Project 0 Notes

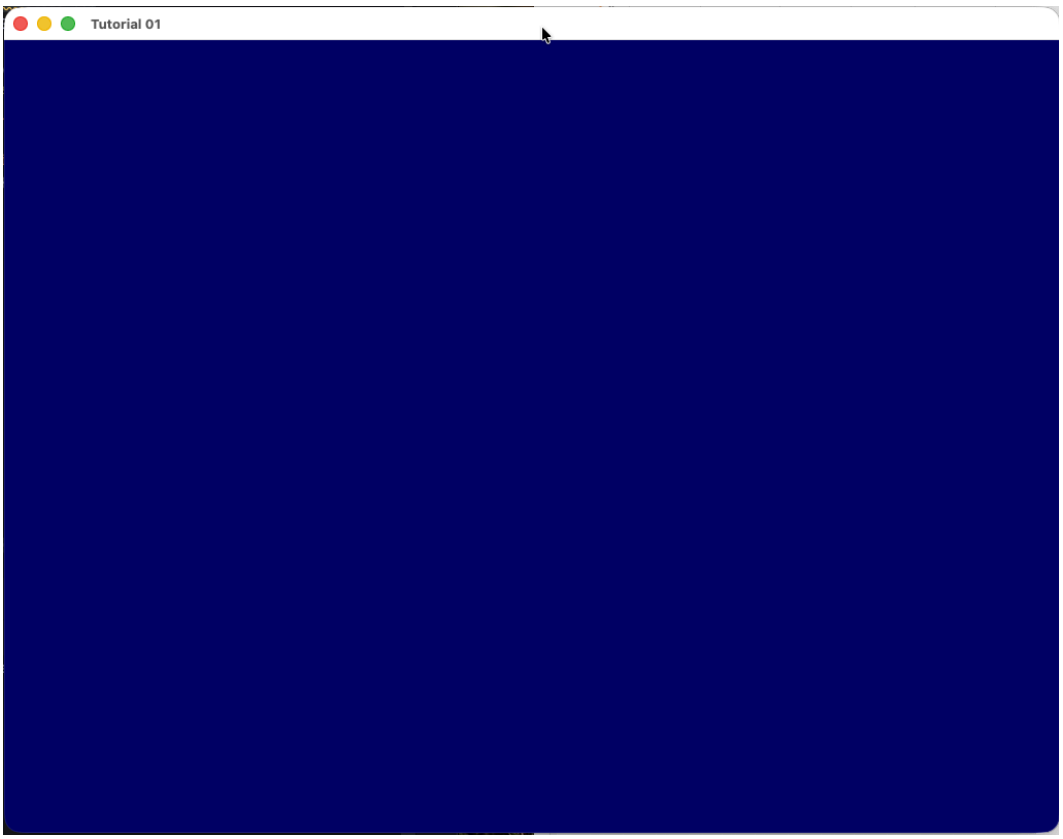
AI agents used to generate the solution: Codex, Claude, Copilot, Gemini.

Task 1: Tutorial

We replace the legacy CMakeList.txt for the OpenGL Tutorials with this updated one.

Tutorial 01

Running the program, a blank window with a dark blue background is displayed:



Tutorial 12

n = mod(18, 10) + 4 = 12

Running the program directly will encounter errors:

```
ARB_debug_output unavailable. You have to use glGetError() and/or gDebugger to catch
mistakes.

Compiling shader : StandardShading.vertexshader
```

```
Compiling shader : StandardShading_WithSyntaxErrors.fragmentshader

ERROR: 0:4: 'HELLO' : syntax error: syntax error


Linking program

ERROR: One or more attached shaders not successfully compiled
```

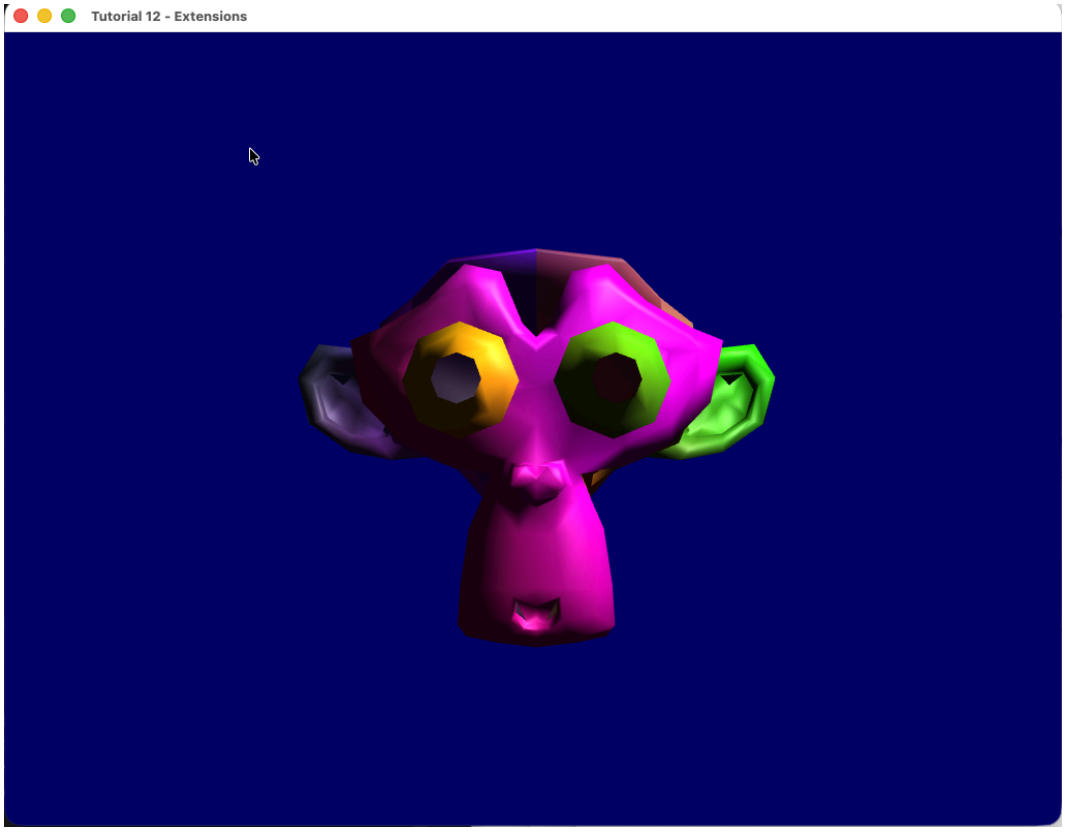
According to the error outputs, line 4 in `StandardShading_WithSyntaxErrors.fragmentshader` contains a syntax error.

Navigating to the file, we find that the tutorial intentionally introduces this error:

```
// a big syntax error to demonstrate the point...

HELLO WORLD
```

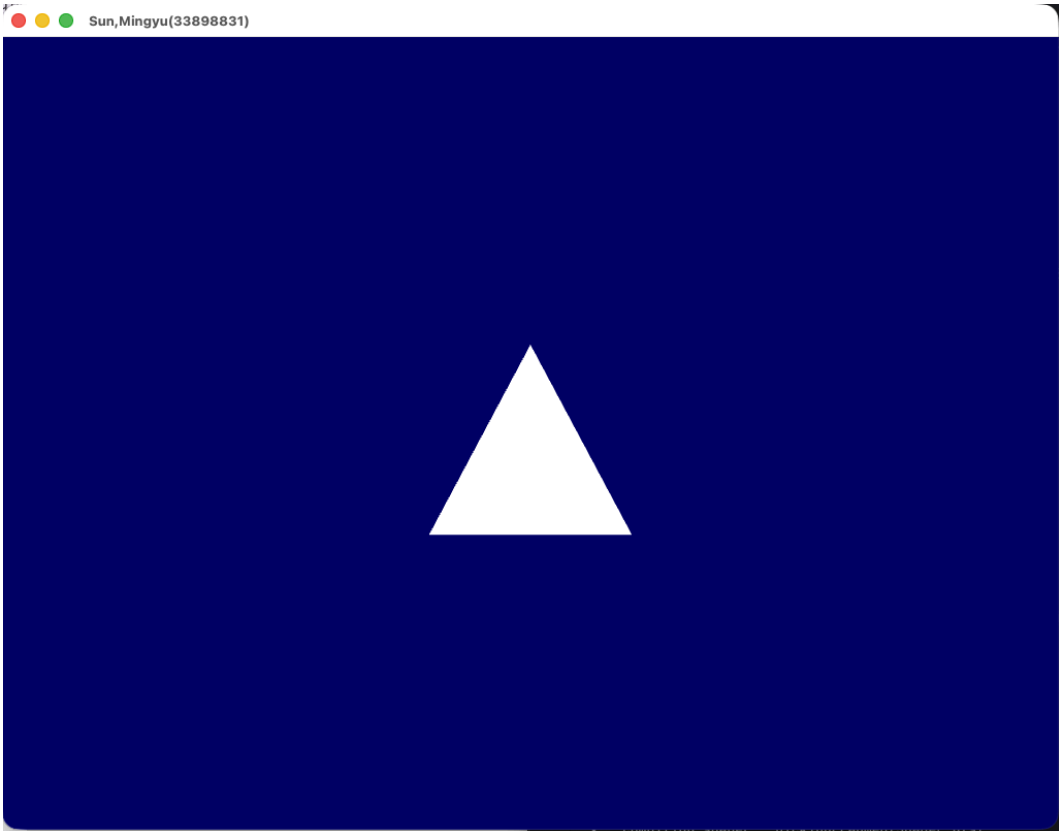
By commenting out this line and running the program again, we can get the following figure:



Task 2: Project 1 set up

Similar to Task 1, we update the CMakeLists.txt to align with newer CMake standards.

After changing the title of the window in source/main.cpp, we run the program, and get the following figure:



Insights and Observations

- The tutorial introduces OpenGL with almost no prerequisites. However, an update to CMakeLists.txt is required to utilize it.
- OpenGL itself is platform-independent, but window creation, input handling, and context management are entirely platform-dependent.
- We must explicitly provide platform-specific binaries to bridge the platform gap if we do not include raw source code.