Prototype of Texel Density Balancer Day-1

Rameel Malik

August 11, 2025

Abstract

This is day 1 of learning cuda pipeline. All the progress done in day 1 is in this report and project files are available in respective folders

1 What is GLFW and why did we use it?

- **Full form:** *Graphics Library Framework.*
- **Purpose:** GLFW is a **small library** that handles:
  - Creating a **window** on your desktop.
  - Creating an **OpenGL context** inside that window.
  - Handling **input events** like keyboard/mouse presses.

1.1 Why does OpenGL need this?

- **OpenGL** (often shortened to **GL**) is an API for drawing 2D/3D graphics.
- But OpenGL by itself **cannot create a window**.

  It only knows how to draw *inside* a window **after** a "GL context" is created.
- A **GL context** is like a "graphics state machine" the driver sets up — it holds all the GPU state (textures, shaders, buffers).

  Without it, there's nowhere for GL commands to run.
- On Windows, creating a GL context directly means dealing with old, messy Win32 APIs — GLFW makes this simple and cross-platform.

So:

- GLFW: "Here's your window and a working GL context, you can now issue GL commands."
- Without GLFW, we'd have to write $\tilde{1}00$ lines of Windows-specific boilerplate just to get the same thing.

2 What is GL?

Stands for **OpenGL** — *Open Graphics Library.*

It's a standard API for telling the GPU how to draw things.

Example GL commands:

- `glClearColor(...)` — "set background color"
- `glDrawArrays(...)` — "draw triangles"
- `glTexImage2D(...)` — "upload a texture to the GPU"

These functions are not built into the C++ language — they come from your graphics driver.

3 What is GLEW and why did we use it?

- **Full form:** *OpenGL Extension Wrangler.*
- **Purpose:** On Windows, your system's `gl.h` header **only contains OpenGL 1.1 function names** from the 1990s.

  Any newer GL features (like those from GL 3.3, which we're using) **must be loaded at runtime**.
- **GLEW** is a **loader library**: it looks up the actual memory addresses of the newer GL functions inside your GPU driver and stores them in function pointers, so you can call them in your code.

Without GLEW (or an alternative like GLAD):

- If you type `glGenBuffers()` on Windows, the compiler will say "never heard of it."
- With GLEW, it knows exactly where in the driver that function lives.

4 What is `glewExperimental = GL`$_T RUE;|$?

This is just a setting before you call `glewInit()`.

Why: Some modern OpenGL functions are marked as "experimental" in GLEW's database — especially in core profiles (like our OpenGL 3.3 Core).

Setting this to `GL_TRUE` says: "I don't care if it's called experimental — give me **all available functions**."

Without it, certain newer functions might not be loaded, even though your GPU supports them.

5 What is the render loop?

The next code will be directly imported from a file (Random Code now)
texitemize

Render loop = the `while (!glfwWindowShouldClose(window)) { ... }` block.

This is the heartbeat of any interactive graphics program.

Every frame, you typically:

1. Poll for input (keyboard, mouse).
2. Update your game/scene logic.
3. Draw the scene.
4. Swap the finished frame to the screen.

Why we need it:

- Without a loop, the window would appear for 1 frame (less than a millisecond) and then the program would exit.
- In *our* Day-1 program, the render loop does:
  - `glfwPollEvents();` → handle keyboard/window events.
  - `glfwGetFramebufferSize()` → get the window size in pixels.
  - `glViewport(...)` → tell GL the drawing area.
  - `glClearColor(...)` + `glClear(...)` → fill with a dark blue color.
  - `glfwSwapBuffers(window);` → show the color on the screen.

It repeats this 60 times per second (because we set VSync on with `glfwSwapInterval(1)`).

6 How did we print GPU info?

OpenGL gives you some built-in string queries:

```
1    glGetString(GL_VENDOR);     // e.g., "NVIDIA Corporation"
2    glGetString(GL_RENDERER);   // e.g., "NVIDIA GeForce RTX 3090/PCIe/SSE2"
3    glGetString(GL_VERSION);    // e.g., "3.3.0 NVIDIA 580.88"
```

These come from the graphics driver **after** the context is created.

If you called them before creating a context, they would return `nullptr` because no GPU state exists yet.

Why this is useful:

- Confirms you're running on the GPU you expect.
- Shows the GL version — important if you need features from a certain version.