adobe substance designer

Node-Based Image Manipulation Interface Assignment

Overview

In this assignment, you will create a node-based interface for image manipulation in C++. Your application should allow users to load images, process them through a series of connected nodes, and save the resulting output. This interface should function similarly to node-based editors like Substance Designer, where operations are represented visually and can be connected in various configurations.

Requirements

Core Functionality

1. Create a desktop application with a graphical user interface that displays:
    - A canvas area for creating and connecting nodes
    - A properties panel for adjusting node parameters
    - Basic file operations (open image, save result)
2. Implement a node system with:
    - Input nodes for loading images
    - Processing nodes for manipulating images
    - Output nodes for saving results
    - Connection system to link nodes together and define the processing pipeline
3. Develop at least 10 different processing nodes (see Required Nodes section)
4. Support real-time preview of results when parameters are changed
5. Implement proper error handling for invalid connections and operations

Required Nodes

You must implement the following nodes with the specified functionality:

Basic Nodes

1. **Image Input Node**
    - Load images from file system
    - Display image metadata (dimensions, file size, format)
    - Support common image formats (JPG, PNG, BMP)
2. **Output Node**
    - Save processed image to disk
    - Allow selection of output format and quality settings
    - Display a preview of the final image
3. **Brightness/Contrast Node**

  - Adjust image brightness with a slider (-100 to +100)
  - Adjust image contrast with a slider (0 to 3)
  - Provide reset buttons for each parameter
4. **Color Channel Splitter**
  - Split RGB/RGBA image into separate channel outputs
  - Option to output grayscale representation of each channel

Intermediate Nodes

5. **Blur Node**
  - Implement Gaussian blur with configurable radius (1-20px)
  - Option for uniform or directional blur
  - Include preview of kernel for educational purposes
6. **Threshold Node**
  - Convert to binary image based on threshold value
  - Include options for different thresholding methods (binary, adaptive, Otsu)
  - Display histogram of image to assist with threshold selection
7. **Edge Detection Node**
  - Implement both Sobel and Canny edge detection algorithms
  - Allow configuration of parameters (thresholds, kernel size)
  - Option to overlay edges on original image
8. **Blend Node**
  - Combine two images using different blend modes
  - Support at least 5 blend modes (normal, multiply, screen, overlay, difference)
  - Include opacity/mix slider

Advanced Nodes

9. **Noise Generation Node**
  - Create procedural noise patterns (Perlin, Simplex, Worley)
  - Allow configuration of noise parameters (scale, octaves, persistence)
  - Option to use noise as displacement map or direct color output
10. **Convolution Filter Node**
  - Provide a 3x3 or 5x5 matrix for custom kernel definition
  - Include presets for common filters (sharpen, emboss, edge enhance)
  - Display visual feedback of the kernel effect

Technical Requirements

1. Use C++ as the primary programming language
2. Utilize OpenCV or a similar library for image processing operations
3. Implement a graph-based execution system that:
  - Detects circular dependencies
  - Processes nodes in the correct order
  - Caches results to avoid redundant processing
4. Create an intuitive node connection interface
5. Ensure memory management is handled correctly
6. Provide detailed documentation for your code

Evaluation Criteria

Your submission will be evaluated based on:

1. **Functionality** (40%)
   - Does the application meet all requirements?
   - Are all nodes implemented correctly?
   - Is the node system flexible and working as expected?
2. **Code Quality** (30%)
   - Is the code well-structured and organized?
   - Are proper OOP principles applied?
   - Is the code documented and easy to understand?
   - How well are edge cases and errors handled?
3. **Performance** (15%)
   - Does the application handle large images efficiently?
   - Is the UI responsive during processing?
   - Is memory managed appropriately?
4. **User Experience** (15%)
   - Is the interface intuitive and easy to use?
   - Are operations visually clear?
   - Is appropriate feedback provided to the user?

Getting Started

To help you get started, we recommend:

1. First focus on creating the node framework and connection system
2. Implement the basic nodes to establish the pipeline
3. Add more complex nodes once the system is functional
4. Refine the UI and user experience
5. Optimize for performance and handle edge cases

Submission Process

To submit your completed assignment, please follow these instructions carefully:

1. **Create a GitHub Repository**
   - Create a new **public** GitHub repository named "node-based-image-processor" or similar
   - Initialize it with a README.md file
2. **Development Process Requirements**
   - Make regular commits throughout your development process
   - Your commit history should demonstrate incremental progress (not just one or two large commits)
   - Include descriptive commit messages that explain what changes were made
   - We recommend making commits after completing each major component or feature

3. **Repository Structure**
    - Organize your code as outlined in the assignment
    - Include a comprehensive README.md with:
        - Project overview
        - Features implemented
        - Build instructions
4. **Documentation**
    - Document your code with clear comments
    - Include a design document explaining your architecture decisions
    - Document any third-party libraries used and why
5. **Screen Recording**
    - Create a 3-5 minute screen recording demonstrating your application
    - Show all implemented nodes in action
    - Demonstrate connecting nodes and processing images
    - Explain any particularly interesting or challenging aspects of your implementation
    - Upload the recording to a hosting site (YouTube, Vimeo, Google Drive, etc.) and ensure it's publicly accessible
6. **Final Submission**
    - Complete the Google Form submission: https://forms.gle/wz3ar8V42ro3wSYVA
    - Include:
        - Your full name and contact information
        - Link to your GitHub repository
        - Link to your screen recording
        - Brief summary of your approach (500 words max)
        - Any notes on known issues or limitations
        - Approximate time spent on the assignment

Submission Deadline

- All submissions must be completed by: 15th April 2025; 11:59 PM
- Late submissions will not be accepted

Evaluation Criteria

Your submission will be evaluated based on:

- **Functionality**: Does the application meet all requirements?
- **Code Quality**: Is the code well-structured, clean, and properly commented?
- **Architecture**: Is the node system well-designed and extensible?
- **UI/UX**: Is the interface intuitive and responsive?
- **Performance**: Does the application handle images efficiently?
- **Creativity**: Any additional features or innovative approaches?

Important Notes

- Ensure all code you submit is **your own work**. Properly cite any external resources or libraries used.
- Your repository must remain public until the evaluation process is complete.

Questions

If you have any questions about the assignment, mail at rahul@mixar.app or satyam@mixar.app

We look forward to seeing your creative solutions!