your own words to explain your solution. You may use line numbers to reference the code.
    - Results
        - Provide screenshots of your results for each task, showcasing key features or outputs. In these screenshots, **you must show the entire window, including your name on the window title.**
        - **For Task 5: "Make a stamp", your screenshot should demonstrate the handling of the edge cases.**
        - Clearly label and describe what each screenshot represents.
        - If applicable, discuss performance observations, challenges encouraged, and how they were addressed.
- Formatting
    - The report should be well-organized, concise, and clearly written in a structured format.
    - Use headings, bullets, and concise explanations where appropriate.
    - Ensure the screenshots are clearly labeled.
    - Submit your report as a PDF.

Using AI/Internet Resources in Assignments: AI tools and online resources can support your learning and provide inspiration, but your final work must demonstrate your own understanding and implementation. Refer to the course syllabus for specific guidelines on AI use in assignments. Be sure to properly cite and acknowledge any AI or internet sources you use, explaining where and how they contributed to your work in your written report.