PixelPlot: C++ Image Processing with GPU Shader DSL

Overview:

This project demonstrates an image processing pipeline built in C++, using OpenGL fragment shaders.

It integrates a mini DSL (Domain-Specific Language) to define filters like grayscale, blur, etc.,

and dynamically generates GPU shader code to process images.

Key Concepts:

1. CPU-Based Pipeline (image_pipeline/)

- Uses stb_image to load and save images.
- Applies filters like grayscale and blur using C++ loops.
- Good for understanding the basics and validating logic without GPU.

2. GPU-Based Pipeline (pixel_plot/)

- Uses OpenGL and GLFW to initialize a rendering context.
- Accepts a .dsl file like:

  grayscale

  blur 5
- Parses the DSL to generate a fragment shader (generated.frag).
- Loads an image into a texture and applies the shader in real-time.
- Runs very fast and efficient using GPU parallelism.

3. Shader Generation

- The DSL parser reads each line and translates it into GLSL code.
- Example:

    grayscale => converts RGB to average gray

    blur 5   => TODO: apply a 5x5 Gaussian kernel (not yet implemented)
- The final GLSL code is written and compiled at runtime.

4. Project Structure

- pixel_plot/src/   - Main OpenGL C++ code
- pixel_plot/shaders/   - Vertex + generated fragment shaders
- pixel_plot/include/   - stb_image headers
- pixel_plot/filters.dsl   - Filter steps in DSL

5. Tools Used

- stb_image.h / stb_image_write.h for image I/O
- OpenGL (via GLFW and GLAD) for rendering
- CMake for build configuration
- Git for version control

To Run:

```
mkdir build && cd build
cmake ..
cmake --build .
./PixelPilot
```