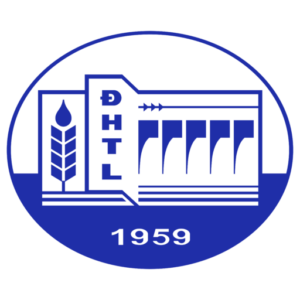
**BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN**

**PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI**

----🖎🕮✍----

****

**BÁO CÁO MÔN: Đồ họa máy tính**

**ĐỀ TÀI**

**MÔ PHỎNG ĐÊM NOEL BẰNG C++**

|  |  |
| --- | --- |
| **Sinh viên thực hiện** | **Mã sinh viên** |
| Nguyễn Văn Phong | 2251068230 |
| Bùi Thiện Phát |  |
| **Giảng viên hướng dẫn** | |
| Th.S Nguyễn Thị Kim Phụng | |

**TP HỒ CHÍ MINH NĂM 2O24**

# Lời Mở Đầu

Trong thế giới công nghệ ngày nay, đồ họa máy tính đã trở thành một phần không thể thiếu trong nhiều lĩnh vực, từ giải trí, nghệ thuật đến các ứng dụng thực tế như mô phỏng, thiết kế và giáo dục. Đặc biệt, đồ họa 3D đã mở ra những khả năng vô cùng phong phú và đa dạng, giúp tạo ra các không gian sống động, chân thực. Với sự phát triển mạnh mẽ của các công cụ lập trình và phần mềm đồ họa, việc tạo ra các mô hình 3D trở nên dễ dàng và hấp dẫn hơn bao giờ hết.

Đề tài "Đêm Noel bằng 3D" được thực hiện trong khuôn khổ môn **Đồ họa Máy tính** nhằm ứng dụng các kiến thức về đồ họa 3D để mô phỏng một không gian đêm Noel đầy màu sắc và ấm áp. Mục tiêu của đề tài này là tạo ra một cảnh vật 3D tái hiện không khí giáng sinh với các yếu tố đặc trưng như cây thông Noel, tuyết rơi, và ánh đèn lung linh, sử dụng các công cụ lập trình như Dev C++ kết hợp với thư viện OpenGL.

Thông qua việc thực hiện đề tài này, em không chỉ nắm bắt được các khái niệm cơ bản của đồ họa 3D, mà còn phát triển các kỹ năng trong việc sử dụng các thư viện đồ họa, tối ưu hóa mã nguồn, và xử lý các hiệu ứng hình ảnh. Đêm Noel là một chủ đề đầy tính sáng tạo và cảm xúc, và việc tái hiện không gian này bằng công nghệ 3D sẽ mang lại cho người xem những trải nghiệm thú vị, sống động.

# MỤC LỤC

[Lời Mở Đầu 2](#_Toc186387008)

[MỤC LỤC 3](#_Toc186387009)

[CHƯƠNG I: MÔ TẢ BÀI TOÁN 4](#_Toc186387010)

[**1.1** **Lý do chọn đề tài** 4](#_Toc186387011)

[**1.2** **Ý nghĩa khi chọn đề tài** 5](#_Toc186387012)

[**1.3** **Cơ sở lý thuyết** 6](#_Toc186387013)

[**1.3.1 Đồ họa máy tính** 6](#_Toc186387014)

[**1.3.2 Đồ họa 3D và các khái niệm cơ bản** 6](#_Toc186387015)

[**1.3.3 Open GL Trong Dev C++** 7](#_Toc186387016)

[**1.3.4 Các Phương Pháp Vẽ 3D** 7](#_Toc186387017)

[**1.4** **Phần mềm và công cụ hỗ trợ** 8](#_Toc186387018)

[**1.4.1 Dev C++** 8](#_Toc186387019)

[**1.4.2 Một số vấn đề gặp phải** 8](#_Toc186387020)

[**1.4.3 Thư viện OpenGL và các công cụ hỗ trợ** 9](#_Toc186387021)

[CHƯƠNG II: KẾT QUẢ NGHIÊN CỨU 11](#_Toc186387022)

[**2.1** **Thiết kế tuyết rơi** 11](#_Toc186387023)

[**2.1.1 Tổng quan** 11](#_Toc186387024)

[**2.1.2 Các thành phần chính** 11](#_Toc186387025)

[**2.1.3 Hàm chính** 12](#_Toc186387026)

[**2.1.4 Hiệu suất** 13](#_Toc186387027)

[**2.2** **Thiết kế mặt đất** 13](#_Toc186387028)

[**2.3** **Thiết kế người tuyết** 16](#_Toc186387029)

[**2.4** **Thiết kế cây thông** 18](#_Toc186387030)

[**2.5** **Thiết kế ngôi nhà** 22](#_Toc186387031)

[**2.6** **Thiết kế hộp quà** 25](#_Toc186387032)

[**2.7** **Thiết kế di chuyển** 28](#_Toc186387033)

[**2.8** **Thiết kế âm thanh và ánh sáng** 30](#_Toc186387034)

[**2.8.1 Âm thanh trong ứng dụng** 31](#_Toc186387035)

[**2.8.2 Ánh sáng trong ứng dụng** 31](#_Toc186387036)

[CHƯƠNG III: KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM 35](#_Toc186387037)

[**3.1** **Kiến thức đã học được** 35](#_Toc186387038)

[**3.2** **Bài học kinh nghiệm** 35](#_Toc186387039)

[**3.3** **Kết luận** 36](#_Toc186387040)

[Tài Liệu Kham Khảo 38](#_Toc186387041)

# CHƯƠNG I: MÔ TẢ BÀI TOÁN

* 1. **Lý do chọn đề tài**

Lý do em chọn đề tài "Đêm Noel bằng 3D" là sự kết hợp giữa niềm đam mê với công nghệ đồ họa máy tính và mong muốn tạo ra một không gian 3D sinh động, mang lại cảm giác ấm áp và vui tươi của mùa Giáng sinh. Đêm Noel, với không gian lung linh ánh đèn, cây thông Noel, và không khí vui tươi, luôn là hình ảnh gắn liền với những kỷ niệm đẹp trong cuộc sống. Em mong muốn tái hiện lại không gian này thông qua công nghệ 3D, nhằm tạo ra một sản phẩm không chỉ có giá trị về mặt kỹ thuật mà còn mang tính nghệ thuật, đem đến cho người xem cảm xúc như đang sống trong một đêm Giáng sinh thực sự.

Bên cạnh đó, việc lựa chọn đề tài này cũng xuất phát từ mong muốn áp dụng và phát triển các kiến thức đã học về **đồ họa máy tính** và **lập trình 3D**. Môn học đồ họa máy tính đã cung cấp cho em những kiến thức cơ bản về mô hình hóa 3D, ánh sáng, kết cấu vật liệu, và cách sử dụng các thư viện đồ họa như OpenGL trong môi trường lập trình. Đêm Noel là một chủ đề phù hợp để em thực hành và nâng cao kỹ năng lập trình, đặc biệt là trong việc tạo dựng các cảnh vật, ánh sáng, và các hiệu ứng đồ họa 3D.Ngoài ra, việc thực hiện đề tài này cũng giúp em khám phá và nâng cao khả năng giải quyết vấn đề khi đối mặt với các thử thách trong lập trình 3D, như tối ưu hóa hiệu suất, xử lý ánh sáng và vật liệu, cũng như tạo ra những hiệu ứng thực tế cho cảnh vật. Từ đó, em có thể rèn luyện khả năng tư duy logic, sáng tạo và làm việc với các công cụ đồ họa chuyên nghiệp.

Cuối cùng, đề tài này không chỉ mang tính chất học thuật mà còn là cơ hội để em kết hợp sự sáng tạo và kỹ thuật, tạo ra một sản phẩm mang tính nghệ thuật, giúp người xem cảm nhận được vẻ đẹp của mùa Giáng sinh qua công nghệ đồ họa 3D.

* 1. **Ý nghĩa khi chọn đề tài**

Việc chọn đề tài "Đêm Noel bằng 3D" không chỉ đơn thuần là một dự án học thuật mà còn mang lại rất nhiều ý nghĩa về cả mặt kỹ thuật và tinh thần. Trước hết, việc mô phỏng không gian đêm Noel trong môi trường 3D cho phép chúng ta thể hiện sự sáng tạo và áp dụng kiến thức đồ họa máy tính vào thực tế. Đêm Noel là một chủ đề thân thuộc, giàu giá trị văn hóa và cảm xúc, và chúng ta mong muốn tái hiện lại không khí ấm áp, kỳ diệu của mùa lễ hội qua lăng kính công nghệ, mang đến cho người xem trải nghiệm hình ảnh sống động và đầy cảm hứng.

Ngoài ra, đề tài này có ý nghĩa quan trọng trong việc rèn luyện các kỹ năng về lập trình 3D và đồ họa máy tính, đặc biệt là trong việc sử dụng các công cụ lập trình như Dev C++ và thư viện OpenGL. Đây là cơ hội để chúng ta khám phá và áp dụng các phương pháp mô hình hóa 3D, kỹ thuật ánh sáng, và tạo ra các hiệu ứng thực tế như tuyết rơi hay ánh đèn Giáng sinh. Những kỹ năng này không chỉ có giá trị trong học tập mà còn mở rộng cơ hội nghề nghiệp trong các lĩnh vực đồ họa máy tính, game, hoặc thiết kế mô phỏng.

Hơn nữa, đề tài này còn giúp phát triển tư duy sáng tạo, khả năng giải quyết vấn đề và cải thiện khả năng làm việc với các công cụ đồ họa chuyên nghiệp. Khi làm việc với đồ họa 3D, chúng ta không chỉ học cách làm việc với mã nguồn mà còn phải hiểu và áp dụng các nguyên lý nghệ thuật như bố cục, màu sắc và ánh sáng, từ đó nâng cao khả năng cảm thụ và tạo ra những tác phẩm có giá trị thẩm mỹ cao.

Cuối cùng, chọn đề tài này còn là cách để chúng ta thể hiện niềm đam mê với công nghệ và nghệ thuật. Đêm Noel không chỉ là một chủ đề dễ thương mà còn là biểu tượng của niềm vui, sự sum vầy và ấm áp. Qua đó, chúng ta mong muốn gửi gắm thông điệp về tình yêu thương và sự kết nối trong cuộc sống, đồng thời làm cho công nghệ đồ họa máy tính trở nên gần gũi và dễ tiếp cận hơn với mọi người.

* 1. **Cơ sở lý thuyết**

**1.3.1 Đồ họa máy tính**

Đồ họa máy tính là một lĩnh vực nghiên cứu trong ngành khoa học máy tính, liên quan đến việc tạo ra, xử lý và trình bày các hình ảnh và đồ họa trên máy tính. Đồ họa máy tính bao gồm nhiều thể loại như đồ họa 2D, đồ họa 3D, mô phỏng, và hình ảnh động. Các ứng dụng của đồ họa máy tính rất đa dạng, bao gồm thiết kế đồ họa, phát triển game, mô phỏng vật lý, và hình ảnh trong các phần mềm khoa học kỹ thuật. Để tạo ra đồ họa, máy tính sử dụng các công cụ phần mềm và phần cứng như card đồ họa (GPU) và các API như OpenGL hay DirectX.

**1.3.2 Đồ họa 3D và các khái niệm cơ bản**

Đồ họa 3D là quá trình tạo ra các đối tượng ba chiều trên màn hình máy tính, nơi các vật thể được mô phỏng trong không gian 3D với chiều sâu, chiều rộng và chiều cao. Các khái niệm cơ bản trong đồ họa 3D bao gồm:

* **Mô Hình 3D (3D Models):** Các đối tượng được xây dựng trong không gian ba chiều, có thể là hình học đơn giản như hình lập phương, hình cầu hoặc các mô hình phức tạp hơn như người, xe cộ, hoặc môi trường 3D.
* **Toạ Độ 3D (3D Coordinates):** Trong không gian 3D, mỗi điểm được xác định bằng ba giá trị tọa độ (x, y, z), đại diện cho vị trí của điểm đó trong không gian ba chiều.
* **Ánh Sáng 3D (3D Lighting):** Ánh sáng trong đồ họa 3D ảnh hưởng đến cách mà các vật thể được hiển thị trên màn hình, từ đó tạo ra các hiệu ứng bóng, phản chiếu, và bóng đổ.
* **Cảnh 3D (3D Scene):** Là một tập hợp của các đối tượng 3D, ánh sáng và camera, tất cả phối hợp với nhau để tạo ra hình ảnh cuối cùng.
* **Render:** Quá trình biến mô hình 3D thành hình ảnh 2D có thể nhìn thấy trên màn hình, với các hiệu ứng ánh sáng và bóng đổ.

**1.3.3 Open GL Trong Dev C++**

**OpenGL (Open Graphics Library)** là một API đồ họa mạnh mẽ, đa nền tảng, dùng để phát triển ứng dụng đồ họa 2D và 3D. OpenGL cung cấp một bộ công cụ để vẽ và quản lý các đối tượng đồ họa, xử lý ánh sáng, vật liệu, và các hiệu ứng khác trong không gian 3D. Trong Dev C++, OpenGL có thể được tích hợp vào các dự án thông qua việc cài đặt các thư viện hỗ trợ như GLUT (OpenGL Utility Toolkit) hoặc GLEW (OpenGL Extension Wrangler).

Quá trình sử dụng OpenGL trong Dev C++ gồm các bước cơ bản như sau:

* Cài đặt các thư viện OpenGL.
* Thiết lập các thuộc tính đồ họa (màu sắc, ánh sáng, vật liệu, v.v.).
* Sử dụng các hàm OpenGL để vẽ các đối tượng 3D.
* Xử lý các sự kiện đầu vào (như bàn phím, chuột).
* Render các đối tượng lên màn hình.

**1.3.4 Các Phương Pháp Vẽ 3D**

Có nhiều phương pháp để vẽ đồ họa 3D, bao gồm các kỹ thuật cơ bản và nâng cao. Dưới đây là các phương pháp chính trong vẽ 3D:

* **Phong Shading:** Phương pháp này sử dụng tính toán ánh sáng tại mỗi điểm trên bề mặt của vật thể. Phong Shading tạo ra hiệu ứng ánh sáng mềm mại và làm cho bề mặt vật thể trông tự nhiên hơn.
* **Gouraud Shading:** Đây là phương pháp sử dụng ánh sáng tại các điểm cực trị (vertices) và sau đó nội suy màu sắc giữa các điểm đó để tạo ra hiệu ứng ánh sáng mượt mà. So với Phong Shading, Gouraud Shading thường có hiệu suất tốt hơn nhưng có thể bị mất chi tiết ở các vùng có sự chuyển tiếp ánh sáng mạnh.
* **Ray Tracing:** Ray Tracing là một kỹ thuật vẽ ảnh 3D phức tạp hơn, trong đó các tia ánh sáng được bắn ra từ máy ảnh (camera) và theo dõi đường đi của chúng khi chúng tương tác với các vật thể trong cảnh. Ray Tracing cho phép tạo ra các hiệu ứng như phản chiếu, khúc xạ, và bóng đổ chân thực, nhưng đòi hỏi tài nguyên tính toán lớn.
* **Wireframe Rendering:** Là một phương pháp đơn giản trong vẽ 3D, nơi chỉ có các đường viền của các đối tượng 3D được hiển thị. Mặc dù không có chi tiết về màu sắc hay ánh sáng, nhưng phương pháp này giúp dễ dàng nhìn thấy cấu trúc của mô hình 3D.
* **Polygonal Rendering:** Phương pháp này sử dụng các đa giác (polygon) để tạo thành các mô hình 3D. Các mô hình 3D thường được chia thành các đa giác, và quá trình render sẽ xử lý các đa giác này để tạo ra hình ảnh cuối cùng.
  1. **Phần mềm và công cụ hỗ trợ**

**1.4.1 Dev C++**

Dev C++ là một IDE (Integrated Development Environment) phổ biến cho lập trình C/C++. Trong dự án này, Dev C++ đã được sử dụng để phát triển và biên dịch mã nguồn C++ sử dụng thư viện OpenGL.

Dev C++ cung cấp môi trường phát triển dễ sử dụng với các tính năng như tự động hoàn thành mã nguồn, gỡ lỗi (debugging), và tích hợp với các công cụ biên dịch (compiler) như GCC.

Trong trường hợp này, Dev C++ được sử dụng để xây dựng ứng dụng đồ họa với OpenGL. Bạn có thể thấy trong mã nguồn của mình, việc sử dụng các thư viện như GL/glut.h giúp lập trình viên dễ dàng vẽ và xử lý các đối tượng 3D trong không gian.

**1.4.2 Một số vấn đề gặp phải**

Vấn đề về biên dịch và liên kết thư viện: Một trong những vấn đề phổ biến khi sử dụng Dev C++ với OpenGL là việc cấu hình đúng các thư viện cần thiết. Để mã của bạn có thể biên dịch và chạy đúng, bạn cần đảm bảo rằng các thư viện GL và GLUT đã được thêm vào đúng cách. Nếu không, chương trình có thể không nhận diện được các hàm từ OpenGL và sẽ báo lỗi trong quá trình biên dịch.

Vấn đề về hiển thị đồ họa: Đôi khi, khi chạy chương trình OpenGL, các đối tượng hoặc các hiệu ứng đồ họa không hiển thị đúng như mong đợi. Điều này có thể liên quan đến việc cấu hình lại cửa sổ hiển thị, ánh sáng, hoặc camera không được thiết lập chính xác. Trong trường hợp mã bạn cung cấp, việc điều chỉnh các tham số như góc nhìn của camera và các cài đặt ánh sáng có thể gây ra những thay đổi trong việc hiển thị đối tượng.

Vấn đề với hiệu suất: Khi số lượng tuyết rơi trong cảnh quá lớn (ví dụ như 15,000 bông tuyết trong mã của bạn), việc vẽ chúng có thể gây ra sự giảm hiệu suất đáng kể. Điều này có thể làm chương trình chạy chậm, nhất là khi sử dụng phần cứng có cấu hình thấp.

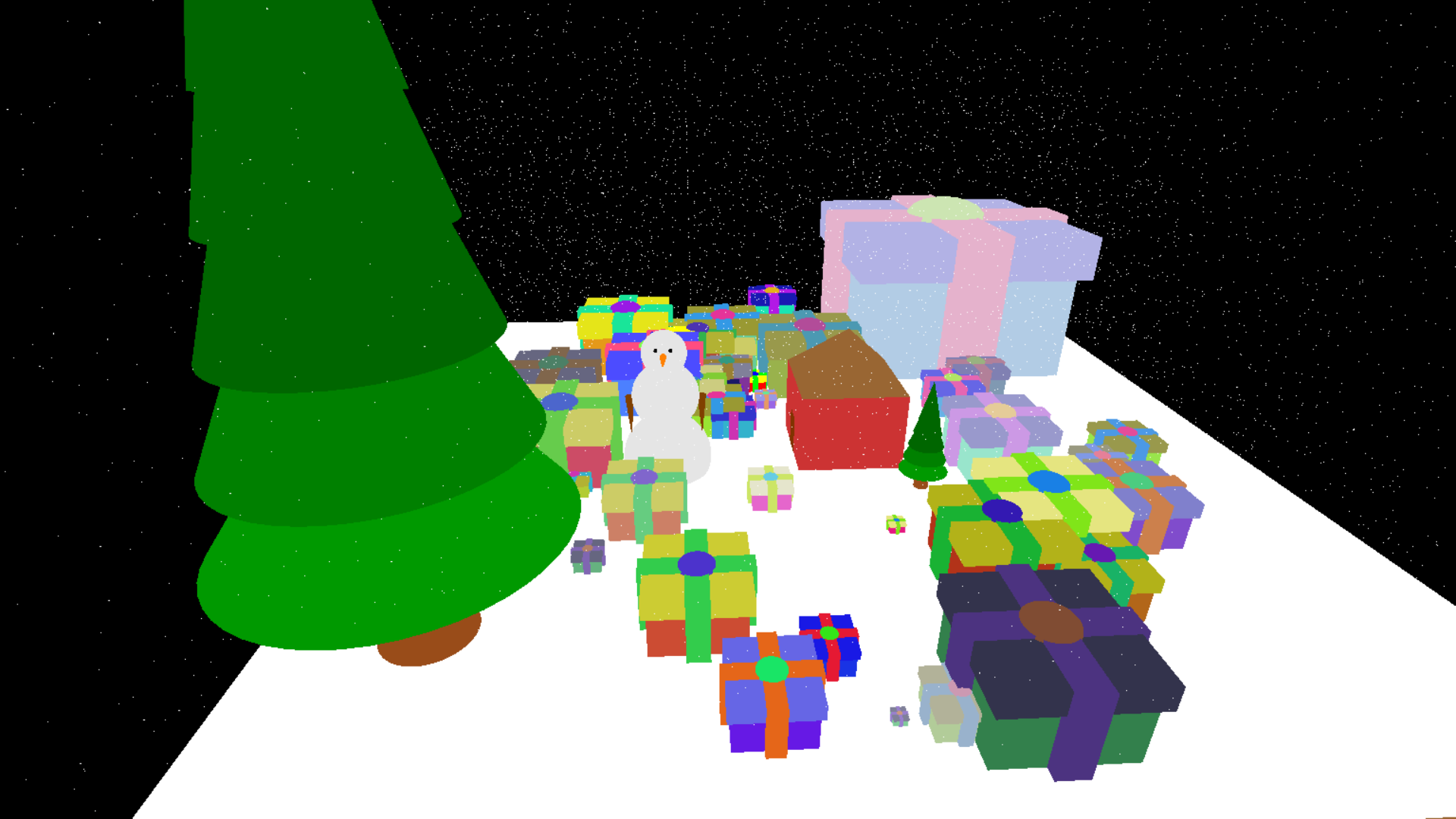
Cài đặt thiếu thư viện: Thư viện GLUT đôi khi không có sẵn trong các phiên bản Dev C++ và cần được tải về và cấu hình thêm. Điều này có thể khiến cho việc biên dịch mã gặp phải lỗi không tìm thấy các hàm liên quan đến GLUT.

**1.4.3 Thư viện OpenGL và các công cụ hỗ trợ**

**Thư viện OpenGL:** Trong mã của bạn, thư viện OpenGL (#include <GL/glut.h>) được sử dụng để hỗ trợ việc vẽ đồ họa 3D. OpenGL cung cấp các hàm và công cụ mạnh mẽ cho việc tạo hình ảnh 3D, ánh sáng, vật liệu, và các phép biến đổi không gian. Các hàm OpenGL như glBegin, glEnd, glTranslatef, glRotatef được sử dụng để vẽ các đối tượng trong không gian 3D.

**Thư viện GLUT (OpenGL Utility Toolkit):** GLUT là một thư viện tiện ích cho OpenGL giúp xử lý các cửa sổ đồ họa, nhập liệu từ bàn phím, chuột, và các sự kiện khác. Trong mã của bạn, các hàm như glutInit, glutDisplayFunc, và glutMainLoop được sử dụng để khởi tạo cửa sổ hiển thị đồ họa, cập nhật màn hình, và điều khiển vòng lặp chính của ứng dụng.

# CHƯƠNG II: KẾT QUẢ NGHIÊN CỨU

****

Hình : Mô phỏng khung cảnh đêm noel

* 1. **Thiết kế tuyết rơi**

Mục Tiêu: Phần làm tuyết rơi được thiết kế nhằm mô phỏng hiệu ứng tuyết rơi chân thực trong môi trường 3D. Hiệu ứng này góp phần tăng tính chân thực và thẩm mỹ cho cảnh quan trong chương trình.

**2.1.1 Tổng quan**

Hiệu ứng tuyết rơi được xây dựng bằng cách:

* Tạo các hạt tuyết (snowflakes) với vị trí, kích thước và tốc độ rơi ngẫu nhiên.
* Liên tục cập nhật vị trí của từng hạt tuyết để mô phỏng chuyển động rơi.
* Vẽ các hạt tuyết ở vị trí mới trong mỗi khung hình.

**2.1.2 Các thành phần chính**

1. Dữ liệu hạt tuyết (struct Snowflake):

* x, y, z: Vị trí 3D của hạt tuyết.
* size: Kích thước của hạt tuyết, tạo cảm giác thực tế về khoảng cách.
* speed: Tốc độ rơi của từng hạt, thay đổi ngẫu nhiên để tăng tính đa dạng.

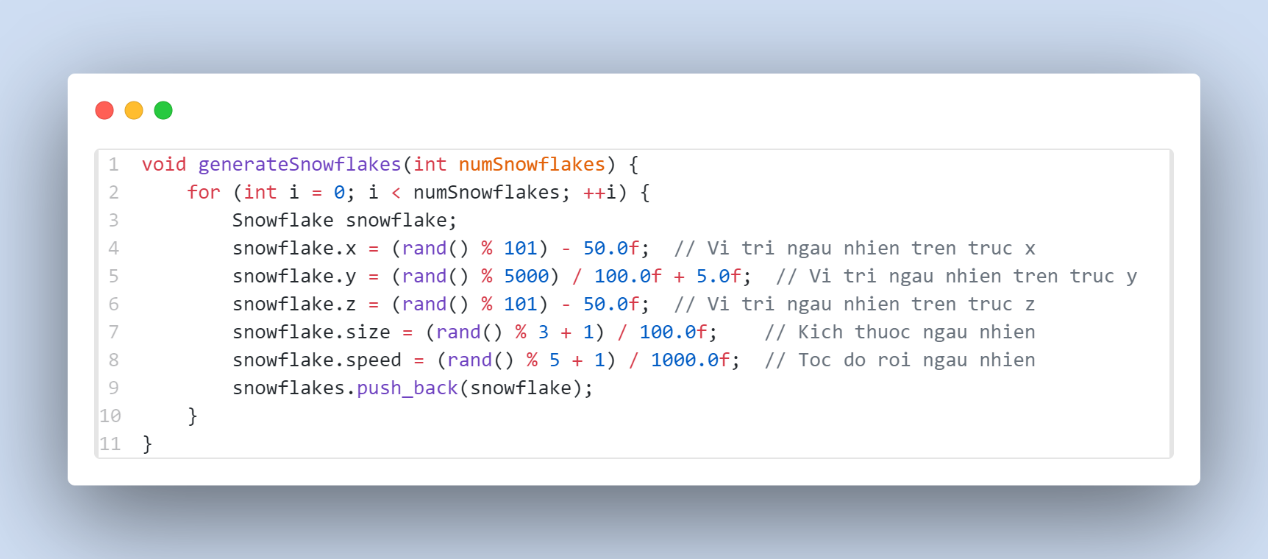
1. Danh sách hạt tuyết (std::vector<Snowflake>)

* Quản lý tập hợp các hạt tuyết hiện có trong cảnh.

**2.1.3 Hàm chính**

1. Tạo hạt tuyết (generateSnowflakes)

* Sinh ngẫu nhiên vị trí, kích thước và tốc độ rơi cho các hạt tuyết.
* Số lượng hạt tuyết được xác định tùy vào yêu cầu, ví dụ: 15,000 hạt.



1. Cập nhật vị trí hạt tuyết (updateSnowflakes)

* Giảm giá trị y (chiều cao) theo tốc độ rơi của từng hạt.
* Khi hạt tuyết rơi xuống dưới một mức cố định, nó được tái sinh ở vị trí mới phía trên.



c. Vẽ hạt tuyết (drawSnowflakes)

* Các hạt tuyết được vẽ bằng các đường thẳng nhỏ để giảm tải đồ họa và mô phỏng hiệu ứng rơi.



**2.1.4 Hiệu suất**

* Việc tạo và cập nhật 15,000 hạt tuyết có thể ảnh hưởng đến hiệu suất nếu không được tối ưu hóa.
* Tuy nhiên, chương trình sử dụng OpenGL để tận dụng GPU, giúp đạt hiệu suất tốt.
  1. **Thiết kế mặt đất**

**Mô tả chức năng**: Hàm drawLand được thiết kế để tạo và vẽ mặt đất (landscape) trong một không gian 3D. Ngoài mặt đất chính, hàm còn vẽ các mặt xung quanh để tạo sự hoàn chỉnh cho cảnh quan.

Chi tiết mã nguồn:



Phân tích hoạt động

* Vẽ các bề mặt:
  + Hàm sử dụng glBegin(GL\_QUADS) để bắt đầu vẽ các hình chữ nhật (bề mặt) trong không gian 3D.
  + Gồm 6 bề mặt: mặt trên, mặt dưới, mặt trước, mặt sau, mặt trái và mặt phải. Mỗi bề mặt được định nghĩa bởi 4 đỉnh thông qua các lệnh glVertex3f(x, y, z).
* Tạo màu sắc:
  + Mỗi bề mặt được tô màu bằng glColor3f(r, g, b), sử dụng hệ màu RGB.
  + Ví dụ:
    - Mặt trên: Màu trắng (1.0, 1.0, 1.0).
    - Mặt dưới: Màu xám (0.447, 0.447, 0.447).
    - Các mặt khác: Màu nâu (0.6, 0.4, 0.2).
* Tọa độ các đỉnh:
  + Các đỉnh được xác định để tạo thành một hình hộp chữ nhật bao phủ mặt đất và không gian bên dưới.
  + Ví dụ:
    - Mặt trên: Bắt đầu từ tọa độ (-50, 0, -50) đến (50, 0, 50).
    - Mặt dưới: Bắt đầu từ (-50, -5, -50) đến (50, -5, 50).
* Kết thúc lệnh vẽ:
  + Sử dụng glEnd() để kết thúc khối lệnh vẽ.

Ưu điểm:

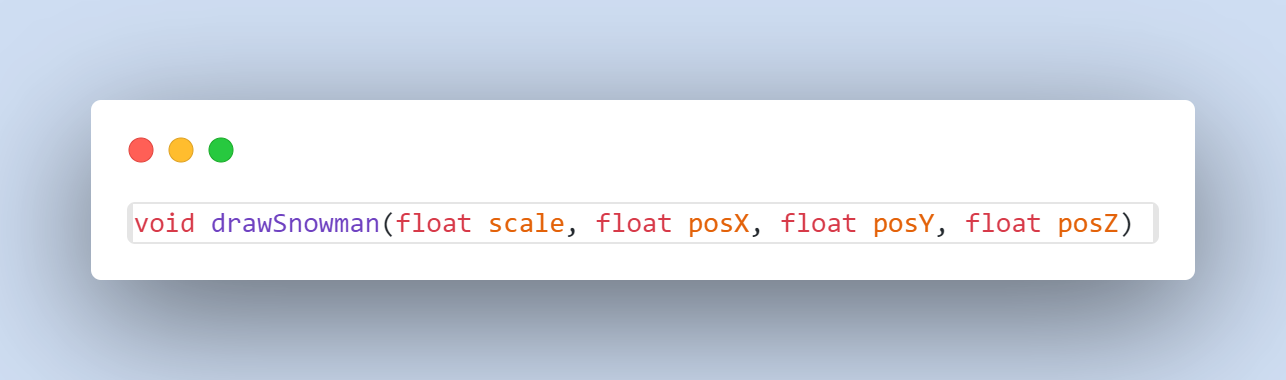
* Dễ dàng tùy chỉnh kích thước và màu sắc của từng bề mặt.
* Cách tổ chức mã rõ ràng, dễ hiểu và mở rộng.

Nhược điểm:

* Mã cứng nhắc (hard-code) trong việc định nghĩa tọa độ các đỉnh và màu sắc, gây khó khăn khi thay đổi kích thước hoặc vị trí.
* Không sử dụng các tính năng tiên tiến của OpenGL như VAO/VBO để tối ưu hóa hiệu suất.
  1. **Thiết kế người tuyết**

**Mô tả chức năng:** Hàm drawSnowman được thiết kế để vẽ một người tuyết có thể tùy chỉnh kích thước và vị trí. Người tuyết gồm ba phần chính: thân, đầu, tay và các chi tiết như mắt, mũi. Các thông số đầu vào cho phép điều chỉnh tỷ lệ và tọa độ của người tuyết trong không gian 3D.

**Định nghĩa hàm:**

****

* **scale**: Xác định tỷ lệ kích thước tổng thể của người tuyết.
* **posX, posY, posZ**: Xác định tọa độ trung tâm của thân dưới người tuyết.

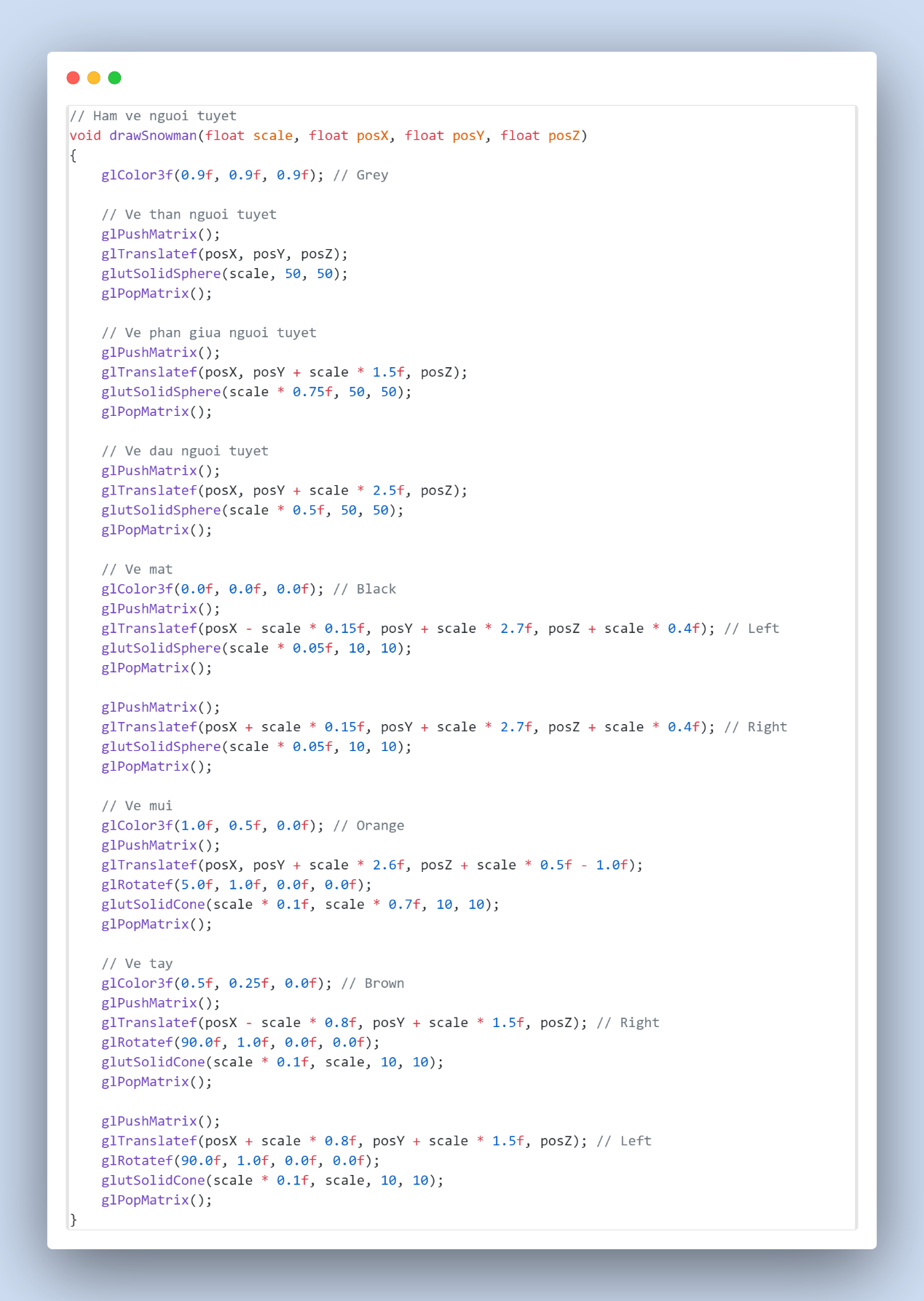
**Chi tiết hoạt động:**

* Vẽ thân người tuyết:
  + Thân người tuyết gồm ba khối cầu: thân dưới, thân giữa, và đầu.
  + Các khối cầu được vẽ bằng hàm glutSolidSphere với bán kính tỉ lệ theo scale.
* Vẽ mắt:
  + Hai mắt là hai khối cầu nhỏ màu đen (glColor3f(0.0f, 0.0f, 0.0f)).
  + Mắt được đặt lệch theo trục X và hơi nhô ra theo trục Z.
* Vẽ mũi:
  + Mũi là một hình nón màu cam (glColor3f(1.0f, 0.5f, 0.0f)), được vẽ bằng hàm glutSolidCone.
  + Vị trí mũi nhô về phía trước (trục Z) và nằm chính giữa khuôn mặt.
* Vẽ tay:
  + Tay là hai hình nón màu nâu (glColor3f(0.5f, 0.25f, 0.0f)).
  + Tay được đặt hai bên thân giữa, với góc nghiêng 90 độ theo trục X để hướng ra ngoài.

**Đặc điểm nổi bật:**

* **Tính động:** Hàm hỗ trợ thay đổi tỷ lệ và vị trí của người tuyết, giúp linh hoạt trong việc bố trí nhiều người tuyết trong cảnh.
* **Chi tiết trực quan:** Các chi tiết như mắt, mũi, và tay tạo nên hình dáng sinh động cho người tuyết.

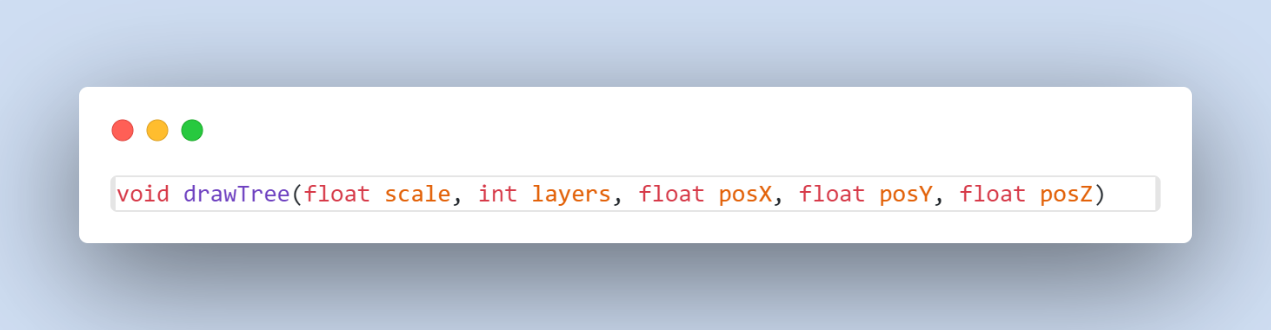
**Mã nguồn:**



* 1. **Thiết kế cây thông**

Hàm drawTree được thiết kế để vẽ một cây thông với thân cây, các tầng lá, và một ngôi sao trên đỉnh (nếu kích thước đủ lớn). Hàm hỗ trợ tùy chỉnh kích thước tổng thể, số tầng lá, và vị trí cây thông trong không gian 3D.

**Định nghĩa hàm:**

****

* scale: Tỷ lệ kích thước tổng thể của cây thông.
* layers: Số lượng tầng lá của cây thông.
* posX, posY, posZ: Tọa độ gốc (đế thân cây) trong không gian 3D.

**Chi tiết hoạt động:**

* Vẽ thân cây:
  + Thân cây được vẽ bằng hình trụ (gluCylinder) với bán kính thay đổi từ gốc đến ngọn.
  + Sử dụng hàm gluNewQuadric để tạo một đối tượng hình học.
  + Màu thân cây được thiết lập là màu nâu (glColor3f(0.6f, 0.3f, 0.1f)).

Các bước:

* + Di chuyển đến vị trí gốc thân cây bằng glTranslatef.
  + Quay hình trụ 90 độ để dựng đứng trên trục Y bằng glRotatef.
  + Xóa đối tượng hình học sau khi vẽ xong để giải phóng bộ nhớ.
* Vẽ lá cây:
  + Lá cây được vẽ bằng các hình nón (glutSolidCone) xếp chồng lên nhau, với bán kính và chiều cao giảm dần theo từng tầng.
  + Màu sắc lá cây là màu xanh lá cây, đậm dần ở các tầng trên.

Các bước:

* + Dịch chuyển lên vị trí phía trên thân cây.
  + Duyệt qua các tầng từ 1 đến layers:
    - Xác định chiều cao và tỷ lệ của tầng lá bằng cách giảm dần theo số tầng.
    - Thay đổi màu sắc lá cây theo tầng.
    - Vẽ hình nón cho từng tầng.
* Vẽ ngôi sao trên đỉnh (nếu scale >= 5):
  + Ngôi sao được vẽ bằng GL\_TRIANGLE\_FAN, sử dụng 10 điểm để tạo hình dạng sao 5 cánh.
  + Màu ngôi sao là màu vàng (glColor3f(1.0f, 1.0f, 0.0f)).

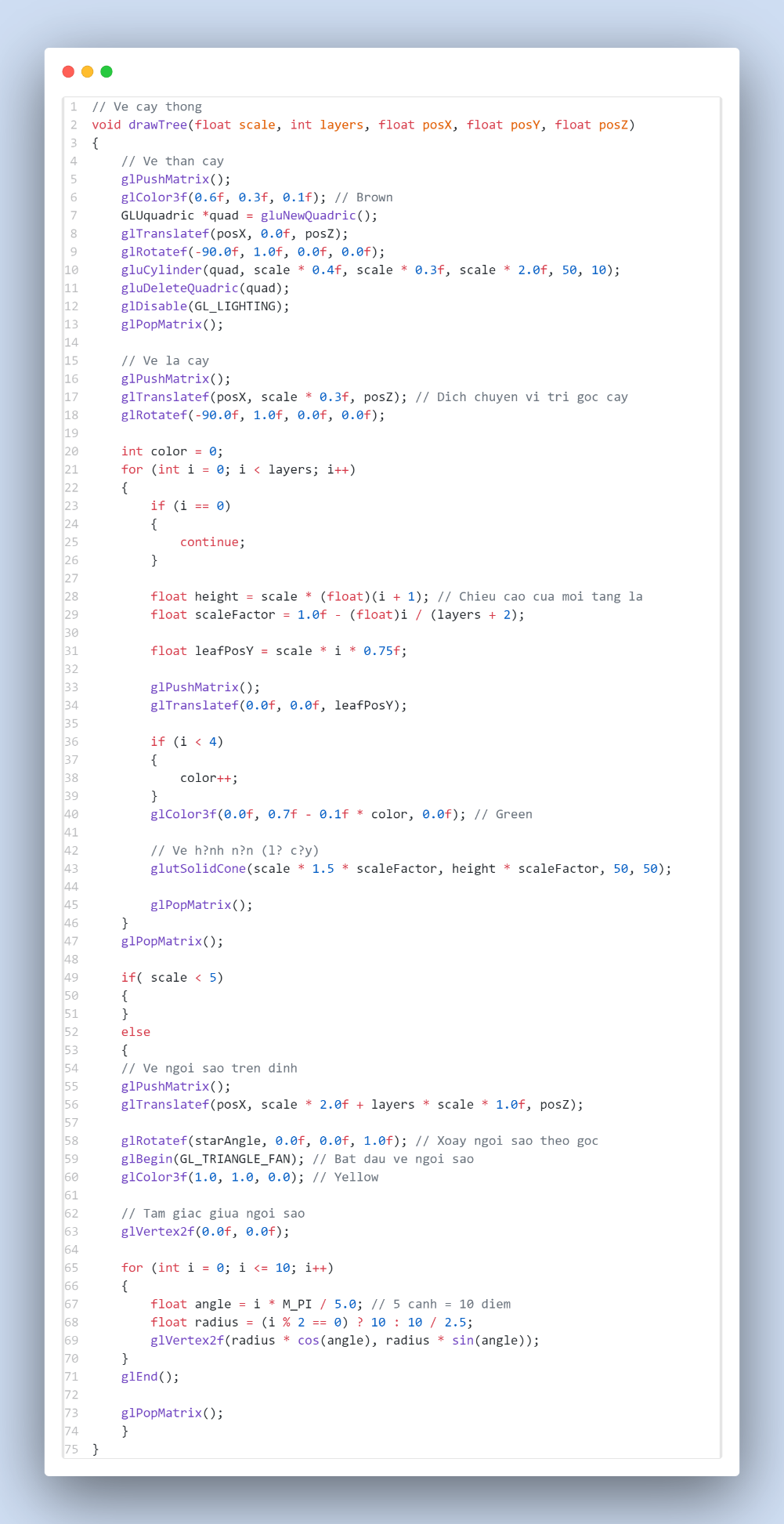
Các bước:

* + Dịch chuyển lên vị trí đỉnh cây.
  + Xoay ngôi sao theo góc starAngle.
  + Sử dụng công thức hình học để tính tọa độ các điểm của ngôi sao.

**Đặc điểm nổi bật:**

* Tính linh hoạt: Hàm cho phép điều chỉnh tỷ lệ (scale) và số tầng lá (layers), phù hợp để tạo nhiều cây thông với hình dạng khác nhau trong một cảnh.
* Hiệu ứng trực quan: Ngôi sao trên đỉnh xoay quanh trục tạo hiệu ứng động.
* Quản lý bộ nhớ: Sử dụng và xóa đúng cách đối tượng GLUquadric để tránh rò rỉ bộ nhớ.

**Mã nguồn:**



* 1. **Thiết kế ngôi nhà**

**Mô tả chức năng:** Hàm drawHouse được thiết kế để vẽ một ngôi nhà đơn giản với các thành phần sau:

* Thân nhà: Là một hình hộp chữ nhật (lập phương).
* Mái nhà: Là một hình chóp đặt lên trên thân nhà.
* Cửa: Là một hình chữ nhật đặt trên một mặt của thân nhà.

**Định nghĩa hàm:**

**void drawHouse();**

* Hàm không nhận tham số đầu vào và không trả về giá trị.
* Ngôi nhà được vẽ với vị trí cố định, tỷ lệ kích thước được xác định bởi các phép biến đổi bên trong hàm.

**Chi tiết hoạt động:**

* Dịch chuyển và biến đổi:
  + glPushMatrix và glPopMatrix: Lưu và khôi phục trạng thái ma trận biến đổi, đảm bảo các phép biến đổi chỉ áp dụng cục bộ cho ngôi nhà.
  + glTranslatef: Dịch chuyển toàn bộ ngôi nhà đến tọa độ (3.0, 0.0, 0.0).
  + glScalef: Phóng to ngôi nhà theo tỷ lệ 3 lần trên cả ba trục.
* Thân nhà:
  + Hình dạng: Thân nhà là một hình hộp chữ nhật được tạo bởi 6 mặt (toàn bộ là các GL\_QUADS).
  + Màu sắc: Màu đỏ được thiết lập bằng glColor3f(0.8f, 0.2f, 0.2f).
  + Chi tiết mặt:
    - Mặt trước, mặt sau, mặt trái, mặt phải, mặt trên, mặt dưới được vẽ riêng biệt bằng các đỉnh tọa độ 3D.
  + Tọa độ các mặt:
    - Các đỉnh tọa độ được thiết lập sao cho tạo thành hình hộp có kích thước 4x3x4 (dài x rộng x cao).
* Mái nhà:
  + Hình dạng: Mái nhà là một hình chóp với đáy là hình chữ nhật và đỉnh tại (0.0, 5.0, 0.0).
  + Màu sắc: Màu nâu được thiết lập bằng glColor3f(0.6f, 0.4f, 0.2f).
  + Chi tiết mặt:
    - Mái được tạo bởi 4 mặt tam giác (GL\_TRIANGLES) kết nối đỉnh và các cạnh của hình hộp.
* Cửa:
  + Hình dạng: Cửa là một hình chữ nhật nằm trên mặt trước của thân nhà.
  + Màu sắc: Màu nâu đậm hơn thân nhà, thiết lập bằng glColor3f(0.5f, 0.2f, 0.0f).
  + Chi tiết mặt:
    - Cửa được vẽ bằng một GL\_QUADS, đặt ở tọa độ tương đối trên mặt trước của thân nhà.
    - Tọa độ cửa: Chiều rộng 1.0, chiều cao 1.5.

**Mã nguồn đầy đủ:**

****

* 1. **Thiết kế hộp quà**

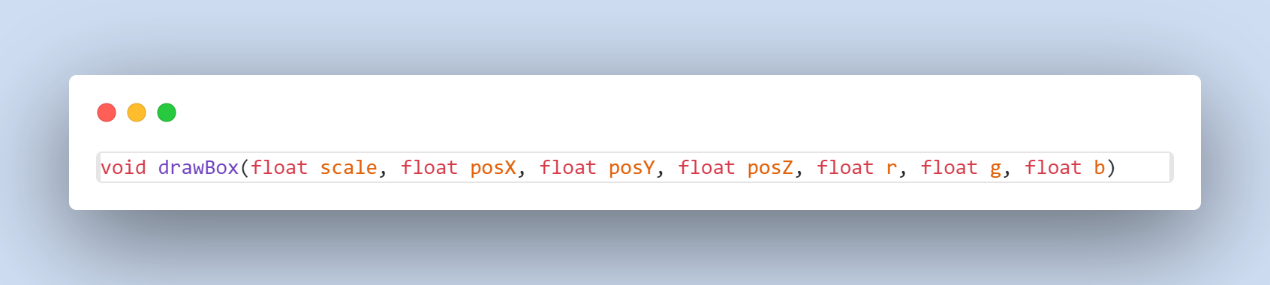
Hàm drawBox được thiết kế để vẽ một hộp quà với các thành phần:

* Hộp chính: Là một hình lập phương.
* Nắp hộp: Là một hình lập phương phẳng đặt trên hộp chính.
* Dải ruy băng: Gồm hai dải ngang được tạo từ hình hộp chữ nhật.
* Nơ trên nắp hộp: Là một hình cầu tượng trưng.

Hàm cho phép tùy chỉnh:

* Tỷ lệ hộp quà (scale).
* Vị trí (posX, posY, posZ).
* Màu sắc thông qua các tham số màu đỏ (r), xanh lá (g), và xanh dương (b).

**Định nghĩa hàm:**



**Tham số đầu vào:**

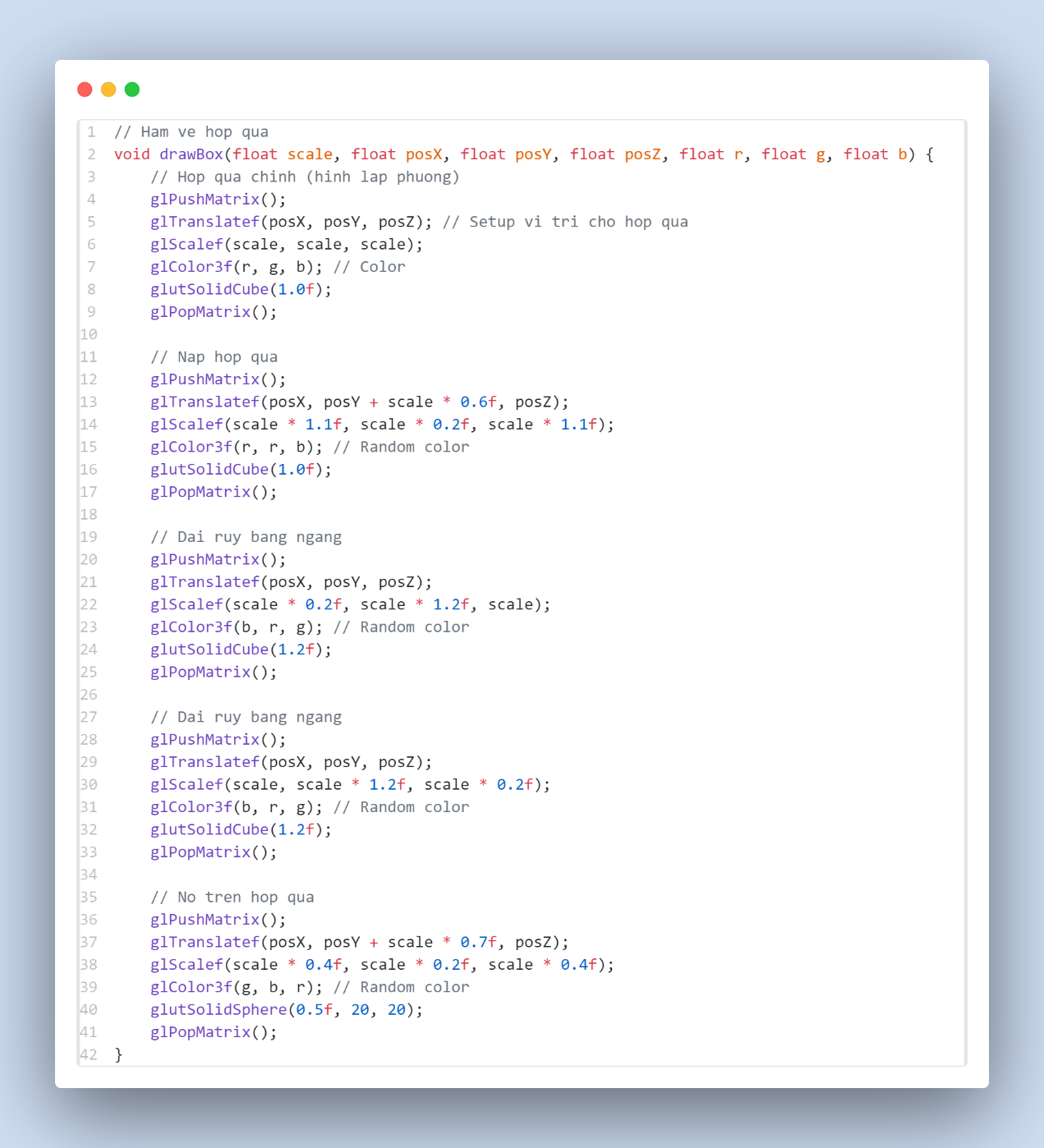
* scale: Tỷ lệ kích thước của hộp quà.
* posX, posY, posZ: Tọa độ vị trí của hộp quà trong không gian 3D.
* r, g, b: Màu sắc của hộp quà (giá trị từ 0.0 đến 1.0 cho từng kênh màu RGB).

**Hàm không trả về giá trị.**

**Chi tiết hoạt động**

* Hộp chính:
  + Hình dạng: Hộp quà chính là một hình lập phương (glutSolidCube).
  + Kích thước: Được điều chỉnh bởi scale.
  + Vị trí: Được đặt tại tọa độ (posX, posY, posZ).
  + Màu sắc: Được thiết lập từ tham số đầu vào r, g, b.
  + Lệnh chính: **glutSolidCube(1.0f);**
* Nắp hộp:
  + Hình dạng: Nắp hộp là một hình lập phương mỏng.
  + Kích thước:
    - Chiều rộng và dài lớn hơn hộp chính 10% (scale \* 1.1f).
    - Chiều cao nhỏ hơn hộp chính (scale \* 0.2f).
  + Vị trí: Được đặt ở phía trên hộp chính, tại tọa độ (posX, posY + scale \* 0.6f, posZ).
  + Màu sắc: Kết hợp các tham số màu đầu vào để tạo màu nắp khác biệt.
  + Lệnh chính: **glutSolidCube(1.0f);**
* Dải ruy băng ngang (hai phần):
  + Hình dạng: Hai dải ruy băng được tạo bởi hình lập phương kéo dài thành hình hộp chữ nhật.
  + Chi tiết:
    - Dải 1: Chạy dọc theo trục Y và được ép dẹp theo trục X.
    - glScalef(scale \* 0.2f, scale \* 1.2f, scale);
    - Dải 2: Chạy dọc theo trục Z và được ép dẹp theo trục X.
    - glScalef(scale, scale \* 1.2f, scale \* 0.2f);
  + Vị trí: Cả hai dải đều nằm ở giữa hộp quà (posX, posY, posZ).
  + Màu sắc: Sử dụng tổ hợp b, r, g để tạo sự nổi bật.
  + Lệnh chính: **glutSolidCube(1.2f);**
* Nơ trên nắp hộp:
  + Hình dạng: Nơ được vẽ dưới dạng một hình cầu (glutSolidSphere).
  + Kích thước:
    - Bán kính: 0.5f (tùy chỉnh theo scale).
  + Vị trí: Đặt ở trên nắp hộp, tại tọa độ (posX, posY + scale \* 0.7f, posZ).
  + Màu sắc: Sử dụng tổ hợp g, b, r.
  + Lệnh chính: **glutSolidSphere(0.5f, 20, 20);**

**Mã nguồn đầy đủ:**

****

* 1. **Thiết kế di chuyển**

Hàm di chuyển trong ứng dụng OpenGL này cho phép người dùng điều khiển camera trong không gian 3D. Người dùng có thể thay đổi vị trí camera bằng cách sử dụng các phím trên bàn phím (W, A, S, D, Space, C), xoay góc nhìn thông qua di chuyển chuột và điều chỉnh các góc của camera theo hướng mong muốn.

Các thành phần trong mã nguồn di chuyển

* Thông tin camera và tốc độ di chuyển:
  + camX, camY, camZ: Các biến lưu trữ vị trí của camera trong không gian 3D. Ban đầu camera được đặt tại vị trí (0.0, 2.0, 10.0).
  + camYaw, camPitch: Các góc quay của camera quanh các trục X và Y. camYaw điều chỉnh góc xoay quanh trục Y (quay trái/phải), trong khi camPitch điều chỉnh góc lên/xuống (quay lên/xuống).
  + velocity: Tốc độ di chuyển của camera, được đặt là 0.03 đơn vị mỗi lần cập nhật.
  + rotationSpeed: Tốc độ xoay của camera khi di chuyển chuột, với giá trị là 0.005.
* Các trạng thái bàn phím và chuột:
  + keyStates: Một std::map dùng để theo dõi trạng thái của các phím thông thường (chẳng hạn như W, A, S, D, Space, C).
  + specialKeyStates: Một std::map theo dõi trạng thái các phím đặc biệt (chẳng hạn như phím mũi tên).
  + isLeftMousePressed: Cờ xác định liệu chuột trái có đang được nhấn hay không.
  + lastMouseX, lastMouseY: Vị trí của chuột khi người dùng bắt đầu nhấn chuột trái. Dùng để tính toán thay đổi góc xoay của camera.

Các hàm xử lý di chuyển

* handleKeyboardDown:
  + Hàm này được gọi khi một phím thông thường (như W, A, S, D) được nhấn. Trạng thái của phím này được lưu trữ trong keyStates để cập nhật di chuyển của camera trong vòng lặp update.
* handleKeyboardUp:
  + Khi một phím được thả ra, trạng thái của nó sẽ được đặt thành false trong keyStates.
* handleSpecialKeyboardDown và handleSpecialKeyboardUp:
  + Các hàm này dùng để theo dõi trạng thái của các phím đặc biệt, chẳng hạn như các phím mũi tên.
* update:
  + Trong hàm update, các điều kiện được kiểm tra để di chuyển camera theo các phím được nhấn. Các phím di chuyển như W, S, A, D thay đổi vị trí của camera, và các phím Space, C điều chỉnh chiều cao của camera.
* display:
  + Trong hàm display, camera được di chuyển và điều chỉnh theo các thay đổi trong camX, camY, camZ bằng cách sử dụng gluLookAt. Đây là hàm cài đặt góc nhìn của camera trong không gian 3D, với các tham số như vị trí camera và mục tiêu mà camera nhìn vào.
* mouse và motion:
  + mouse xử lý sự kiện nhấn hoặc thả chuột trái, xác định xem chuột có được nhấn hay không.
  + motion xử lý sự kiện di chuyển chuột. Khi chuột trái được nhấn, góc quay của camera (camYaw và camPitch) sẽ thay đổi dựa trên sự di chuyển của chuột, giúp người dùng xoay góc nhìn của camera.

Các phím điều khiển và hành động:

* W: Di chuyển camera tiến về phía trước (theo hướng camYaw).
* S: Di chuyển camera lùi lại (ngược với hướng camYaw).
* A: Di chuyển camera sang trái (theo hướng vuông góc với camYaw).
* D: Di chuyển camera sang phải.
* Space: Tăng chiều cao của camera (di chuyển lên).
* C: Giảm chiều cao của camera (di chuyển xuống).

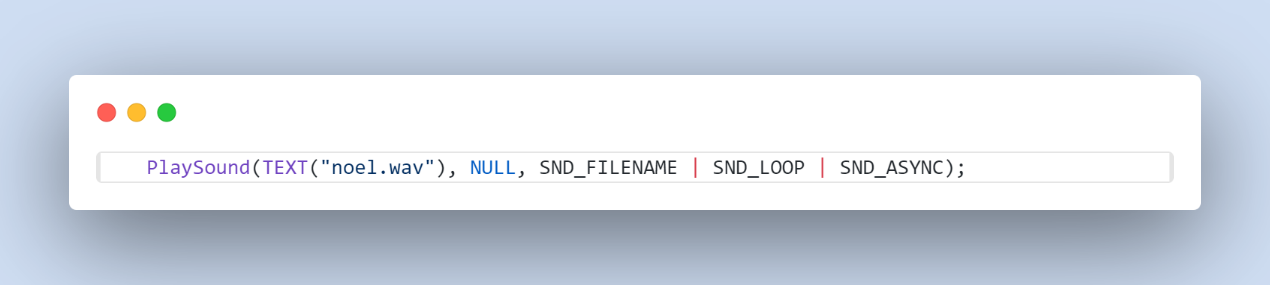
Chi tiết hoạt động của hệ thống:

* Di chuyển camera: Tốc độ di chuyển của camera được điều khiển bởi các phím W, A, S, D. Di chuyển trên mặt phẳng XY sẽ làm thay đổi vị trí camX và camZ (hướng trước/sau và trái/phải). Di chuyển dọc trục Y sẽ thay đổi camY, cho phép người dùng điều chỉnh vị trí lên hoặc xuống.
* Xoay camera: Việc thay đổi hướng nhìn của camera được thực hiện khi người dùng di chuyển chuột khi giữ nút chuột trái. Mỗi lần di chuyển chuột, camYaw và camPitch được thay đổi, cho phép người dùng xoay camera quanh các trục X và Y.
* Góc nhìn camera: Hàm gluLookAt được sử dụng để cập nhật góc nhìn của camera. Vị trí của camera được xác định bởi camX, camY, camZ, trong khi hướng mà camera nhìn vào được tính toán từ các góc camYaw và camPitch.
  1. **Thiết kế âm thanh và ánh sáng**

Trong ứng dụng đồ họa này, âm thanh và ánh sáng đóng vai trò quan trọng để tạo ra không gian ảo sống động. Bằng cách sử dụng OpenGL và các thư viện bổ trợ, chúng ta có thể mô phỏng một môi trường ba chiều với ánh sáng và âm thanh. Cụ thể, ứng dụng này mô phỏng một cảnh Noel với các yếu tố điều khiển chuyển động camera, ánh sáng, và âm thanh nền.

**2.8.1 Âm thanh trong ứng dụng**

Âm thanh trong ứng dụng được thêm vào để tạo ra một không gian sống động hơn. Bằng cách sử dụng thư viện mmsystem.h của Windows, chúng ta có thể phát âm thanh nền trong ứng dụng. Cụ thể, trong chương trình này, âm thanh "noel.wav" được phát khi ứng dụng bắt đầu:



* SND\_FILENAME: Chỉ định rằng tệp âm thanh được tải từ tệp tin có đường dẫn.
* SND\_LOOP: Phát âm thanh liên tục (lặp lại).
* SND\_ASYNC: Phát âm thanh không đồng bộ, tức là âm thanh sẽ phát song song với các thao tác khác trong chương trình.

Điều này giúp người dùng cảm nhận được không khí Noel khi di chuyển trong không gian 3D.

**2.8.2 Ánh sáng trong ứng dụng**

Ánh sáng trong đồ họa 3D đóng vai trò rất quan trọng trong việc tạo ra các hiệu ứng hình ảnh, giúp các đối tượng trong không gian 3D trở nên sống động và có chiều sâu. Trong OpenGL, việc cấu hình ánh sáng giúp mô phỏng các loại ánh sáng khác nhau và cách chúng tương tác với các vật thể trong không gian. Dưới đây là giải thích chi tiết về các thành phần ánh sáng trong mã nguồn.

Trong OpenGL, ánh sáng được định nghĩa thông qua các thuộc tính của nguồn sáng và các thuộc tính của vật liệu. Trong chương trình, ánh sáng được thiết lập với nguồn sáng GL\_LIGHT0 và có các thành phần chính sau:

1. Ánh sáng môi trường (Ambient Light)

Ánh sáng môi trường giúp làm sáng tổng thể không gian 3D bằng cách chiếu đều ánh sáng vào tất cả các bề mặt, bất kể vị trí của chúng. Đây là loại ánh sáng không tạo ra bóng, chỉ giúp cân bằng và tránh cho các vật thể bị tối hoàn toàn.

* Cấu hình: **GLfloat ambientLight[] = {0.5f, 0.5f, 0.5f, 1.0f};**
  + Các giá trị này xác định cường độ ánh sáng môi trường cho các màu đỏ, xanh lá, xanh dương và alpha (độ trong suốt). Trong trường hợp này, tất cả các giá trị màu đều là 0.5, nghĩa là ánh sáng môi trường có cường độ vừa phải cho tất cả các kênh màu.
  + Giá trị alpha là 1.0f, tức là ánh sáng không trong suốt và chiếu sáng hoàn toàn.

1. Ánh sáng khuếch tán (Diffuse Light)

Ánh sáng khuếch tán tạo ra ánh sáng chiếu vào các bề mặt vật thể và làm sáng các bề mặt đó. Ánh sáng khuếch tán phụ thuộc vào góc giữa bề mặt và hướng chiếu sáng. Các bề mặt trực tiếp đối diện với nguồn sáng sẽ nhận được ánh sáng mạnh hơn, trong khi các bề mặt nằm ngoài góc chiếu sáng sẽ nhận ít ánh sáng hơn.

* Cấu hình: **GLfloat diffuseLight[] = {1.0f, 1.0f, 1.0f, 1.0f};**
  + Các giá trị này cho biết ánh sáng khuếch tán có màu trắng (RGB = 1.0f, 1.0f, 1.0f) và cường độ tối đa (alpha = 1.0f). Ánh sáng này giúp làm sáng các vật thể trong không gian 3D và tạo ra các bóng đổ và chi tiết.

1. Ánh sáng phản xạ (Specular Light)

Ánh sáng phản xạ tạo ra các điểm sáng trên bề mặt vật thể khi ánh sáng phản chiếu từ bề mặt này. Nó giúp tăng thêm chi tiết và làm nổi bật các vùng có độ bóng cao (chẳng hạn như kim loại, thủy tinh, v.v.). Ánh sáng phản xạ mạnh ở các điểm có góc chiếu sáng nhỏ, tạo ra các điểm sáng lấp lánh.

* Cấu hình: **GLfloat specularLight[] = {0.5f, 0.5f, 0.5f, 1.0f};**
  + Các giá trị này cho thấy ánh sáng phản xạ có cường độ vừa phải (0.5f cho mỗi kênh màu) và có màu trắng. Việc cấu hình ánh sáng phản xạ giúp tạo ra các điểm sáng trên bề mặt các vật thể có tính phản chiếu.

1. Vị trí của nguồn sáng (Light Position)

Vị trí của nguồn sáng quyết định nơi ánh sáng xuất phát và cách ánh sáng chiếu vào các vật thể trong không gian 3D. Trong trường hợp này, ánh sáng được đặt tại một điểm cố định trong không gian ba chiều.

* Cấu hình: **GLfloat lightPosition[] = {0.0f, 0.0f, 40.0f, 1.0f};**
  + Vị trí ánh sáng được đặt tại (0.0f, 0.0f, 40.0f), nghĩa là nó nằm trên trục Z với độ cao 40 đơn vị. Các giá trị cuối cùng (1.0f) xác định rằng nguồn sáng là một nguồn ánh sáng điểm (có vị trí cố định).

Khi sử dụng ánh sáng điểm (glLightfv(GL\_LIGHT0, GL\_POSITION, lightPosition);), ánh sáng sẽ chiếu đều từ vị trí này vào các vật thể.

1. Cấu hình vật liệu

Để ánh sáng có thể ảnh hưởng đến các vật thể, các vật thể trong OpenGL cần phải có các thuộc tính vật liệu. Mã của bạn cấu hình các đặc tính của vật liệu như sau:

* Cấu hình vật liệu:
* GLfloat mat\_specular[] = {0.5f, 0.5f, 0.5f, 1.0f};
* GLfloat mat\_shininess[] = {25.0f};
  + mat\_specular[] xác định độ sáng phản xạ của vật liệu (sáng trắng).
  + mat\_shininess[] xác định độ bóng của vật liệu. Giá trị càng lớn, vật liệu càng bóng và ánh sáng phản chiếu sẽ rõ ràng hơn.

1. Kích hoạt ánh sáng trong OpenGL

Sau khi cấu hình ánh sáng, bạn cần phải bật các tính năng ánh sáng của OpenGL để ánh sáng có hiệu lực. Cụ thể, bạn cần bật GL\_LIGHTING và nguồn sáng GL\_LIGHT0:

**glEnable(GL\_LIGHTING);**

**glEnable(GL\_LIGHT0);**

Điều này sẽ kích hoạt hệ thống ánh sáng trong OpenGL và cho phép ánh sáng chiếu vào các vật thể trong cảnh.

# CHƯƠNG III: KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM

* 1. **Kiến thức đã học được**

Trong quá trình thực hiện dự án môn đồ hoạ máy tính, chúng em đã học được những kiến thức và kỹ năng sau:

* Kiến thức về đồ hoạ máy tính: Chúng em đã học được những khái niệm cơ bản về đồ hoạ máy tính, như hệ tọa độ, đối tượng, thuộc tính, phép biến đổi, phép chiếu, phép ánh xạ, phép tô màu, phép đổ bóng, phép hiển thị, v.v .
* Kỹ năng thiết kế 3d: Chúng em đã học được cách thiết kế một ngôi nhà 3d, bao gồm cách vẽ hình khối, cách tạo chi tiết, cách tạo kết cấu, cách tạo môi trường.
* Kỹ năng lập trình: Chúng em đã học được cách sử dụng ngôn ngữ lập trình C++ và thư viện OpenGL để viết chương trình đồ hoạ máy tính, bao gồm cách khởi tạo màn hình đồ hoạ, cách vẽ các đối tượng, cách thực hiện các phép biến đổi, cách tạo các hiệu ứng, cách hiển thị kết quả.
  1. **Bài học kinh nghiệm**

Trong quá trình thực hiện dự án đêm noel 3d môn đồ hoạ máy tính, Chúng em đã có những bài học kinh nghiệm sau:

* Thành công: Chúng em đã hoàn thành dự án theo đúng yêu cầu của giáo viên, tạo ra một khung cảnh đẹp mắt, sinh động, và ấn tượng. Chúng em đã nhận được những phản hồi tích cực từ giáo viên và Chúng em bè. Chúng em đã cải thiện được nhiều kỹ năng về đồ hoạ máy tính, thiết kế 3d, lập trình, sử dụng phần mềm...
* Thất bại: Chúng em đã gặp một số khó khăn trong quá trình thực hiện dự án, như thiếu ý tưởng, thiếu kiến thức, thiếu kinh nghiệm, thiếu thời gian, gặp lỗi phần mềm, gặp lỗi chương trình... Chúng em đã không thể làm được một số yêu cầu của dự án, như tạo ra những hiệu ứng phức tạp, tạo ra những chi tiết tinh xảo, tạo ra những môi trường đa dạng...
* Khó khăn: Chúng em đã phải đối mặt với một số khó khăn trong quá trình thực hiện dự án, như thiếu tài liệu tham khảo, thiếu nguồn học tập, thiếu người hướng dẫn, thiếu thiết bị, thiếu phần mềm... Chúng em đã phải tự tìm kiếm, tự học hỏi, tự giải quyết những vấn đề mà Chúng em gặp phải.
* Vấn đề: Chúng em đã phải giải quyết một số vấn đề trong quá trình thực hiện dự án, như vấn đề về hệ tọa độ, vấn đề về phép biến đổi, vấn đề về phép chiếu, vấn đề về phép ánh xạ, vấn đề về phép tô màu, vấn đề về phép đổ bóng, vấn đề về phép phản xạ, vấn đề về phép chiếu sáng, vấn đề về phép hiển thị...
* Giải pháp: Chúng em đã tìm ra và áp dụng một số giải pháp để giải quyết những vấn đề mà Chúng em gặp phải, như sử dụng các công thức toán học, sử dụng các thuật toán đồ hoạ, sử dụng các hàm có sẵn trong thư viện glut, sử dụng các công cụ trong phần mềm thiết kế đồ hoạ, sử dụng các nguồn tài liệu tham khảo, sử dụng các nguồn học tập trực tuyến...
* Cải tiến: Chúng em đã cải tiến và nâng cao chất lượng của dự án của chúng em, bằng cách thêm vào những chi tiết mới, thay đổi những kết cấu cũ, tạo ra những hiệu ứng mới, tạo ra những môi trường mới, tối ưu hóa chương trình, tăng tốc độ hiển thị.
  1. **Kết luận**

Qua quá trình thực hiện dự án đêm noel môn đồ hoạ máy tính, Chúng em đã đạt được những mục tiêu sau:

* Tạo ra một ngôi nhà 3d, những cây thông, hộp quà, người tuyết,… và đặc biệt là khung cảnh tuyết rơi với những chi tiết, kết cấu, hiệu ứng, và môi trường đẹp mắt, sinh động, và ấn tượng.
* Học được những kiến thức và kỹ năng về đồ hoạ máy tính, thiết kế 3d, lập trình, sử dụng phần mềm...
* Rút ra được những bài học kinh nghiệm quý báu từ những thành công, thất bại, khó khăn, vấn đề, giải pháp, cải tiến, đề xuất...

Qua quá trình thực hiện dự án, chúng em đã có những đánh giá sau:

* Điểm mạnh: Chúng em đã sáng tạo được một dự án đồ hoạ độc đáo, hấp dẫn, và thú vị. Chúng em đã cải thiện được nhiều kỹ năng về đồ hoạ máy tính, thiết kế 3d, lập trình, sử dụng phần mềm...
* Điểm yếu: Chúng em đã gặp một số khó khăn trong quá trình thực hiện dự án, như thiếu ý tưởng, thiếu kiến thức, thiếu kinh nghiệm, thiếu thời gian, gặp lỗi phần mềm, gặp lỗi chương trình... Chúng em đã không thể làm được một số yêu cầu của dự án, như tạo ra những hiệu ứng phức tạp, tạo ra những chi tiết tinh xảo, tạo ra những môi trường đa dạng...

# Tài Liệu Kham Khảo

**OpenGL Documentation**: OpenGL là một thư viện đồ họa 3D được sử dụng rộng rãi trong các ứng dụng đồ họa và mô phỏng. Chương trình của bạn sử dụng OpenGL để vẽ và mô phỏng các đối tượng 3D (bông tuyết) trong không gian 3D.  
Tài liệu chính thức:

**+OpenGL Programming Guide**

**+Trang web chính thức của OpenGL:** [**https://www.opengl.org/**](https://www.opengl.org/)

Dev C++ IDE: Dev C++ là môi trường phát triển tích hợp (IDE) cho C++ được sử dụng trong dự án này. IDE này hỗ trợ việc biên dịch, chạy và kiểm tra chương trình dễ dàng. Nó sử dụng MinGW (Minimalist GNU for Windows) làm trình biên dịch.  
Tài liệu và thông tin tham khảo về Dev C++:

+**Dev C++ User Manual**: http://www.bloodshed.net/devcpp.html

+**MinGW Documentation**: <https://www.mingw.org/>

GLUT Library: GLUT (OpenGL Utility Toolkit) là một thư viện giúp quản lý cửa sổ, xử lý sự kiện bàn phím và chuột cho các ứng dụng OpenGL. Chương trình của bạn sử dụng GLUT để tạo và điều khiển cửa sổ OpenGL cho việc hiển thị hình ảnh tuyết rơi.  
Tài liệu tham khảo về GLUT:

+**GLUT DCM**: https://www.opengl.org/resources/libraries/glut/

C++ Programming Language

C++ là ngôn ngữ lập trình chính được sử dụng trong dự án này. Mã nguồn được viết hoàn toàn bằng C++, sử dụng cú pháp cơ bản để khai báo và xử lý các đối tượng bông tuyết và các thuật toán chuyển động trong không gian 3D.  
Tài liệu tham khảo C++:

+**C++ Programming Language** (Bjarne Stroustrup):

+**C++ Documentation**: https://en.cppreference.com/w/

OpenGL and Graphics Programming

**OpenGL and Graphics Programming**  
Chương trình này mô phỏng cảnh tuyết rơi với các đối tượng 3D trong không gian sử dụng OpenGL. Hiểu biết về cách sử dụng OpenGL để vẽ hình học và làm việc với các phép toán 3D rất quan trọng đối với dự án này.

**+OpenGL 4.5 Programming Guide (Dave Shreiner, Graham Sellers, John Kessenich, Bill Licea-Kane): Cung cấp các khái niệm cơ bản về OpenGL và các cách thức vẽ đối tượng trong không gian 3D.**

**+Real-Time Rendering (Tomas Akenine-Möller, Eric Haines): Cung cấp các kỹ thuật và phương pháp rendering trong đồ họa máy tính.**

Basic Physics in Animation

**Basic Physics in Animation**  
Để mô phỏng hiệu ứng tuyết rơi, một số kiến thức cơ bản về vật lý chuyển động trong không gian như trọng lực và tốc độ là cần thiết. Mặc dù chương trình hiện tại chưa sử dụng các yếu tố vật lý phức tạp, nhưng các cải tiến có thể được thực hiện trong tương lai.

**+Mathematics for 3D Game Programming and Computer Graphics (Eric Lengyel): Bao gồm các thuật toán toán học và vật lý dùng trong đồ họa 3D.**

3D Simulation and Animation

+**3D Computer Graphics: A Mathematical Introduction with OpenGL** (Samuel R. Buss): Cung cấp các phương pháp toán học và thuật toán được sử dụng trong đồ họa 3D

+**Computer Graphics: Principles and Practice** (James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes): Là một trong những tài liệu tham khảo nổi tiếng về đồ họa máy tính, bao gồm cả việc sử dụng OpenGL và các phương pháp đồ họa 3D.

Graphics Programming with C++ and OpenGL

+**Beginning OpenGL Game Programming** (William R. Hunt, J. P. Lewis): Cung cấp các kỹ thuật cơ bản đến nâng cao trong việc lập trình game và ứng dụng đồ họa 3D với C++ và OpenGL.

Mathematics for Computer Graphics

Việc tính toán các tọa độ và chuyển động trong không gian 3D đòi hỏi một số kiến thức cơ bản về đại số tuyến tính và hình học. Các tài liệu này sẽ giúp hiểu rõ hơn về các phép toán được sử dụng trong mã nguồn của bạn.

+**3D Math Primer for Graphics and Game Development** ( Fletcher Dunn, Ian Parberry): Hướng dẫn về toán học 3D cho đồ họa máy tính và lập trình game.