

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN**

BÁO CÁO CÁ NHÂN

CỜ VUA 3D

Học phần: 2111COMP104701 – Đồ họa máy tính

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
Thành phố Hồ Chí Minh, ngày 09 tháng 12 năm 2021**

TRƯỜNG ĐẠI HỌC SƯ PHẠM TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CÁ NHÂN
CỜ VUA 3D

Học phần: 2111COMP104701 – Đồ họa máy tính

Giảng viên hướng dẫn: Th.S Nguyễn Đỗ Thái Nguyên.

Sinh viên thực hiện:

+ NGUYỄN VĂN PHONG

4501104175

Thành phố Hồ Chí Minh, ngày 09 tháng 12 năm 2021

NHIỆM VỤ TRONG NHÓM.....	2
DANH MỤC CÁC HÌNH VẼ.....	3
CHƯƠNG 1. OBJECT PICKING	1
1.1 Stencil buffer.....	1
1.2 Object picking sử dụng Stencil buffer	1
1.3 Highlight đối tượng bằng stencil test.....	2
CHƯƠNG 2. RULE.....	3
CHƯƠNG 3. SKYBOX.....	4
3.1 Cubemap	4
3.2 Tạo một Skybox từ cubemap	4
TÀI LIỆU THAM KHẢO	6

NHIỆM VỤ TRONG NHÓM

- Chọn các quân cờ và ô cờ, Luật cho các quân cờ và game.

DANH MỤC CÁC HÌNH VẼ

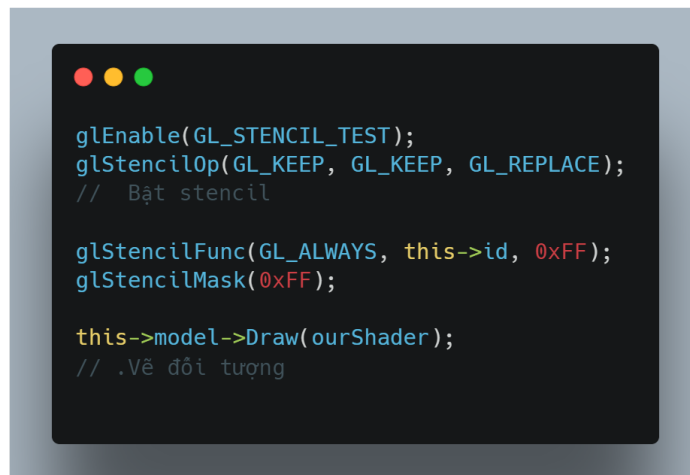
Hình 1-1 Code để định danh một đối tượng trong chương trình.....	1
Hình 1-2 Code lấy giá trị stencil bufer tại pixel đã nhấn chuột	1
Hình 2-1 Tổ chức hướng đối tượng hỗ trợ luật cờ	3
Hình 3-1 Skybox của Chess3D	4
Hình 3-2 Tạo một cubemap.....	4
Hình 3-3 Code đọc các texture từ file	5
Hình 3-4 wrappin và filtering cho cubemap.....	5
Hình 3-5 Kết xuất Skybox ra màn hình.....	5

CHƯƠNG 1. OBJECT PICKING

Trong dự án cờ vua 3D này, việc phát triển chức năng chọn được các đối tượng như cờ hay ô là điều bắt buộc. Có nhiều phương pháp để sử dụng và sau khi tham khảo thì em chọn phương pháp sử dụng stencil buffer để định danh và chọn đối tượng vì lý do dễ cài đặt và có thể chọn đối tượng chính xác hơn các phương pháp khác.

1.1 Stencil buffer

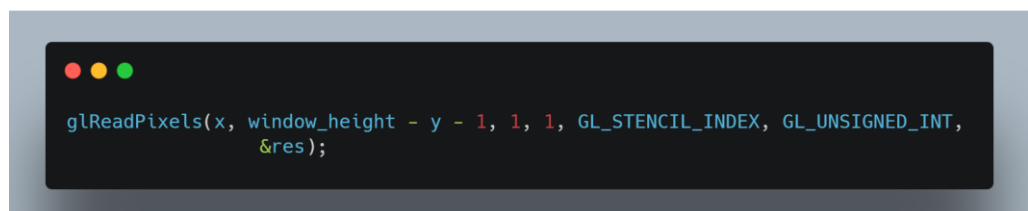
Stencil buffer chứa 8bit cho mỗi giá trị stencil tương đương 256 giá trị giá trị stencil khác nhau cho mỗi điểm ảnh. Chúng ta có thể đặt giá trị này tùy ý vì thế có thể dùng id của đối tượng để đặt cho giá trị này. Do mỗi điểm ảnh chỉ có tối đa 256 giá trị vậy nên chúng ta có thể định danh tối đa cho 256 vật thể (0 – 255), đây cũng là nhược điểm của phương pháp này, nhưng vì cờ vua chỉ có 32 quân cờ và 64 ô nên có bỏ qua nhược điểm này.



Hình 1-1 Code để định danh một đối tượng trong chương trình

1.2 Object picking sử dụng Stencil buffer

Sau khi đã định danh một vật thể chúng ta dùng hàm **glReadPixels** để lấy giá trị của stencil buffer tại pixel đó



Hình 1-2 Code lấy giá trị stencil buffer tại pixel đã nhấn chuột

x, y là tọa độ nhận được khi chuột nên chúng ta phải đổi sang hệ tọa độ của OpenGL, giá trị nhận được biến res sẽ là giá trị của stencil buffer tại pixel đó cũng như là id của đối tượng.

1.3 Highlight đối tượng bằng stencil test

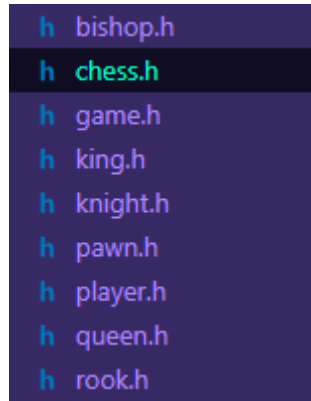
Để highlight được đối tượng, em đã làm các bước:

- Đặt stencil op thành GL_ALWAYS trước khi vẽ một đối tượng, cập nhật stencil buffer với giá trị là id của đối tượng.
- Render đối tượng.
- Tắt stencil writing và depth testing.
- Phóng to đối tượng lên một chút.
- Sử dụng một fragment shader khác có output là màu đỏ.
- Vẽ lại đối tượng nhưng chỉ trên những pixel có giá trị stencil khác id đối tượng, lúc này ta sẽ nhận được viền.
- Bật lại depth test.

CHƯƠNG 2. RULE

Phần này em thiết kế theo hướng đối tượng các quân cờ và tính toán các vị trí mà quân đó có thể đi. Class Player có các thuộc tính thông tin người chơi như các quân cờ đang có, class Game có thuộc tính trạng thái game như turn, các hàm hỗ trợ kiểm tra trường hợp chiếu (check), chiếu hết (checkmate), xử lý các logic việc chọn cờ như ngăn chặn người chơi 2 đi trong lượt người chơi 1, v.v...

Các file được thiết kế:



```
h bishop.h
h chess.h
h game.h
h king.h
h knight.h
h pawn.h
h player.h
h queen.h
h rook.h
```

Hình 2-1 Tổ chức hướng đối tượng hỗ trợ luật cờ

CHƯƠNG 3. SKYBOX

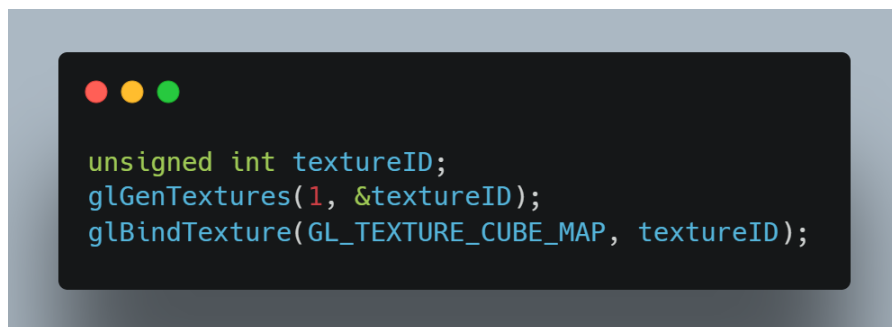


Hình 3-1 Skybox của Chess3D

Skybox nhóm em làm ở đây là bầu trời xung quanh. Để làm được điều này nhóm em sử dụng cubemap trong OpenGL.

3.1 Cubemap

Cubemap là một texture chứa 6 texture riêng lẻ mà mỗi texture tạo thành một mặt của hình khối.



Hình 3-2 Tạo một cubemap

3.2 Tạo một Skybox từ cubemap

Sau khi tạo xong một cubemap, em duyệt qua các 6 file chứa texture là mặt trước, sau, trái, phải, trên, dưới và khởi tạo texture bằng lệnh `glTexImage2D`.

```

int width, height, nrChannels;
for (unsigned int i = 0; i < faces.size(); i++) {
    unsigned char *data = stbi_load(faces[i].c_str(), &width, &height, &nrChannels, 0);
    if (data) {
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0, GL_RGB, width, height, 0, GL_RGB,
                     GL_UNSIGNED_BYTE, data);
        stbi_image_free(data);
    } else {
        std::cout << "Cubemap texture failed to load at path: " << faces[i] << std::endl;
        stbi_image_free(data);
    }
}

```

Hình 3-3 Code đọc các texture từ file

Cuối cũng như các texture bình thường, em dùng hàm `glTexParameteri` chỉ định wrapping và filtering cho texture.

```

glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);

```

Hình 3-4 wrappin và filtering cho cubemap

Sau khi có được cubemap skybox, em sẽ scale kích thước lớn để cubemap bao trùm toàn bộ bàn cờ rồi vẽ đối tượng ra như vậy là em đã giải quyết được skybox.

```

glDepthMask(GL_FALSE);
skyboxShader.use();
glBindVertexArray(skyboxVAO);
glBindTexture(GL_TEXTURE_CUBE_MAP, cubemapTexture);
glDrawArrays(GL_TRIANGLES, 0, 36);
glDepthMask(GL_TRUE);

```

Hình 3-5 Kết xuất Skybox ra màn hình

TÀI LIỆU THAM KHẢO

- [1] Joey de Vries(2014), Learn OpenGL, <https://learnopengl.com>, truy cập ngày 20/11/2021.
- [2] Movania, Muhammad Mobeen. *OpenGL development cookbook*. Packt Publishing Ltd, 2013.
- [3] Victor Gordan (2021), OpenGL Course - Create 3D and 2D Graphics With C++, <https://www.youtube.com/watch?v=45MIykWJ-C4>, truy cập ngày 25/11/2021.