Mục lục

Thư viện openGL

anthule

Ngày 20 tháng 9 năm 2023

1 Giới thiệu

OpenGL (Open Graphics Library) là một API đa nền tảng, đa ngôn ngữ cho render đồ họa vector 2D và 3D. API thường được sử dụng để tương tác với bộ xử lý đồ họa (GPU), nhằm đạt được tốc độ render phần cứng

- API: Application Programming Interface: là một giao diện mà một hệ thống máy tính hay ứng dụng cung cấp để cho phép các yêu cầu dịch vụ có thể được tạo ra từ các chương trình máy tính khác, và/hoặc cho phép dữ liệu có thể được trao đổi qua lại giữa chúng.
- GPU: Graphics Processing Unit Là bộ Vi xử lý chuyên phân tích những khối dữ liệu hình ảnh. GPU còn xử lý thông tin đa luồng, song song và bộ nhớ ở tốc độ cao. Kỹ thuật GPU đang dần trở nên dễ lập trình, cung cấp nhiều tiềm năng cho việc tăng tốc xử lí cho nhiều chương trình với nhiều mục đích khác nhau, hơn cả chíp xử lí thông thường (CPUs).

- render: gọi tắt là **kết xuất**, là một quá trình kiến tạo một hình ảnh từ một mô hình (hoặc một tập hợp các mô hình) thành một cảnh phim hoặc hình ảnh nào đó bằng cách sử dụng phần mềm máy tính.

  - Mô hình: là mô tả của các đối tượng ba chiều bằng một ngôn ngữ được định nghĩa chặt chẽ hoặc bằng một cấu trúc dữ liệu. Mô tả này bao gồm các thông tin về hình học, điểm nhìn, chất liệu và bố trí ánh sáng của đối tượng. Hình ảnh này có thể là một hình ảnh số (digital image) hoặc một hình ảnh đồ họa điểm (raster graphics image). Thuật ngữ này có thể tương đồng với "quá trình một họa sĩ vẽ" một phong cảnh nào đấy.

2 Cách sử dụng OpenGL

2.1 Cài đặt

1. **Cài trên visual studio code**

   Cài theo `https://www.youtube.com/watch?v=Y4F0tI7WlDs` Để có được file glwr3.dll thì ta tải glwr bản binary.

   Tham khảo `https://github.com/Coddeus/opengl_cpp_vscode_sample` file main.cpp và tasks.json.

2. **Cài trên visual studio 2022**

   - Install Cmake
   - Tải GLFW binary
   - Lấy repo `https://github.com/JoeyDeVries/LearnOpenGL` rồi chạy file CMakeLists.txt trên visual studio 2022, ấn build solution.

Một số thư viện

1. **Thư viện GLFW (Graphics Library Framework))**

   - GLFW là một thư viện mã nguồn mở được sử dụng để tạo cửa sổ và quản lý sự kiện đồ họa trong ứng dụng OpenGL.
   - Nó cung cấp các chức năng cơ bản như tạo cửa sổ, xử lý sự kiện bàn phím và chuột, và quản lý vòng lặp chính cho ứng dụng OpenGL.

- GLFW là một thư viện nhẹ và đơn giản, thích hợp cho các dự án OpenGL cơ bản và các ứng dụng đơn giản.

2. **Thư viện GLAD (Multi-Language GL/GLES/EGL/GLX/WGL Loader-Generator based on the official specs.)**

   OpenGL là một tiêu chuẩn hoặc một đặc tả đồ họa, không phải là một phần mềm hoàn chỉnh được cài đặt sẵn trên máy tính. Thay vào đó, nó chỉ định cách các chức năng đồ họa cơ bản hoạt động và tương tác với card đồ họa. Để thực hiện việc vẽ đồ họa bằng OpenGL, bạn cần một trình điều khiển đồ họa (driver) cụ thể cho card đồ họa bạn đang sử dụng.

   Mỗi nhà sản xuất card đồ họa có thể cung cấp một phiên bản OpenGL driver khác nhau dựa trên phiên bản card đồ họa của họ và các tính năng đặc biệt của nó. Điều này dẫn đến việc có nhiều phiên bản khác nhau của driver OpenGL.

   Vấn đề là vì có nhiều phiên bản driver khác nhau, nên vị trí cụ thể của các hàm trong driver OpenGL không thể biết trước tại thời điểm biên dịch mã nguồn của bạn. Do đó, bạn cần phải truy vấn vị trí của các hàm OpenGL này trong driver tại thời điểm chạy (run-time).

   Điều này thực hiện thông qua thư viện như GLAD hoặc GLEW, làm cho việc truy cập các hàm OpenGL trở nên dễ dàng hơn. Bạn sẽ sử dụng các hàm như gladLoadGLLoader() để nạp các con trỏ hàm OpenGL và sau đó sử dụng các con trỏ này để gọi các chức năng OpenGL cụ thể trong mã của bạn.

Một số vấn đề

1. Làm sao để chạy nhiều file cpp trong vscode

2.2 Tạo cửa sổ

2.2.1 Theo kiểu cũ - ở LearnOpenGL

1. Tại thư mục src, tạo file main.cpp.

2. Tại dòng đầu của file main.cpp, gọi thư viện glad.h (glad trước glwr thì mới gọi ra được con trỏ các hàm của glwr), rồi gọi thư viện glwr.h.

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
```

3. **glfwInit()**:

   Việc sử dụng $\#include < GLFW/glfw3.h >$ trong mã nguồn của bạn là để bao gồm các khai báo và định nghĩa liên quan đến GLFW. Tuy nhiên, việc đưa mã nguồn của thư viện vào mã của bạn chỉ giúp bạn truy cập các khai báo và hàm từ GLFW mà bạn có thể gọi trong mã của mình. Nó không thực sự khởi tạo hoặc quản lý bất kỳ thành phần cụ thể nào của GLFW cho bạn.

   glfwInit() là một hàm cụ thể của GLFW được sử dụng để khởi tạo thư viện GLFW và chuẩn bị môi trường đồ họa. Khi bạn gọi glfwInit(), nó thực sự thực hiện công việc khởi tạo thư viện và cài đặt các tài nguyên cần thiết cho việc làm việc với đồ họa.

4. **glfwWindowHint()**:

   Hàm glfwWindowHint trong thư viện GLFW được sử dụng để đặt các cài đặt cho cửa sổ đồ họa mà bạn sẽ tạo sau đó bằng hàm glfwCreateWindow. Dưới đây là một số cài đặt phổ biến mà bạn có thể sử dụng với glfwWindowHint:

   - `glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, majorVersion)` và `glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, minorVersion)`:
     Đặt phiên bản của OpenGL bạn muốn sử dụng. Ví dụ, để đặt phiên bản OpenGL 4.5, bạn có thể sử dụng

     ```
     glfwWindowHint(
         GLFW_CONTEXT_VERSION_MAJOR, 4);
     glfwWindowHint(
         GLFW_CONTEXT_VERSION_MINOR, 5);
     ```

   - `glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE)`:
     Đặt hồ sơ OpenGL bạn muốn sử dụng. Hồ sơ `GLFW_OPENGL_CORE_PROFILE` sử dụng phiên bản core của OpenGL, trong khi `GLFW_OPENGL_COMPAT_PROFILE` sử dụng phiên bản tương thích ngược.

     ```
     glfwWindowHint(GLFW_OPENGL_PROFILE,
         GLFW_OPENGL_CORE_PROFILE);
     ```

5. **glfwCreateWindow()**:

```
GLFWwindow* window = glfwCreateWindow(800, 600,
    "LearnOpenGL", NULL, NULL);
```

Hàm glfwCreateWindow yêu cầu hai đối số đầu tiên lần lượt là chiều rộng và chiều cao của cửa sổ. Đối số thứ ba cho phép bạn đặt tên cho cửa sổ; hiện tại, bạn gọi nó là "LearnOpenGL" nhưng bạn có thể đặt tên tùy ý. Bạn có thể bỏ qua hai đối số cuối cùng. Hàm trả về một đối tượng GLFWwindow mà sau này bạn sẽ cần cho các hoạt động GLFW khác. Sau đó, bạn chỉ định cho GLFW làm cho ngữ cảnh của cửa sổ của bạn trở thành ngữ cảnh chính trên luồng hiện tại.

6. Giữ cửa sổ ko bị đóng lại:

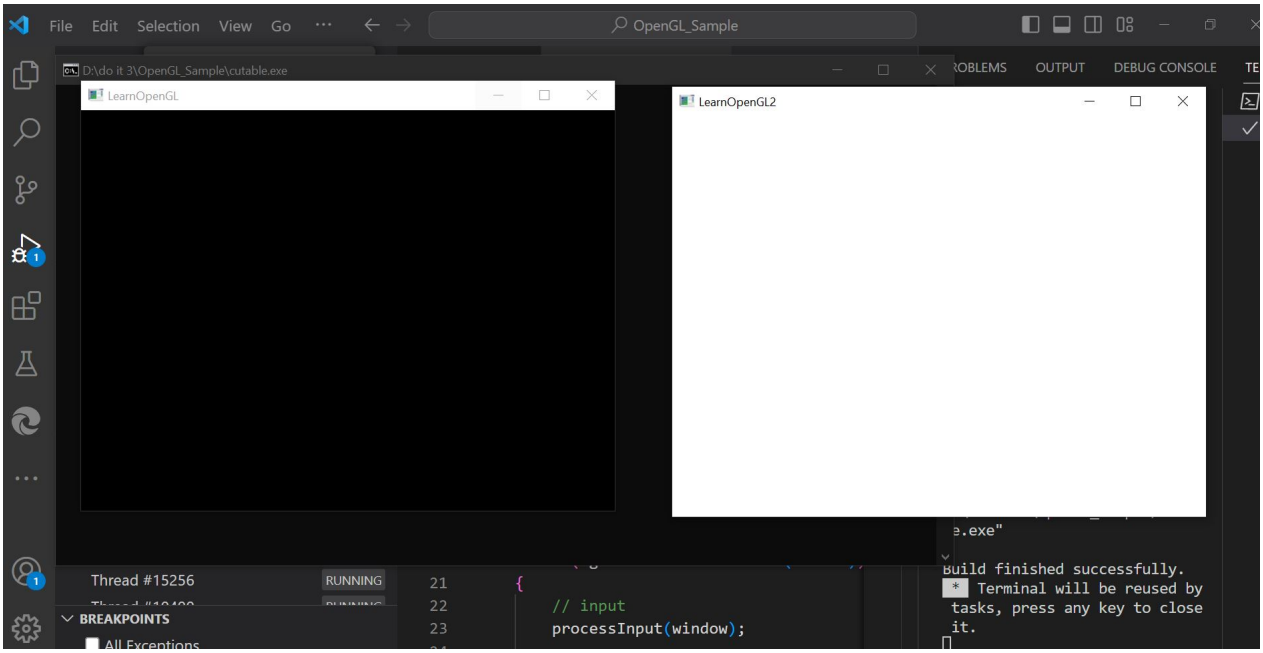
```
while(!glfwWindowShouldClose(window))
{
    glfwSwapBuffers(window);
    glfwPollEvents();
}
```

- while (!glfwWindowShouldClose(window)): Đây là một vòng lặp chạy liên tục cho đến khi biểu thức glfwWindowShouldClose(window) trả về giá trị true. Hàm này kiểm tra xem GLFW đã được hướng dẫn để đóng cửa sổ hay chưa. Nếu cửa sổ được đóng, biểu thức này trả về true và vòng lặp dừng chạy, sau đó bạn có thể đóng ứng dụng.
- glfwSwapBuffers(window): Hàm này thực hiện việc hoán đổi (swap) hai bộ đệm màu. GLFW sử dụng hai bộ đệm màu để hiển thị hình ảnh trên cửa sổ. Một bộ đệm được vẽ và lắng nghe dữ liệu đồ họa mới (color buffer), trong khi bộ đệm còn lại hiển thị nội dung đã vẽ. Khi bạn gọi glfwSwapBuffers(window), nó sẽ chuyển đổi giữa hai bộ đệm, làm cho nội dung đã vẽ xuất hiện trên cửa sổ.
- glfwPollEvents(): Hàm này kiểm tra xem có sự kiện nào được kích hoạt (như sự kiện nhập từ bàn phím, chuyển động chuột, vv.), cập nhật trạng thái của cửa sổ và gọi các hàm tương ứng (các hàm được đăng ký thông qua các phương thức gọi lại callback). Nó giúp bạn xử lý sự kiện và tương tác người dùng trong ứng dụng của bạn.

- Một số dòng code khác: ko có xài thì cũng vẫn hiện cửa sổ lên.
  - `glViewport(0, 0, 800, 600)`: Đặt kích thước
  - `glClearColor(0.2f, 0.3f, 0.3f, 1.0f)`: Đặt màu nền
  - `glClear(GL_COLOR_BUFFER_BIT)`: Xóa màu nền
  - `glfwTerminate()`: Đóng GLFW
  - `glfwSetFramebufferSizeCallback(window, framebuffer_size_callback)`: Đăng kí hàm callback để khi thay đổi kích thước cửa sổ thì cửa sổ vẫn hiện lên
- Tạo 2 cửa sổ:

```
GLFWwindow* window1 = glfwCreateWindow(
    width1, height1,
"Cua so 1", NULL, NULL);
if (window1 == NULL) {
    glfwTerminate();
    return -1;
}

GLFWwindow* window2 = glfwCreateWindow(
    width2, height2,
"Cua so 2", NULL, NULL);
if (window2 == NULL) {
    glfwTerminate();
    return -1;
}
glfwMakeContextCurrent(window1);
glfwMakeContextCurrent(window2);
```



- Dòng code xử lí lỗi:
  - Lỗi ko tạo được cửa sổ

```
if (window == NULL)
{
    std::cout << "Failed to
       create GLFW window"
      << std::endl;
    glfwTerminate();
    return -1;
}
```

  - Lỗi ko tạo được ngữ cảnh

```
glfwMakeContextCurrent(window);
if (!gladLoadGLLoader((GLADloadproc)
   glfwGetProcAddress))
{
    std::cout << "Failed to initialize
       GLAD" << std::endl;
    return -1;
```

```
    }
```

2.2.2 Theo kiểu mới - refacted code

Tạo 1 file tên là `standard_includes.cpp` trong thư mục include. File này sẽ chứa các hàm init_glfw chứa các hàm con để tạo cửa sổ, tạo ngữ cảnh, tạo vòng lặp để giữ cửa sổ ko bị đóng lại, xử lí lỗi, đóng GLFW, ..

Sau đó trong hàm main ta gọi hàm init_glfw ra và tập trung vào việc vẽ hình.

2.3 Vẽ một tam giác