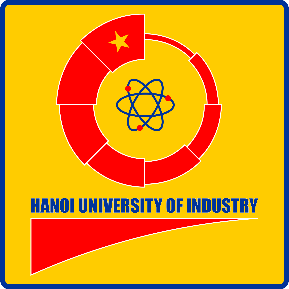
**Trường Đại Học Công Nghiệp Hà Nội**

**Khoa Công Nghệ Thông Tin**

**----------🙡🕮🙣----------**

****

**BÁO CÁO THỰC NGHIỆM**

**HỌC PHẦN ĐỒ HỌA MÁY TÍNH**

**Đề tài: Mô phỏng gian hàng máy bay đồ chơi**

**bằng OpenGL khả lập trình**

**và Visual C++**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **ThS.Vũ Minh Yến** |
| **Nhóm thực hiện:**  **Lớp:** | **Nhóm 2**  **20232IT6010001** |
| **Thành Viên:** | **Trần Thùy Dương – 2022600435**  **Nguyễn Minh Hằng – 2022605373**  **Hà Huyền Trang – 2022603537**  **Phan Thanh Huyền – 2022607046** |

**Hà Nội, 2024**

**MỤC LỤC**

[**LỜI MỞ ĐẦU** 4](#_Toc191138760)

[**Chương 1. Xác định và phân tích bài toán** 5](#_Toc191138761)

[1.1. Bài toán 5](#_Toc191138762)

[1.2. Mô tả các đối tượng cần thiết kế 5](#_Toc191138763)

[1.3. Mô tả bố cục khung cảnh chung 5](#_Toc191138764)

[1.4. Mô tả kịch bản của chương trình 5](#_Toc191138765)

[**Chương 2. Cài đặt chương trình** 5](#_Toc191138766)

[2.1. Kỹ thuật tạo mô hình 5](#_Toc191138767)

[2.2. Camera 6](#_Toc191138768)

[2.3. Chiếu sáng 7](#_Toc191138769)

[2.3.1. Directional light 7](#_Toc191138770)

[2.3.2. Point light 7](#_Toc191138771)

[2.4. Texture 8](#_Toc191138772)

[2.5. Giao diện (UI) 9](#_Toc191138773)

[**Chương 3. Kết quả đạt được** 10](#_Toc191138774)

[**KẾT LUẬN** 10](#_Toc191138775)

[**TÀI LIỆU THAM KHẢO** 11](#_Toc191138776)

# **LỜI MỞ ĐẦU**

Trong thời đại ngày nay, dưới sự bùng nổ của cuộc cách mạng công nghiệp 4.0, sự phát triển vượt bậc về công nghệ đồ họa đã mang lại cho con người chúng ta vô vàn tiện nghi, cùng những trải nghiệm vô cùng mới mẻ và chân thực.

Đồ họa máy tính là một lĩnh vực của khoa học máy tính nghiên cứu về toán học, các thuật toán cũng như các kĩ thuật để cho phép tạo, hiển thị và điều khiển hình ảnh trên màn hình máy tính. Đồ họa máy tính có liên quan ít nhiều đến một số lĩnh vực như đại số, hình học giải tích, hình học họa hình, quang học,... và kĩ thuật máy tính, đặc biệt là chế tạo phần cứng (các loại màn hình, các thiết bị xuất, nhập, các vỉ mạch đồ họa...). Đồ họa máy tính dần trở thành công cụ không thể thiếu trong nhiều lĩnh vực đời sống với con người. Theo đó, môn Đồ họa máy tính trở thành môn học vô cùng thiết thực và quan trọng đối với sinh viên khối ngành Công nghệ, Kỹ thuật.

Là sinh viên khoa Công nghệ thông tin trường Đại học Công Nghiệp Hà Nội, chúng em cũng được tiếp xúc với môn học đồ họa máy tính. Với những kiến thức và nghiên cứu trong quá trình học và được sự hướng dẫn của cô Vũ Minh Yến, chúng em đã thực hiện đề tài " Mô phỏng phòng làm việc bằng OpenGL khả lập trình và Visual C++".

Trong quá trình thực hiện nghiên cứu, có lẽ chúng em vẫn còn những thiếu sót, kết quả nghiên cứu chưa được hoàn thiện nhất. Vì vậy, chúng em mong thầy cô và các bạn có thể chia sẻ nhận xét và góp ý để kết quả nghiên cứu của chúng em hoàn thiện hơn. Chúng em xin chân thành cảm ơn!

# **Chương 1. Xác định và phân tích bài toán**

## 1.1. Bài toán

Bài toán mô phỏng phòng làm việc gồm:

- 3 bức tường và sàn nhà

## 1.2. Mô tả các đối tượng cần thiết kế

1.2.1. Mô hình 3 bức tường và sàn nhà:

## 1.3. Mô tả bố cục khung cảnh chung

## 1.4. Mô tả kịch bản của chương trình

# **Chương 2. Cài đặt chương trình**

## 2.1. Kỹ thuật tạo mô hình

2.1

## 2.2. Camera

Camera trong cảnh được xây dựng để có thể quay bằng chuột như các phần mềm 3D hiện đại:

* Zoom camera bằng middle mouse: Ta cần tính forward vector và di chuyển camera gần xa theo vector này theo giá trị của middle mouse sẽ mô phỏng được zoom in/out:

|  |
| --- |
| void engine::zoomCamera(float value) {  if (value > 0) // Scroll up  {  setCameraPos(CAM\_POS\_4 + CAM\_DIR\_4 \* mainCamera.zoomSpeed);  }  else if (value < 0) // Scroll down  {  setCameraPos(CAM\_POS\_4 - CAM\_DIR\_4 \* mainCamera.zoomSpeed);  } } |

* Camera xoay theo 2 trục horizontal và vertical bằng left mouse :

Để lấy giá trị hướng của chuột, ta có thể tính được vector(deltaX, deltaY) là vector hướng của chuột bằng cách lấy vị trí chuột hiện tại trừ đi vị trí chuột trước đó, vị trí chuột sẽ là tham số trong hàm **glutMotionFunc**:

|  |
| --- |
| int deltaX = mouseX - lastMouseX; int deltaY = mouseY - lastMouseY;  lastMouseX = mouseX; lastMouseY = mouseY; |

Horizontal chính là trục y, nên có ta quay camera xung quanh trục y bằng cách nhân position với ma trận rotateY:

|  |
| --- |
| float yaw = deltaX \* rotationSpeed; yawAngle += yaw;  mat4 yawRot = Angel::RotateY(-yaw); setCameraPos(yawRot \* vec4(CAM\_POS\_4)); |

Còn đối với vertical là vector camera right, nên ta phải xác định camera right trước rồi sau đó quay camera quanh trục này:

|  |
| --- |
| float pitch = deltaY \* rotationSpeed; vec3 vertical = rotateAxis(CAM\_POS\_3, pitch \* Deg2Rad, CAM\_RIGHT); setCameraPos(vertical); |

Hàm rotate axis sẽ quay position theo 1 trục bất kì dựa theo công thức Rodrigues. Tuy nhiên công thức này chỉ cho phép chúng ta quay theo axis từ -90 cho đến 90 độ nên ta chỉ có thể quay camera theo horizontal -90 đến 90. Tổng hợp 2 phép quay theo trục y và camera right thì ta có thể điều khiển camera như mong đợi.

Để áp dụng camera cho tất cả các objects trong cảnh, ta chỉ cần gọi hàm dưới đây và truyền vào model location và view location, hàm sẽ tự động bind các giá trị view, projection của camera vào ma trận view và projection trong shader:

|  |
| --- |
| void engine::useCameraMatrix(GLuint view\_location, GLuint projection\_location) {  auto camView = LookAt(mainCamera.eye\_position, mainCamera.target\_position, mainCamera.up\_direction);  glUniformMatrix4fv(view\_location, 1, GL\_TRUE, camView);  auto projection = Perspective(60, (float)mainCamera.window\_width / mainCamera.window\_height, mainCamera.zNear, mainCamera.zFar);  glUniformMatrix4fv(projection\_location, 1, GL\_TRUE, projection); } |

## 2.3. Chiếu sáng

Trong cảnh, có 2 loại ánh sáng: directional light và point light. Các ánh sáng là số nhiều và ta có thể tính được màu sắc final color bằng tổng màu sắc khi tính toán cho tất cả các lights có trong cảnh.

### 2.3.1. Directional light

Directional light trong cảnh sẽ là duy nhất và nó đại diện cho ánh sáng mặt trời (sun light). Với directional light, ta sẽ không cần quan tâm đến position và at, ta chỉ cần xác định hướng dirLight cho nó và truyền vào phong shader thay vì phải tính lại lightDir như trước.

|  |
| --- |
| // thay vì tính lightDir như này trong shader  vec3 lightDir = normalize(lightPos - FragPos);  // thì ta có thể truyền lightDir trực tiếp  uniform vec3 lightDir; |

### 2.3.2. Point light

Với point light, việc tính ánh sáng cũng như Directional light nhưng ánh sẽ suy giảm theo khoảng cách từ nguồn đến FragPos. Ta có thể mô phỏng nó bằng cách thấy màu sắc đã tính được theo Phong shader nhân với 1 hệ số phụ thuộc vào khoảng cách gọi là attenuation.

Attenuation =

Từ công thức trên, ta có thể thấy attenuation phụ thuộc vào khoảng cách d, d càng lớn thì attenuation càng nhỏ tức là màu sắc càng mờ. Các hệ số quadratic, linear, constant như là hệ số abc trong phương trình bậc 2, ta có thể tự định nghĩa cho ánh sáng này.

|  |
| --- |
| float distance = length(position - FragPos);  float attenuation = 1.0 / (constant + linear \* distance + quadratic \* (distance \* distance));  return color \* attenuation; |

Point ligh trong cảnh không phải là duy nhất, mà có thể có nhiều. Để tính màu cho object, ta chỉ cần phải lặp qua 1 mảng point light, sau đó cộng tất cả các giá trị này sẽ cho ra kết quả chiếu sáng nhiều nguồn sáng.

|  |
| --- |
| #define MAX\_POINT\_LIGHT\_COUNT 10 //std::vector<pointLight> pointLights = {} pointLight\* pointLights; int pointLightCount; |

Dưới đây là code shader, ta sẽ xác định uniform pointLightCount và lặp qua ánh sáng, tính màu cho từng ánh sáng và sau đó cộng tổng các giá trị màu này lại.

|  |
| --- |
| for(int i = 0; i < pointLightCount; i++) {  PointLight light = pointLights[i];  result += calculatePointLights(light); } |

Sau khi tính toán được directional light và point lights, ta cộng các kết quả lại để nhận được kết quả của chiếu sáng nhiều nguồn sáng.

|  |
| --- |
| // defaul color  vec3 result = vec3(0, 0, 0);  //calculate directional light result += calculateDirectionalLights(directionalLight);  //calculate point lights for(int i = 0; i < pointLightCount; i++) {  PointLight light = pointLights[i];  result += calculatePointLights(light); } |

## 2.4. Texture

Để có thể gắn một texture vào object, ta làm theo các bước sau:

1. Xác định texture coordinate

|  |
| --- |
| float vertices[] = {  // positions // colors // tex coords //normals  0.5f, 0.5f, 0.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, // top right  0.5f, -0.5f, 0.0f, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 0.0f, 0.0f, 1.0f, // bottom right  -0.5f, -0.5f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, // bottom left  -0.5f, 0.5f, 0.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f, 1.0f // top left }; |

1. Xác định cách đọc attribute pointer cho textCoord

|  |
| --- |
| GLuint textCoord\_loc = glGetAttribLocation(ourShader.program, "vTextCoord"); glVertexAttribPointer(textCoord\_loc, 2, GL\_FLOAT, GL\_FALSE, 11 \* sizeof(float), (void\*)(6 \* sizeof(float))); glEnableVertexAttribArray(textCoord\_loc); |

1. Gen texture và xác định texture parameters

|  |
| --- |
| glGenTextures(1, &sign\_texture);  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT); glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT); |

1. Tải texture từ image và bind texture (dùng thư viện stbi\_image)

|  |
| --- |
| unsigned char\* data = stbi\_load("plane\_sign.jpg", &width, &height, &nrChannels, 0); if (data) {  glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, width, height, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, data);  glGenerateMipmap(GL\_TEXTURE\_2D); }  glBindTexture(GL\_TEXTURE\_2D, sign\_texture); |

## 2.5. Giao diện (UI)

Giao diện cơ bản trong cảnh bao gồm text và button. Trong openGL, ta có thể vẽ text lên màn hình bnăgf hàm **glutBitmapCharacter.** Việc chúng ta cần làm là lặp qua 1 mảng const char\* và vẽ từng kí tự char lên màn hình tại tọa độ màn (x,y)

|  |
| --- |
| void ui::text2D(const char\* text, int fontSize, int x, int y, color color) {  glColor4f(color.x, color.y, color.z, color.w);  // TEXT position  glRasterPos2i(x, y);  // render text  for (const char\* c = text; \*c != '\0'; c++)  glutBitmapCharacter(font, \*c); } |

Còn đối với button, ta vẽ thêm 1 hình quad với size(x, y) rồi sau đó vẽ text như trên tại vị trí (x, y), sau đó ta có thể lắng nghe sự kiện button và gọi callback onClick bằng cách kiểm tra mouse(x, y) có nằm trong size(x, y) không. Button được triển khai như sau:

|  |
| --- |
| struct button {  int x, y;  int width, height;  void (\*onClick)();  void onEvent(int button, int state, int x, int y);  bool isInside(int mouseX, int mouseY) const; };  void ui::button::onEvent(int button, int state, int x, int y) {  if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_UP && isInside(x, y))  if (onClick)   onClick(); }  bool ui::button::isInside(int mouseX, int mouseY) const {  mouseY = window\_height - mouseY;   return (mouseX >= x && mouseX <= (x + width) &&  mouseY >= y && mouseY <= (y + height)); } |

# **Chương 3. Kết quả đạt được**

# **KẾT LUẬN**

Qua việc thực hiện nghiên cứu đề tài “Mô phỏng phòng làm việc bằng OpenGL khả lập trình và Visual C++” nhóm chúng em đã được biết thêm rất nhiều về môn đồ họa máy tính. Bên cạnh đó, việc làm nghiên cứu giúp chúng em đoàn kết hơn, từ đó cải thiện kỹ năng làm việc nhóm của mình.

Trong quá trình thực hiện đề tài có rất nhiều ý tưởng hay, độc đáo. Nhưng do kiến thức của chúng em hạn hẹp và thời gian không cho phép nên chúng em chưa thể thực hiện được những ý tưởng đó. Tuy nhiên chúng em đã cố gắng để xây dựng một chương trình hoàn chỉnh nhất, đẹp nhất. Trong quá trình xây dựng chương trình nhóm chúng em khó tránh khỏi những sai sót. Vì vậy chúng em rất mong rằng cô và các bạn cùng góp ý với nhóm em để có thể hoàn thành sản phẩm một cách hoàn chỉnh nhất.

Chúng em xin cảm ơn cô Vũ Minh Yến đã tận tình giảng dạy chúng em trong môn Đồ họa máy tính, giúp đỡ chúng em trong quá trình nghiên cứu đề tài và chia sẻ những tài liệu hay về thư viện OpenGl cũng như các kĩ năng lập trình.

Một lần nữa chúng em xin chân thành cảm ơn !

# **TÀI LIỆU THAM KHẢO**

1. Slide bài giảng và các tài liệu học tập của cô Vũ Minh Yến

2. Tài liệu, video trên Internet