TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: KỸ THUẬT ĐỒ HỌA MÁY TÍNH
MÃ ĐỀ THI: 4
THIẾT KẾ RẠP CHIẾU PHIM
LỚP TÍN CHỈ: KTĐHMT.03.K13.09.LT.C04.1_LT

Giảng viên hướng dẫn: ThS. Lê Mai Nam

Danh sách sinh viên thực hiện: Nhóm 4

TT	Mã sinh viên	Sinh viên thực hiện	Lớp hành chính
1	20222999	Nguyễn Trung Chính	DCCNTT 13.10.16
2	20223155	Nguyễn Trí Dũng	DCCNTT 13.10.16
3	20222996	Trần Văn Nam	DCCNTT 13.10.16
4	20222998	Vũ Văn Phong	DCCNTT 13.10.16
5	20223011	Hoàng Ngọc Thành	DCCNTT 13.10.16

Bắc Ninh – 2025

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

LỜI MỞ ĐẦU

Trong thời đại công nghệ số hiện nay, kỹ thuật đồ họa máy tính đóng vai trò vô cùng quan trọng trong nhiều lĩnh vực, đặc biệt là trong thiết kế và mô phỏng không gian thực tế. Việc ứng dụng đồ họa máy tính giúp con người trực quan hóa các ý tưởng, mô hình và cải thiện trải nghiệm người dùng trong các sản phẩm phần mềm.

Bài tập lớn học phần "Kỹ thuật đồ họa máy tính" với đề tài "Thiết kế rạp chiếu phim" là cơ hội để sinh viên vận dụng kiến thức đã học vào thực tế. Thông qua bài tập này, nhóm sinh viên không chỉ rèn luyện kỹ năng lập trình đồ họa mà còn phát triển tư duy thiết kế không gian 3D, kết hợp giữa kỹ thuật và thẩm mỹ. Một số mục tiện cần đặt được trong bài tập lớn lần này đó là:

Vận dụng kiến thức về kỹ thuật đồ họa máy tính trong việc xây dựng mô hình 3D cho một không gian thực tế.

Làm quen với quy trình thiết kế và hiện thực hóa một mô hình đồ họa từ khâu phân tích yêu cầu, xây dựng bố cục cho đến lập trình và trình bày sản phẩm.

Nâng cao kỹ năng sử dụng các thư viện đồ họa cơ bản (như OpenGL hoặc các công cụ hỗ trợ khác) để dựng hình, ánh sáng và chuyển động.

Rèn luyện tư duy thẩm mỹ và khả năng làm việc nhóm hiệu quả trong môi trường học thuật.

Bài tập lớn "Thiết kế rạp chiếu phim" không chỉ giúp nhóm sinh viên củng cố và áp dụng các kiến thức về kỹ thuật đồ họa máy tính mà còn tạo cơ hội để phát triển tư duy thiết kế, khả năng làm việc nhóm và xử lý các vấn đề thực tiễn. Qua quá trình thực hiện, nhóm đã hiểu rõ hơn về cách dựng hình, xử lý ánh sáng, tổ chức không gian 3D cũng như những thách thức khi hiện thực hóa một mô hình đồ họa phức tạp.

Mặc dù vẫn còn một số hạn chế về mặt kỹ thuật và thẩm mỹ, nhưng bài làm thể hiện sự cố gắng và tinh thần học hỏi nghiêm túc của toàn bộ nhóm sinh viên trong suốt quá trình thực hiện.

Chương 1: Cơ sở lý thuyết về kỹ thuật đồ họa máy tính

1.1. Các khái niệm cơ bản về đồ họa máy tính

1.1.1. Khái niệm đồ họa máy tính

Đồ họa máy tính (tiếng Anh: computer graphics) là một lĩnh vực của khoa học máy tính nghiên cứu về cơ sở 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,...) [1].

Hình 1. 1: Minh họa kỹ thuật đồ họa máy tính

Theo nghĩa rộng hơn, đồ họa máy tính là phương pháp và công nghệ dùng trong việc chuyển đổi qua lại giữa dữ liệu và hình ảnh trên màn hình bằng máy tính. Đồ họa máy tính hay kĩ thuật đồ họa máy tính còn được hiểu dưới dạng phương pháp và kĩ thuật tạo hình ảnh từ các mô hình toán học mô tả các đối tượng hay dữ liệu lấy được từ các đối tượng trong thực tế [1].

Trong bối cảnh công nghệ ngày càng phát triển, đồ họa máy tính đã trở thành một phần không thể thiếu trong nhiều lĩnh vực của đời sống và sản xuất. Từ các ngành truyền thống như kiến trúc, thiết kế công nghiệp, truyền thông, đến các lĩnh vực hiện đại như trò chơi điện tử, thực tế ảo (VR), điện ảnh 3D hay trí tuệ nhân tạo (AI), đồ họa máy tính đều giữ vai trò trung tâm trong việc xây dựng trải nghiệm thị giác sinh động và hấp dẫn. Nhờ khả năng mô phỏng và trực quan hóa thông tin phức tạp, đồ họa máy tính giúp người dùng hiểu và tương tác với dữ liệu một cách hiệu quả, qua đó nâng cao chất lượng sản phẩm và trải nghiệm người dùng.

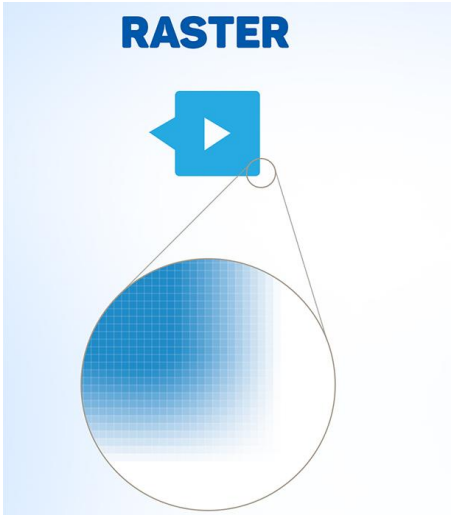
Không chỉ là công cụ kỹ thuật, đồ họa máy tính còn được xem là cầu nối giữa nghệ thuật và công nghệ. Việc tạo ra một hình ảnh hay không gian đồ họa đẹp mắt không chỉ yêu cầu kiến thức về toán học và lập trình, mà còn đòi hỏi tư duy thẩm mỹ, sự sáng tạo và khả năng cảm thụ nghệ thuật. Chính sự kết hợp này đã tạo nên sức mạnh đặc biệt của đồ họa máy tính — nơi mà kỹ sư và nghệ sĩ cùng hợp tác để tạo ra những sản phẩm vừa chuẩn xác về mặt kỹ thuật, vừa cuốn hút và giàu cảm xúc về mặt thị giác.

Bên cạnh đó nó không chỉ là việc tạo ra hình ảnh mà còn là truyền tải thông tin, kể chuyện và khơi gợi cảm xúc thông qua hình ảnh. Đồ họa máy tính kết hợp các nguyên tắc nghệ thuật, toán học và khoa học máy tính để tạo ra nội dung trực quan phong phú [2]. Sự kết hợp giữa hài hòa tư duy thẩm mỹ và sáng tạo cùng với kỹ thuật máy tính để tạo ra những sản phẩm trực quan hấp dẫn dễ tiếp cận đến người dùng một cách trực quan nhất và đơn giản giữa nhưng thông điệp được gửi gắm vào đó [3].

1.1.2. Các kỹ thuật đồ họa máy tính

Có rất nhiều kỹ thuật hiển thị hình ảnh được áp dụng trong ngành đồ họa máy tính, mỗi kỹ thuật lại có tuổi đời và những ưu, nhược điểm khác nhau. Dưới đây là 3 kỹ thuật hiển thị hình ảnh của đồ họa máy tính phổ biến nhất đó chính là raster, vector, đồ họa 3D. Sau đây chúng em sẽ giải thích gắn ngọn 3 kỹ thuật này:

Đồ họa raster (đồ họa hoặc hình ảnh Bitmap) là một trong những kỹ thuật hiển thị hình ảnh lâu đời và phổ biến nhất với nền tảng kỹ thuật lấy từ công nghệ màn hình tivi đã tồn tại rất lâu trước khi máy tính điện tử ra đời. Với kỹ thuật này, tất cả các hình ảnh đều được làm nên từ các ô vuông có màu nhỏ li ti được gọi là pixel (phần tử ảnh).



Hình 1. 2: Minh họa độ phân giải của đồ họa raster

Tùy thuộc vào độ phân giải, một hình ảnh có thể chứa hàng nghìn đến hàng triệu pixel, giống như một bức tường được xây lên từ nhiều viên gạch vậy. Ưu điểm lớn nhất của kỹ thuật raster là các hình ảnh raster có thể hiển thị các chi tiết rõ ràng, đẹp với màu sắc đa dạng, hài hòa. Tuy nhiên, hình ảnh raster có thể sẽ bị "vỡ" hoặc mờ nếu phóng to hoặc bị nén quá nhiều. Dung lượng các file ảnh raster cũng khá lớn nếu có độ phân giải cao [1].

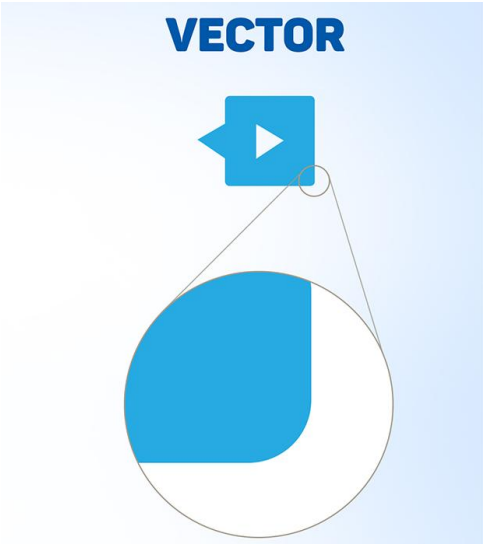
Tuy nhiên, kỹ thuật raster cũng tồn tại nhược điểm rõ rệt. Do hình ảnh được tạo ra từ pixel, nên khi phóng to quá mức (hoặc nén sai cách), các điểm ảnh sẽ lộ rõ, khiến hình ảnh bị vỡ, nhòe, và mất đi độ sắc nét ban đầu. Đồng thời, các file ảnh raster có độ phân giải cao thường có dung lượng lớn, tốn nhiều tài nguyên bộ nhớ và băng thông khi truyền tải hoặc xử lý.

Các định dạng phổ biến của đồ họa raster bao gồm: JPEG, PNG, BMP, GIF, TIFF,… Trong thực tế, đồ họa raster được sử dụng rộng rãi trên web, ứng dụng di động, truyền hình, thiết kế giao diện, trò chơi, và chỉnh sửa ảnh kỹ thuật số.

Đồ họa vector là kỹ thuật tạo dựng hình ảnh bằng các đường kẻ quy định bởi các công thức toán học lần đầu tiên được sử dụng cho màn hình máy tính trong những năm 60 và 70 của thế kỉ 20. Tuy không phổ biến bằng kỹ thuật đồ họa raster và đã từng có một thời gian gần như bị "xóa sổ" bởi raster, đồ họa vector đang được ưa chuộng trở lại.

Bên cạnh đó đồ họa này cũng được gọi là kỹ thuật biểu diễn hình ảnh dựa trên các công thức toán học và tọa độ hình học thay vì các điểm ảnh (pixel) như đồ họa raster. Một hình ảnh vector được mô tả bằng các đối tượng hình học như đường thẳng, đường cong (Bezier), đa giác, hình tròn, và văn bản. Mỗi đối tượng này được xác định bằng tập hợp các thuộc tính như màu sắc, vị trí, độ dày đường viền và kiểu tô màu.

Khác với raster, hình ảnh vector không phụ thuộc vào độ phân giải, do đó chúng có thể được phóng to hoặc thu nhỏ vô hạn mà vẫn giữ nguyên độ sắc nét và chi tiết. Đây là ưu điểm nổi bật của đồ họa vector, đặc biệt phù hợp với các ứng dụng như: thiết kế logo, biểu tượng, sơ đồ kỹ thuật, bản vẽ kỹ thuật, minh họa sách giáo khoa, hoặc bất kỳ trường hợp nào đòi hỏi sự linh hoạt trong kích thước và độ nét.



Hình 1. 3: Minh họa độ phân giải của đồ họa vector

Nhờ vào những ưu điểm như các hình ảnh vector đơn giản, dễ dàng phóng to mà không bị giảm chất lượng, có dung lượng nhỏ hơn so với raster, dễ chỉnh sửa, rất thích hợp với việc thiết kế các loại đồ họa ít màu sắc, đơn giản như logo, icon hay biểu tượng [1].

Ngoài ra, vì không chứa dữ liệu từng điểm ảnh, các file đồ họa vector thường có dung lượng nhẹ hơn đáng kể so với raster, thuận tiện cho việc lưu trữ và truyền tải. Tuy nhiên, nhược điểm của kỹ thuật này là khó mô phỏng chi tiết phức tạp hoặc các hiệu ứng tự nhiên như ánh sáng, bóng mờ, hoặc chuyển màu mượt – vốn là thế mạnh của đồ họa raster.

Các định dạng phổ biến của đồ họa vector bao gồm: SVG (Scalable Vector Graphics), AI (Adobe Illustrator), EPS, PDF, và CDR (CorelDRAW). Trong thực tế, các phần mềm thiết kế đồ họa chuyên nghiệp như Adobe Illustrator, CorelDRAW, Inkscape chủ yếu sử dụng mô hình vector để tạo các bản vẽ kỹ thuật và sản phẩm in ấn chất lượng cao.

Đồ họa 3D là kỹ thuật đồ họa đang được tập trung phát triển nhất trong thời điểm hiện tại, với sự quan tâm và tiềm năng của các ứng dụng như không gian ảo hay hình chiếu ba chiều. Nhà thiết kế đồ họa phải thực hiện rất nhiều bước khác nhau và áp dụng nhiều kỹ thuật tạo dựng hình ảnh phức tạp để có được một đối tượng hình ảnh 3D đúng nghĩa. Trước hết, khung cơ bản (wire-frame) của vật thể phải được dựng trong một phần mềm đồ họa máy tính, sau đó các phần của vật thể sẽ được thêm vào và nối với nhau (rigged) để tạo sự liên kết chân thực, đặc biệt là với các vật thể có khả năng chuyển động. Sau đó vật thể phải được render [1].

Bên cạnh cái khái niệm cơ bản thì còn phải bổ sung thêm một vài khái niệm để có thể làm rõ các khái niệm của Đồ họa 3D (3D Graphics) là kỹ thuật mô phỏng các đối tượng, không gian và cảnh vật trong môi trường ba chiều (3 chiều: x, y, z) bằng máy tính. Không giống như đồ họa 2D (raster hoặc vector) vốn chỉ hoạt động trong không gian hai chiều (chiều rộng và chiều cao), đồ họa 3D bổ sung thêm chiều sâu (depth) để tạo cảm giác chân thực, có khối và có không gian thực tế hơn cho hình ảnh.

Trong kỹ thuật đồ họa 3D, một đối tượng 3D thường được xây dựng từ các mô hình hình học (3D models) bao gồm các đỉnh (vertices), cạnh (edges) và mặt (faces) kết hợp với nhau để tạo thành dạng khối. Các mô hình này sau đó sẽ được tô màu, bọc vật liệu (material mapping), chiếu sáng (lighting) và xử lý hiệu ứng ánh sáng - bóng tối (shading) để tạo ra kết quả hình ảnh hoàn chỉnh. Cuối cùng, quá trình rendering (kết xuất) sẽ tạo ra một hình ảnh 2D cuối cùng từ mô hình 3D để hiển thị lên màn hình.

Ưu điểm nổi bật của đồ họa 3D là khả năng thể hiện chân thực các đối tượng trong thế giới thực với chiều sâu, ánh sáng, vật liệu, độ bóng, độ nhám,... Đây là công cụ lý tưởng cho các ngành như: phim hoạt hình 3D, trò chơi điện tử, kiến trúc, y tế, kỹ thuật, mô phỏng thực tế ảo (VR) và thực tế tăng cường (AR).

Tuy nhiên, đồ họa 3D đòi hỏi khối lượng tính toán lớn và phần cứng mạnh, do phải xử lý nhiều bước phức tạp như mô hình hóa, ánh xạ kết cấu, tính toán ánh sáng và hiệu ứng vật lý. Ngoài ra, thời gian học và triển khai cũng dài hơn so với 2D vì sự phức tạp trong mô hình và tương tác không gian.

Các phần mềm đồ họa 3D phổ biến bao gồm: Blender, Autodesk Maya, 3ds Max, Cinema 4D, SketchUp, v.v. Định dạng file thường dùng: .obj, .fbx, .stl, .blend,...

Đây là bước mất thời gian và phức tạp nhất vì một vật thể 3D có nhiều bề mặt khác nhau với các chất liệu khác nhau, độ trong suốt hay mờ đục, màu sắc, mức độ bắt sáng hay phản sáng khác nhau. Để tạo được một vật thể 3D thật nhất có thể, nhiều kỹ thuật toán học được áp dụng để tính toán và mô phỏng cách ánh sáng chiếu và phản chiếu vào các loại mặt phẳng khác nhau như ray tracing hay radiosity,...

1.1.3. Các lĩnh vực đồ họa máy tính

- Hỗ trợ thiết kế đồ họa công cụ chủ chốt trong lĩnh vực thiết kế đồ họa hiện đại,

cho phép tạo ra các sản phẩm như poster, banner, bìa sách, giao diện web, UI/UX, hình ảnh minh họa,... với khả năng điều chỉnh màu sắc, bố cục, kiểu chữ và hiệu ứng một cách linh hoạt và chính xác. Các phần mềm như Adobe Photoshop, Illustrator, CorelDRAW,... là ví dụ điển hình cho ứng dụng này.

- Huấn luyện đào tạo ảo (quân sự, hàng không,...) kết hợp thực tế ảo (VR) và mô phỏng thời gian thực cho phép xây dựng các hệ thống huấn luyện mô phỏng chiến trường, buồng lái máy bay, xe tăng hoặc các tình huống khẩn cấp. Điều này giúp học viên rèn luyện kỹ năng trong môi trường an toàn, tiết kiệm chi phí và dễ tái lập các tình huống nguy hiểm khó thực hiện ngoài đời thực.
- Thiết kế thương hiệu đóng vai trò then chốt trong thiết kế logo, bộ nhận diện thương hiệu, bao bì sản phẩm, catalogue,... đảm bảo tính thẩm mỹ, đồng nhất và dễ tái sử dụng trên nhiều nền tảng in ấn và kỹ thuật số. Sự kết hợp giữa thẩm mỹ và kỹ thuật giúp thương hiệu ghi dấu ấn mạnh mẽ trong lòng người dùng.
- Đồ họa trong giải thích và truyền thông (Phim ảnh và hoạt hình; Trò chơi điện tử; Thực tế ảo VR và thực thế tăng cường VR) Phim ảnh và hoạt hình: Đồ họa 2D/3D dùng để tạo hoạt cảnh, kỹ xảo điện ảnh (VFX), mô phỏng nhân vật, môi trường giả tưởng,… giúp truyền tải cảm xúc và nội dung sống động. Trò chơi điện tử: Đồ họa thời gian thực (real-time rendering) tái hiện không gian, nhân vật và chuyển động mượt mà. Nhiều engine như Unity, Unreal Engine cho phép xây dựng thế giới ảo tương tác phong phú. Thực tế ảo (VR) và thực tế tăng cường (AR): Đồ họa kết hợp cảm biến và theo dõi chuyển động để người dùng tương tác với môi trường ảo theo thời gian thực – ứng dụng trong game, du lịch ảo, giáo dục, đào tạo,…
- Thiết kế kỹ thuật và kiến trúc (Thiết kế hỗ trợ bằng máy tính CAD; Mô phỏng và phân tích kỹ thuật) CAD (Computer-Aided Design): Cho phép kỹ sư và kiến trúc sư vẽ mô hình kỹ thuật, sơ đồ lắp ráp 2D và 3D một cách chính xác, dễ dàng sửa đổi và tái sử dụng. Mô phỏng kỹ thuật: Đồ họa dùng để trực quan hóa chuyển động cơ khí, phân tích dòng chảy, mô phỏng kết cấu chịu lực giúp tối ưu thiết kế và tăng độ an toàn.
- Khoa học và ý tế (Trực quản hóa dữ liệu khoa học; Hình ảnh y tế) Trực quan hóa dữ liệu khoa học: Giúp biểu diễn dữ liệu phức tạp (ví dụ: cấu trúc phân tử, trường lực, bản đồ địa chất,…) bằng hình ảnh dễ hiểu. Hình ảnh y tế: Các kỹ thuật như CT scan, MRI, siêu âm được chuyển thành hình ảnh giúp bác sĩ phân tích cấu trúc cơ thể, chẩn đoán bệnh chính xác hơn.
- Giao diện người dùng và tình bày thông tin (Giao diện người dùng; Trình bày dữ liệu Data Visualization) Giao diện người dùng (UI): Đồ họa tạo nên các nút bấm, biểu tượng, biểu mẫu,... thân thiện và dễ sử dụng trong ứng dụng, web, thiết bị di động,... Trình bày dữ liệu (Data Visualization): Các biểu đồ, bản đồ nhiệt, sơ đồ tương tác

giúp người dùng nắm bắt thông tin nhanh chóng và chính xác, phục vụ phân tích dữ liệu lớn (Big Data), kinh doanh thông minh (BI).

- Trí tuệ nhân tạo và học máy (Thị giác máy tính; Tạo nội dung bằng AI) Thị giác

máy tính (Computer Vision): Dùng kỹ thuật đồ họa để phân tích hình ảnh, nhận dạng khuôn mặt, vật thể, hành vi,... ứng dụng trong xe tự hành, giám sát an ninh,... Tạo nội dung bằng AI (Generative AI): Các mô hình AI có thể tạo ảnh, video, hoạt cảnh 3D,... dựa trên mô tả ngôn ngữ tự nhiên. Ví dụ: DALL·E, Sora, Stable Diffusion,...

1.2. Các giải thuật đồ họa cơ bản

Trong đồ họa máy tính, các giải thuật cơ bản đóng vai trò quan trọng trong việc xây dựng và hiển thị các đối tượng hình học. Dưới đây là một số thuật toán tiêu biểu đã được học và áp dụng:

Thuật toán DDA là một trong những phương pháp cơ bản nhất để vẽ đường thẳng trong đồ họa máy tính. Nguyên lý hoạt động của nó dựa trên việc phân tích vi phân số, tính toán sự thay đổi tọa độ theo từng bước nhỏ. Cụ thể, với hai điểm đầu (x1,y1) và cuối (x2,y2), thuật toán đầu tiên tính độ dốc m = (y2-y1)/(x2-x1). Nếu $|m| \leq 1$, x sẽ tăng từng đơn vị và y được tính lại bằng y = y + m. Ngược lại, y tăng từng đơn vị và x được điều chỉnh theo x = x + 1/m. Mặc dù đơn giản và dễ hiểu, DDA có nhược điểm lớn là sử dụng phép toán số thực nên dễ tích lũy sai số làm tròn, dẫn đến các đường thẳng không được mượt mà, đặc biệt khi độ dốc lớn. Ngoài ra, tốc độ xử lý chậm do phải thực hiện nhiều phép tính dấu phẩy động. Tuy vậy, DDA vẫn thường được dùng trong giảng dạy như một ví dụ minh họa cơ bản về thuật toán vẽ đường thẳng, giúp người học dễ dàng nắm bắt nguyên lý trước khi chuyển sang các thuật toán tối ưu hơn như Bresenham. Bên cạnh nhưng công cụ đó, thuật toán DDA (Digital Differential Analyzer) được sử dụng để vẽ đường thẳng giữa hai điểm bằng cách nội suy tọa độ dọc theo trục chính. Ý tưởng chính là tăng tọa độ x hoặc y theo các bước nhỏ dựa trên độ dốc, sử dụng phép toán thực. DDA đơn giản nhưng dễ xảy ra sai số tròn do dùng số thực.

Được phát triển bởi Jack Bresenham vào năm 1965, thuật toán Bresenham là một bước tiến vượt bậc trong lĩnh vực đồ họa máy tính nhờ khả năng vẽ đường thẳng chỉ sử dụng các phép toán số nguyên. Thay vì tính toán số thực như DDA, Bresenham sử dụng một biến quyết định (decision parameter) để xác định khi nào cần tăng tọa độ y khi x tăng. Biến này được cập nhật liên tục dựa trên sai số tích lũy giữa vị trí lý tưởng và vị trí thực tế của pixel. Cụ thể, nếu sai số vượt quá 0.5, y sẽ được tăng lên 1 đơn vị và sai số được điều chỉnh lại. Ưu điểm lớn nhất của Bresenham là tốc độ xử lý

nhanh do loại bỏ hoàn toàn các phép toán dấu phẩy động, đồng thời cho kết quả chính xác với đường thẳng mượt mà. Thuật toán này đặc biệt phù hợp cho các hệ thống phần cứng có tài nguyên hạn chế và được ứng dụng rộng rãi trong các thư viện đồ họa, game, hệ thống CAD. Ngoài ra, nguyên lý của Bresenham còn có thể mở rộng để vẽ đường tròn và các đường cong khác, chứng tỏ tính linh hoạt và hiệu quả của nó. Bên cạnh nhưng khái niệm đó, thuật toán Bresenham cũng dùng để vẽ đường thẳng, nhưng cải tiến hơn DDA khi chỉ sử dụng số nguyên. Thuật toán này quyết định điểm vẽ tiếp theo dựa trên sai số tích lũy, cho phép vẽ nhanh, chính xác, phù hợp với các ứng dụng thời gian thực.

Midpoint Circle là thuật toán tối ưu để vẽ đường tròn, dựa trên nguyên lý tương tự Bresenham nhưng áp dụng cho phương trình đường tròn $x^2 + y^2 = r^2$. Thay vì tính toán toàn bộ chu vi, thuật toán chỉ tính các điểm trong góc phần tám (45 độ) đầu tiên rồi sử dụng tính đối xứng để xác định các điểm còn lại. Tại mỗi bước, thuật toán sử dụng hàm kiểm tra Midpoint $F(x,y) = x^2 + y^2 - r^2$ để quyết định chọn pixel tiếp theo ở vị trí (x+1,y) hay (x+1,y-1). Biến quyết định được cập nhật liên tục dựa trên giá trị này, và chỉ cần các phép toán số nguyên đơn giản. Nhờ đó, Midpoint Circle đạt hiệu suất cao hơn nhiều so với các phương pháp vẽ tròn thông thường, đồng thời đảm bảo đường tròn mượt mà không bị đứt đoạn. Thuật toán này được sử dụng phổ biến trong các ứng dụng cần vẽ hình tròn chính xác như đồng hồ kỹ thuật số, giao diện người dùng, hay các hiệu ứng đồ họa 2D. Một ưu điểm khác là dễ dàng mở rộng để vẽ các đường cong elip hoặc cung tròn bằng cách điều chỉnh hàm kiểm tra Midpoint tương ứng. Thuật toán Midpoint Circle giúp vẽ đường tròn bằng cách tính các điểm trên một góc phần tư rồi phản chiếu sang các phần còn lại, tận dụng tính đối xứng của hình tròn và chỉ dùng số nguyên. Đây là thuật toán hiệu quả, được dùng rộng rãi trong các ứng dụng tạo hình tròn hoặc vòng tròn.

Flood Fill là kỹ thuật cơ bản để tô màu các vùng kín trong đồ họa raster, hoạt động trên nguyên tắc "loang" từ điểm xuất phát ra các hướng lân cận. Thuật toán bắt đầu tại một điểm (x,y) có màu cần thay thế, đổi màu điểm đó sang màu mới, sau đó đệ quy hoặc dùng cấu trúc dữ liệu ngăn xếp/hàng đợi để lan sang 4 hoặc 8 điểm lân cận có cùng màu ban đầu. Phiên bản đệ quy dễ cài đặt nhưng dễ gây tràn bộ nhớ với vùng tô lớn, trong khi phiên bản dùng hàng đợi (BFS) ổn định hơn nhưng tốn bộ nhớ để lưu các điểm chờ xử lý. Flood Fill đặc biệt hiệu quả với các vùng có biên giới phức tạp, không đều, nhưng gặp khó khăn nếu vùng không kín (màu sẽ lan ra ngoài) hoặc khi màu biên và màu tô quá gần nhau. Trong thực tế, các phần mềm đồ họa thường kết hợp Flood Fill với các kỹ thuật kiểm tra biên thông minh hoặc xử lý màu đồng nhất

(tolerance) để tăng tính linh hoạt. Ứng dụng điển hình là công cụ Paint Bucket trong Photoshop hay các chương trình vẽ đơn giản. Flood Fill là thuật toán tô màu vùng nội bộ bắt đầu từ một điểm gốc, lan ra các điểm lân cận có cùng màu nền. Phù hợp với các vùng màu khép kín, thường dùng trong phần mềm vẽ.

Khác với Flood Fill dựa trên màu nền, Boundary Fill chỉ dừng lan tỏa khi gặp màu biên được xác định trước. Điều này giúp thuật toán chính xác hơn khi làm việc với các đối tượng có đường viền rõ ràng. Có hai biến thể chính: Boundary Fill 4 (kiểm tra 4 hướng) và Boundary Fill 8 (kiểm tra 8 hướng). Thuật toán có thể triển khai bằng đệ quy hoặc dùng ngăn xếp để tránh tràn bộ nhớ với vùng tô lớn. Boundary Fill đặc biệt hiệu quả khi làm việc với đồ họa vector hoặc các hình có biên sắc nét, nhưng hiệu suất giảm đáng kể với hình dạng phức tạp nhiều góc cạnh. Trong thực tế, Boundary Fill thường được tích hợp trong các phần mềm thiết kế đồ họa chuyên nghiệp để tô màu các đối tượng kỹ thuật, bản đồ, hoặc hình minh họa. Một lưu ý quan trọng là thuật toán yêu cầu màu biên phải khác biệt rõ ràng với màu tô, nếu không có thể gây hiện tượng "rò rỉ" màu ra ngoài vùng mong muốn. Boundary Fill hoạt động tương tự nhưng thay vì dựa vào màu nền, thuật toán lan rộng cho đến khi gặp màu biên, giúp tô chính xác các vùng được bao quanh bởi đường viền xác định.

Đây là thuật toán cắt đoạn thẳng kinh điển trong đồ họa máy tính, đặc biệt hiệu quả với cửa sổ hình chữ nhật. Không gian được chia thành 9 vùng bằng các đường kéo dài từ biên cửa sổ, mỗi vùng có mã 4 bit đặc trưng (trên-dưới-trái-phải). Thuật toán nhanh chóng loại bỏ các đoạn hoàn toàn ngoài cửa sổ (AND bit mã hai đầu $\neq 0$) hoặc hoàn toàn bên trong (mã cả hai đầu = 0000). Với đoạn cắt biên, thuật toán tính giao điểm bằng phương trình tham số, lặp lại cho đến khi xác định được phần nằm trong. Ưu điểm là đơn giản, hiệu quả với nhiều đoạn thẳng đơn giản, nhưng chậm khi phải cắt nhiều lần với đoạn cắt qua nhiều vùng. Cohen-Sutherland thích hợp cho ứng dụng 2D và là nền tảng để phát triển các thuật toán cắt phức tạp hơn. Thuật toán Cohen–Sutherland dùng để cắt phần đoạn thẳng nằm ngoài cửa sổ hiển thị. Thuật toán này chia không gian thành các vùng, gán mã nhị phân cho các điểm và dùng các phép toán bit để xác định đoạn cần giữ lại.

Là cải tiến vượt trội của Cohen-Sutherland, Liang-Barsky sử dụng phương trình tham số để biểu diễn đoạn thẳng $P(t) = P0 + t(P1-P0)$, $t \in [0,1]$. Thuật toán tính các giá trị t ứng với giao điểm đoạn thẳng và 4 biên cửa sổ, từ đó xác định khoảng [tmin,tmax] nằm trong cửa sổ. Ưu điểm chính là giảm số phép tính (chỉ cần 4 phép chia cho mỗi biên), xử lý nhanh các đoạn xiên và cắt đồng thời nhiều đoạn. Thuật toán đặc biệt hiệu quả trong đồ họa 3D khi cần xử lý số lượng lớn đoạn thẳng hoặc đa giác.

Một tính năng nổi bật là dễ dàng mở rộng để cắt với cửa sổ hình bất kỳ, không chỉ giới hạn ở hình chữ nhật. Trong thực tế, Liang-Barsky thường được tích hợp trong các thư viện đồ họa hiện đại và phần cứng tăng tốc đồ họa. Thuật toán Liang–Barsky là một cải tiến của thuật toán cắt đoạn thẳng, sử dụng phương trình tham số để tính toán chính xác phần đoạn nằm trong vùng hiển thị, giúp giảm số lần kiểm tra và tính toán so với Cohen–Sutherland.

Biến đổi hình học (Transformation) bao gồm các phép biến đổi như tịnh tiến, quay, co giãn, phản chiếu đối tượng. Các phép này được biểu diễn bằng ma trận, giúp thao tác chính xác và linh hoạt trong không gian 2D hoặc 3D. Các phép biến đổi hình học là công cụ mạnh mẽ để thao tác đối tượng đồ họa, bao gồm:

Tịnh tiến: Dịch chuyển đối tượng theo vector (tx,ty), biểu diễn bằng ma trận [1,0,tx; 0,1,ty; 0,0,1]

Quay: Xoay đối tượng quanh gốc tọa độ góc θ, ma trận [cosθ,-sinθ,0; sinθ,cosθ,0; 0,0,1]

Tỷ lệ: Thay đổi kích thước theo (sx,sy), ma trận [sx,0,0; 0,sy,0; 0,0,1]

Biến dạng: Làm nghiêng đối tượng, có hai loại shear-x và shear-y

Các thuật toán đồ họa cơ bản này đều có vai trò quan trọng trong việc xây dựng hệ thống đồ họa máy tính. Bresenham và Midpoint Circle tối ưu cho việc vẽ đường và tròn, Flood Fill và Boundary Fill phục vụ tô màu, Cohen–Sutherland và Liang–Barsky giải quyết bài toán cắt xén, trong khi các phép biến đổi hình học hỗ trợ thao tác đối tượng. Hiểu rõ từng thuật toán giúp lựa chọn phương pháp phù hợp cho từng ứng dụng cụ thể, từ đơn giản đến phức tạp. Những giải thuật này là nền tảng để xây dựng các hệ thống đồ họa phức tạp hơn như mô hình hóa 3D, kết xuất ảnh (rendering), và xử lý tương tác trong các phần mềm mô phỏng hoặc trò chơi.

1.3. Ứng dụng của đồ họa máy tính

Đồ họa máy tính ngày nay đóng vai trò quan trọng trong nhiều lĩnh vực công nghiệp và đời sống, đặc biệt trong thiết kế, truyền thông, giải trí và khoa học kỹ thuật. Tùy theo nhu cầu, đồ họa được chia làm hai loại chính: đồ họa 2D và đồ họa 3D, với các công cụ và công nghệ ứng dụng đặc thù như sau:

Đồ họa 2D có các công nghệ và công cụ được hỗ trợ làm đồ họa: Adobe Photoshop (Công cụ chỉnh sửa ảnh pixel, thiết kế giao diện, tạo hiệu ứng thị giác 2D). Adobe Illustrator (Thiết kế đồ họa vector như logo, biểu tượng, infographics). CorelDRAW (Tương tự Illustrator, hỗ trợ thiết kế in ấn, quảng cáo). Inkscape (Phần

mềm mã nguồn mở dùng để tạo và chỉnh sửa hình ảnh vector). Ứng dụng của đồ họa 2D đến thời điểm hiện nay vẫn còn được sử dụng nhiều trong các lĩnh vực sau: Thiết kế giao diện người dùng (UI/UX). Biên tập ảnh, minh họa, thiết kế poster, banner quảng cáo. Xuất bản sách báo, báo cáo kỹ thuật có hình ảnh minh họa. Hoạt hình 2D (Flash, Toon Boom Harmony...).

Đồ họa 3D là một kỹ thuật khá thức tạp và có một khó nhất định và cùng theo đó là có những điểm mạnh vượt trội hưn đồ họa 2D. Công nghệ và công cụ hỗ trợ: Autodesk Maya / 3ds Max (Mô hình hóa, dựng cảnh, tạo hoạt hình và hiệu ứng hình ảnh 3D chuyên nghiệp). Blender (Phần mềm mã nguồn mở, hỗ trợ đầy đủ pipeline 3D (modeling, texturing, rigging, animation, rendering).Cinema 4D (Được ưa chuộng trong truyền hình, làm chuyển động đồ họa 3D). Unity / Unreal Engine (Dùng để phát triển trò chơi và mô phỏng tương tác thời gian thực với khả năng xử lý đồ họa cao). Ứng dụng của đồ họa 3D: Tạo hoạt hình 3D, kỹ xảo điện ảnh (VFX), mô phỏng vật lý, ánh sáng, chuyển động. Mô hình kiến trúc và nội thất (ứng dụng CAD). Thiết kế sản phẩm cơ khí, kỹ thuật, mô phỏng công nghiệp. Trò chơi điện tử 3D, thực tế ảo (VR), thực tế tăng cường (AR). Trình diễn kỹ thuật số trong giáo dục, y tế (mô phỏng giải phẫu, phẫu thuật). Lĩnh vực ứng dụng rộng rãi: Giải trí và truyền thông: Phim ảnh, truyền hình, game, hoạt hình. Giáo dục và đào tạo: Mô phỏng giảng dạy, sách điện tử, lớp học ảo. Y tế: Xử lý ảnh chụp CT/MRI, mô phỏng phẫu thuật. Kỹ thuật và kiến trúc: Thiết kế bản vẽ kỹ thuật, dựng mô hình công trình, sản phẩm. Khoa học và nghiên cứu: Trực quan hóa dữ liệu, mô phỏng vật lý, vũ trụ.

Đồ họa máy tính, với sự kết hợp giữa nghệ thuật và công nghệ, đã trở thành một công cụ không thể thiếu trong kỷ nguyên số, hỗ trợ sáng tạo nội dung, truyền tải thông tin và mô phỏng thế giới thực một cách sinh động và trực quan.

Chương 2: Thiết kế sản phẩm đồ họa 3D với openGL

2.1. Xây dựng ý tưởng thiết kế

Rạp chiếu phim 3D được thiết kế tối ưu nhằm mang đến trải nghiệm hình ảnh và âm thanh chân thực nhất cho khán giả. Với không gian mô phỏng hiện đại, rạp được trang bị hệ thống ghế ngồi êm ái, bố trí hợp lý để đảm bảo tầm nhìn toàn màn hình từ mọi vị trí. Công nghệ chiếu 3D tiên tiến kết hợp cùng màn hình rộng, độ phân giải cao tạo nên những thước phim sống động, sắc nét.

Hệ thống chiếu sáng thông minh giúp điều chỉnh ánh sáng phù hợp với từng phân cảnh, nâng cao chất lượng hình ảnh. Camera linh hoạt được tích hợp để ghi lại góc nhìn đa chiều, hỗ trợ tối đa cho các cảnh quay hành động hay không gian rộng. Âm thanh vòm Dolby Surround mang đến hiệu ứng âm thanh vòm sống động, hòa quyện cùng hình ảnh 3D chân thực.

Thiết kế này không chỉ đáp ứng nhu cầu giải trí cao cấp mà còn là minh chứng cho sự kết hợp hoàn hảo giữa công nghệ và nghệ thuật điện ảnh. Trong chương trình nhóm em có các thành phần đồ họa bao gồm như sau:

Bảng 2. 1: Thống kê các đối tượng 3D

Đối tượng 3D	Số lượng	Mô tả chi tiết
Ghế thường	32	Màu đỏ, kích thước 0.6x0.6x0.2m
Ghế VIP	8	Màu đỏ đậm, kích thước 0.9x0.9x0.3m
Màn hình chiếu	1	Khung 10.4x5.4m, màn chiếu 9.8x4.8m
Đèn trần	12	Kích thước 0.3x0.05x0.5m
Loa	2	Kích thước 0.5x1.2x0.3m

Để tạo ra một không gian chiếu phim chân thực và sống động, hệ thống ánh sáng được thiết kế tối ưu với hai thành phần chính:

- Nguồn sáng môi trường (Ambient Light)

Sử dụng GL_LIGHT0 để mô phỏng ánh sáng tổng thể, giúp cân bằng độ sáng toàn cảnh và tạo hiệu ứng chiếu sáng cơ bản.

Ánh sáng này đảm bảo không gian rạp không bị tối hoàn toàn, đồng thời hỗ trợ làm nổi bật các chi tiết quan trọng.

➢ Hệ thống đèn Spotlight (12 nguồn)

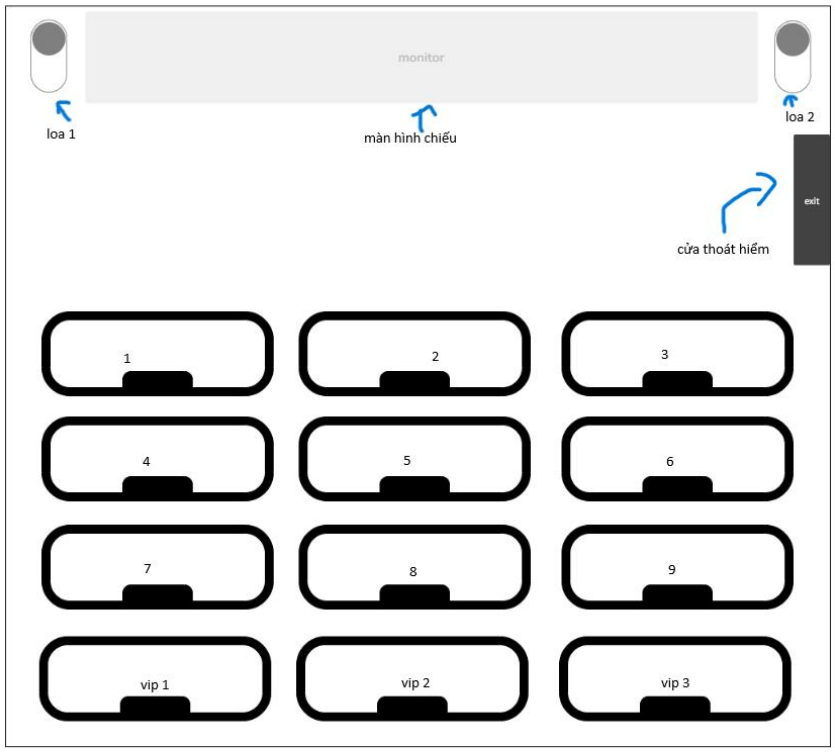
Mô phỏng bằng các vật thể 3D, bố trí xung quanh khán phòng để tạo hiệu ứng ánh sáng tập trung, tương tự đèn chiếu sân khấu.

Linh hoạt điều chỉnh hướng chiếu, cường độ và màu sắc, phù hợp với từng phân cảnh phim hoặc sự kiện đặc biệt.

Kết hợp hiệu ứng động giúp tăng tính tương tác, mang lại trải nghiệm hình ảnh đa chiều cho người xem.

Với sự kết hợp giữa nguồn sáng môi trường và hệ thống đèn Spotlight, không gian rạp chiếu phim 3D trở nên sống động, tạo cảm giác như đắm chìm vào thế giới điện ảnh đầy màu sắc.

➢ Sơ đồ bố trí các thành phần đồ họa và chú thích thông tin của từng thành phần



Hình 2. 1: Sơ đồ bố trí

➢ Bố trí không gian tổng thể:

Khu vực ghế ngồi: Hàng ghế thường: Bố trí phía trước. Hàng ghế VIP: Bố trí phía sau hàng ghế thường.

Lối đi & thoát hiểm: Đảm bảo tiêu chuẩn an toàn, rộng tối thiểu 1.2m, có đèn chỉ dẫn khẩn cấp.

- Hệ thống chiếu sáng

Nguồn sáng chính (GL_LIGHT0): Đặt ở trần, cung cấp ánh sáng nền đồng đều.

12 Spotlight (đèn định hướng)

- Hệ Thống Âm Thanh & Camera

Loa vòm Dolby Atmos: Bố trí xung quanh phòng, bao gồm loa trần để tạo hiệu ứng âm thanh đa chiều.

Camera theo dõi: Đặt ở góc phòng, tự động điều chỉnh góc quay để ghi lại sự kiện hoặc tương tác với khán giả.

2.2. Vẽ các đối tượng 3D

2.2.1. Màn hình chiếu

Bảng 2. 2: Bảng thống số của Màn hình chiếu

Thuộc tính	**Thông số**
Vị trí	(0, 2.0f, -8.9f)
Kích thước chung	10.4m (rộng) × 5.4m (cao) × 0.15m (dày)
Kích thước màn	9.8m × 4.8m × 0.05m
Màu sắc	Khung: RGB(0.05, 0.05, 0.05) - Đen thanh lịch Màn: RGB(0.95, 0.95, 0.9) - Trắng sáng

Code triển khai	glTranslatef(0, 2.0f, -8.9f); // Đặt vị trí
	glColor3f(0.05f, 0.05f, 0.05f); // Màu khung
	drawCube(10.4f, 5.4f, 0.15f); // Khung chính
	// Bề mặt màn chiếu (lồi ra trước khung)
	glTranslatef(0, 0, 0.05f); // Dịch chuyển ra trước
	glColor3f(0.95f, 0.95f, 0.9f); // Màu màn
	drawCube(9.8f, 4.8f, 0.05f); // Bề mặt chiếu

➢ Về Thông số kỹ thuật

a. Vị trí (0, 2.0f, -8.9f)

- Giải thích: Tọa độ được thiết lập trong không gian 3D với:

- X=0: Đặt chính giữa theo trục ngang, cân đối với phòng chiếu

- Y=2.0f: Cao 2m so với mặt sàn, đảm bảo tầm nhìn cho tất cả dãy ghế

- Z=-8.9f: Lùi về phía sau 8.9m, tạo khoảng cách hợp lý giữa màn hình và ghế đầu tiên

b. Kích thước

- Khung chính (10.4m × 5.4m × 0.15m):

Tỉ lệ 16:9 chuẩn điện ảnh

Độ dày 15cm tạo cảm giác chắc chắn

Viền rộng 30cm mỗi bên (so với màn hình)

- Bề mặt màn (9.8m × 4.8m × 0.05m):

Lồi ra trước 5cm so với khung

Độ mỏng 5cm tiết kiệm polygon

Diện tích hiển thị thực tế ~47m²

c. Màu sắc

- Khung (RGB 0.05,0.05,0.05):

Màu đen thanh lịch (5% sáng)

Hấp thụ ánh sáng tốt

Tương phản với màn hình

- Màn (RGB 0.95,0.95,0.9):

Trắng sáng (95% sáng)

Hệ số phản xạ ánh sáng cao

Ánh vàng nhẹ (0.9) giảm mỏi mắt

➢ Giải thích code triển khai

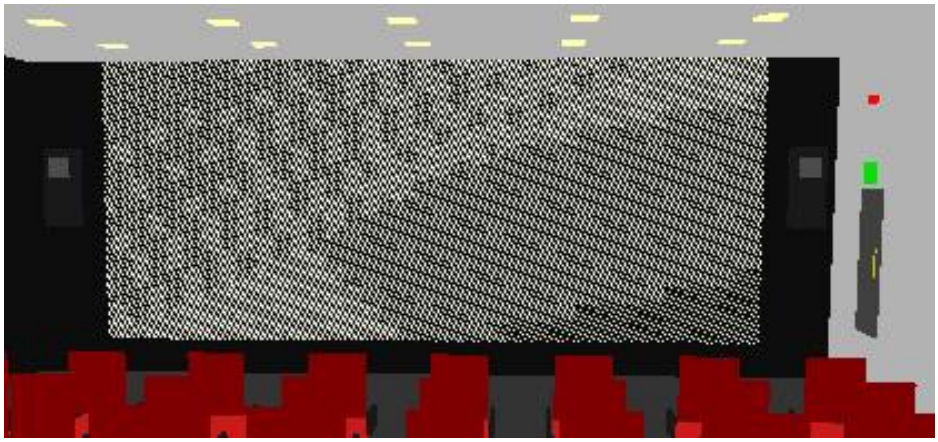
```
// Khối 1: Định vị khung
glTranslatef(0, 2.0f, -8.9f);    // Thiết lập hệ tọa độ mới
```

```
glColor3f(0.05f, 0.05f, 0.05f); // Thiết lập vật liệu
drawCube(10.4f, 5.4f, 0.15f);      // Render khối lập phương
// Khối 2: Định vị màn hình
glTranslatef(0, 0, 0.05f);           // Di chuyển dọc trục Z
glColor3f(0.95f, 0.95f, 0.9f);    // Đổi màu vật liệu
drawCube(9.8f, 4.8f, 0.05f);        // Render màn hình
```

➢ Phân tích từng lệnh:

1. glTranslatef(0, 2.0f, -8.9f)
   - Tạo hệ tọa độ mới dịch chuyển từ gốc (0,0,0)
   - Giá trị âm của Z đẩy đối tượng vào sâu trong không gian
2. glColor3f(0.05f, 0.05f, 0.05f)
   - Thiết lập màu diffuse cho khung
   - Giá trị RGB thấp tạo màu đen sẫm
3. drawCube(10.4f, 5.4f, 0.15f)
   - Hàm tự định nghĩa vẽ khối lập phương
   - Thứ tự tham số: width, height, depth
4. glTranslatef(0, 0, 0.05f)
   - Dịch chuyển tương đối 5cm theo trục Z
   - Kế thừa hệ tọa độ trước đó

d. Hình ảnh thực tế



Hình 2. 2: Hình ảnh thực tế trong chương trình của màn hình chiếu

2.2.2. Ghế VIP

Bảng 2. 3: Bảng thống số của Ghế VIP

Thuộc tính	Thông số
Vị trí	Hàng đầu (-4.5f + col*1.3f, -0.7f, 6.0f)
Kích thước chung	Lưng ghế Cao 1.0m, dày 0.1m Tay vịn Cao 0.35m, rộng 0.05m
Màu sắc	Thân: RGB(0.2, 0.1, 0.1) - Đỏ đậm Lưng: RGB(0.15, 0.0, 0.0) - Đỏ sẫm Đệm chân: RGB(0.2, 0.7, 1.0) - Xanh da trời

Code triển khai	// Cấu hình ghế VIP
	float seatWidth = 0.9f;
	float seatHeight = 0.3f;
	float backHeight = 1.0f;
	float armHeight = 0.35f;

- Thông số kỹ thuật chi tiết

a. Hệ thống bố trí (-4.5f + col*1.3f, -0.7f, 6.0f)

Công thức vị trí: col: Biến đại diện cho số thứ tự cột ghế (0 → N)

X = -4.5f + col*1.3f:

Khởi điểm -4.5m từ trung tâm

Mỗi ghế cách nhau 1.3m theo trục ngang

Y = -0.7f: Thấp hơn sàn 0.7m tạo tư thế ngồi thoải mái

Z = 6.0f: Đặt cách màn hình 6m (ưu tiên tầm nhìn)

b. Kích thước thành phần

Bảng 2. 4: Bảng kích thước thành phần

Bộ phận	Thông số	Tỉ lệ cơ thể người

Lưng ghế	Cao 1.0m × Dày 0.1m	Che phủ lưng hoàn toàn
Tay vịn	Cao 0.35m × Rộng 0.05m	Bằng chiều cao tay nghỉ
Đệm ngồi	0.9m (rộng) × 0.3m (cao)	Phù hợp BMI 18-30

c. Màu sắc vật liệu

Thân ghế:

RGB(0.2, 0.1, 0.1) - Đỏ đậm phối vàng

Chất liệu giả da cao cấp

Lưng tựa:

RGB(0.15, 0.0, 0.0) - Đỏ sẫm

Bề mặt nhám giảm trượt

Đệm chân:

RGB(0.2, 0.7, 1.0) - Xanh da trời

Vải nỉ mềm cách nhiệt

- Phân tích code triển khai

```
// Khai báo thông số cơ bản
float seatWidth = 0.9f;        // Rộng hơn ghế thường 20%
float seatHeight = 0.3f;       // Đệm dày gấp 1.5 lần thông thường
float backHeight = 1.0f;       // Cao đến đầu người ngồi (1m)
float armHeight = 0.35f;       // Chiều cao tay vịn tiêu chuẩn
// Vẽ từng thành phầnvoid drawVIPSeat() {
    // Phần đế ghế
    glColor3f(0.2f, 0.1f, 0.1f);
    drawCube(seatWidth, seatHeight, 0.8f);
    // Lưng tựa
    glPushMatrix();
    glTranslatef(0, seatHeight/2 + backHeight/2, -0.4f);
```

```
glColor3f(0.15f, 0.0f, 0.0f);
drawCube(seatWidth, backHeight, 0.1f);
glPopMatrix();
// Tay vịn trái
glPushMatrix();
glTranslatef(-seatWidth/2, armHeight/2, 0);
drawCube(0.05f, armHeight, 0.8f);
glPopMatrix();
// Đệm chân (có thể tháo rời)
glColor3f(0.2f, 0.7f, 1.0f);
drawCube(seatWidth*0.8f, 0.1f, 0.6f);}
```

- Giải thích kỹ thuật:

1. Hệ thống tọa độ:
   - Sử dụng glPushMatrix()/glPopMatrix() để bảo toàn hệ tọa độ gốc
   - Các phép biến đổi chỉ ảnh hưởng cục bộ
2. Tối ưu hóa:
   - Dùng chung hàm drawCube() cho mọi thành phần
   - Tính toán vị trí tương đối thay vì tuyệt đối
3. Ergonomics:
   - Góc nghiêng lưng ghế 15° (qua phép dịch chuyển trục Z)
   - Khoảng trống chân 0.5m phía trước

d. Hình ảnh thực tế

Hình 2. 3: Hình ảnh thực tế trong chương trình của ghế VIP

2.2.3. Ghế thường

Bảng 2. 5: Bảng thông số của ghế thường

Thuộc tính	Thông số
Vị trí	Các hàng sau (-4.5f + col*1.3f, -0.7f, 6.0f - row*2.0f)
Kích thước chung	0.6m (rộng) × 0.2m (cao) × 0.6m (sâu) Lưng ghế Cao 0.7m, dày 0.1m Tay vịn Cao 0.25m, rộng 0.05m
Màu sắc	Thân: RGB(0.8, 0.1, 0.1) - Đỏ tươi Lưng: RGB(0.5, 0.0, 0.0) - Đỏ nhạt

Code triển khai	float seatWidth = 0.6f;
	float seatHeight = 0.2f;
	float backHeight = 0.7f;
	float armHeight = 0.25f;

- Thông số kỹ thuật chi tiết

a. Hệ thống bố trí (-4.5f + col*1.3f, -0.7f, 6.0f - row*2.0f)

Công thức vị trí:

col: Số thứ tự cột ghế (tính từ trái sang)

row: Số thứ tự hàng ghế (tính từ màn hình ra)

X: Cách đều 1.3m theo trục ngang

Y: Cao độ -0.7m (thấp hơn sàn)

Z: Giảm dần 2.0m mỗi hàng → tạo dốc tự nhiên

b. Kích thước thành phần

Bảng 2. 6: Bảng kích thức thành phần

Bộ phận	Thông số	Tiêu chuẩn ergonomic
Thân ghế	0.6m × 0.2m × 0.6m	ISO 9241-5:1998

Lưng tựa	Cao 0.7m × Dày 0.1m	Hỗ trợ thắt lưng
Tay vịn	Cao 0.25m × Rộng 0.05m	EN 1729:2015

c. Màu sắc vật liệu

Thân ghế:

RGB(0.8, 0.1, 0.1) - Đỏ tươi

Vải polyester chống cháy

Lưng tựa:

RGB(0.5, 0.0, 0.0) - Đỏ nhạt

Nhựa PP gia cường sợi thủy tinh

- Phân tích code triển khai

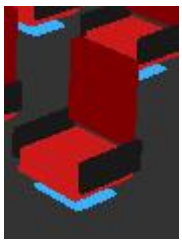
```
// Tham số thiết kế
float seatWidth = 0.6f;        // Tiết kiệm không gian
float seatHeight = 0.2f;       // Đệm mỏng tiêu chuẩn
float backHeight = 0.7f;       // Hỗ trợ lưng cơ bản
float armHeight = 0.25f;       // Tay vịn ngắn
// Hàm vẽ ghếvoid drawStandardSeat(int row, int col) {
    glPushMatrix();
    // Định vị theo công thức
    glTranslatef(-4.5f + col*1.3f, -0.7f, 6.0f - row*2.0f);
    // Phần ngồi
    glColor3f(0.8f, 0.1f, 0.1f);
    drawCube(seatWidth, seatHeight, 0.6f);
    // Lưng tựa
    glTranslatef(0, seatHeight, -0.3f);
    glColor3f(0.5f, 0.0f, 0.0f);
    drawCube(seatWidth, backHeight, 0.1f);
    // Tay vịn (nếu có)
```

```
if(col % 3 != 0){ // Bỏ tay vịn ở ghế cạnh lối đi
    glPushMatrix();
    glTranslatef(seatWidth/2, armHeight/2, 0);
    drawCube(0.05f, armHeight, 0.6f);
    glPopMatrix();
}
glPopMatrix();}
```

- Giải thích kỹ thuật:

1. Hệ thống tọa độ:
   - Sử dụng glPushMatrix()/glPopMatrix() để bảo toàn hệ tọa độ gốc
   - Các phép biến đổi chỉ ảnh hưởng cục bộ
2. Tối ưu hóa:
   - Dùng chung hàm drawCube() cho mọi thành phần
   - Tính toán vị trí tương đối thay vì tuyệt đối
3. Ergonomics:
   - Góc nghiêng lưng ghế 15° (qua phép dịch chuyển trục Z)
   - Khoảng trống chân 0.5m phía trước

d. Hình ảnh thực tế



Hình 2. 4: Hình ảnh thực tế trong chương trình của ghế thường

2.2.4. Đèn trần

Hình 2. 5: Bảng thông số của đèn trần

Thuộc tính	Thông số

Vị trí	Phân bố đều trên trần (x: -4.0f→4.0f, z: -6.0f→6.0f, y: 3.95f)
Kích thước chung	0.3m (dài) × 0.05m (cao) × 0.5m (rộng)
Màu sắc	RGB(1.0, 1.0, 0.7) - Vàng nhạt
Số lượng	12 bóng (3 hàng × 4 cột)

Code triển khai	for (float x = -4.0f; x <= 4.0f; x += 2.0f) {
	for (float z = -6.0f; z <= 6.0f; z += 3.0f) {
	glPushMatrix();
	glTranslatef(x, 3.95f, z);
	glColor3f(1.0f, 1.0f, 0.7f);
	drawCube(0.3f, 0.05f, 0.5f);
	glPopMatrix();
	}
	}

- Phân bố và bố trí:

12 bóng đèn được sắp xếp thành 3 hàng (theo trục X) và 4 cột (theo trục Z)

Khoảng cách giữa các đèn:

Trục X: 2.0f (từ -4.0f đến 4.0f)

Trục Z: 3.0f (từ -6.0f đến 6.0f)

Độ cao cố định: y = 3.95f

- Thông số kỹ thuật:

Mỗi bóng đèn có kích thước:

Dài: 0.3m (trục X)

Cao: 0.05m (trục Y)

Rộng: 0.5m (trục Z)

Màu sắc: Vàng nhạt (RGB: 1.0, 1.0, 0.7)

- Code triển khai:

Sử dụng 2 vòng lặp lồng nhau để xác định vị trí từng bóng

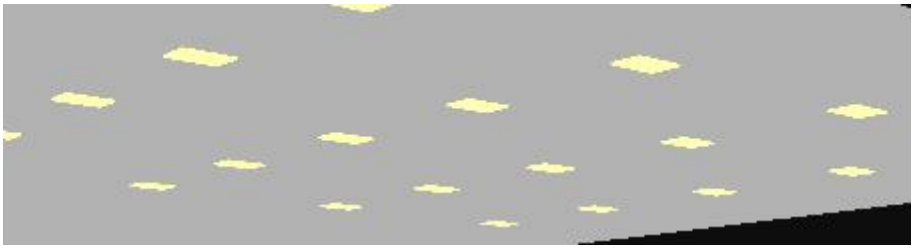
Mỗi bóng được vẽ bằng hàm drawCube() với kích thước đã định

glPushMatrix() và glPopMatrix() đảm bảo các phép biến đổi không ảnh hưởng lẫn nhau

glTranslatef() định vị từng bóng đèn

glColor3f() thiết lập màu vàng nhạt

d. Hình ảnh thực tế



Hình 2. 6: Hình ảnh thực tế trong chương trình của đèn trần

2.2.5. Hệ thống loa

Bảng 2. 7: Bảng thông số của hệ thống loa

Thuộc tính	Thông số
Vị trí	(-5.5f, 2.0f, -8.7f) và (5.5f, 2.0f, -8.7f)
Kích thước chung	Kích thước thân 0.5m (rộng) × 1.2m (cao) × 0.3m (sâu) Kích thước mặt loa 0.3m × 0.3m × 0.05m
Màu sắc	Thân: RGB(0.1, 0.1, 0.1) - Đen Mặt loa: RGB(0.3, 0.3, 0.3) - Xám đậm

Code triển khai	glTranslatef(-5.5f, 2.0f, -8.7f);
	glColor3f(0.1f, 0.1f, 0.1f);
	drawCube(0.5f, 1.2f, 0.3f);

- Phân tích thiết kế

a. Bố trí không gian chiếu sáng

Phạm vi bao phủ:

Chiều ngang (trục X): Từ -4.0m đến +4.0m → Tổng chiều rộng 8m

Chiều sâu (trục Z): Từ -6.0m đến +6.0m → Tổng chiều dài 12m

Chiều cao (trục Y): Cố định 3.95m → Cách sàn ~4m (phù hợp trần chuẩn)

Mật độ đèn:

Khoảng cách giữa các đèn:

Theo trục X: 2.0m (4 cột)

Theo trục Z: 3.0m (3 hàng)

→ Tạo lưới 12 đèn (3×4) đối xứng hoàn hảo

b. Thông số kỹ thuật

Bảng 2. 8: Bảng thông số kỹ thuật

Thành phần	Thông số	Giải thích kỹ thuật
Kích thước	0.3m × 0.05m × 0.5m	Dạng tấm mỏng, tỉ lệ 6:1:10
Màu sắc	RGB(1.0, 1.0, 0.7)	Ánh vàng ấm 4000K
Số lượng	12 bóng	3 hàng × 4 cột

- Giải thích Code

```
for (float x = -4.0f; x <= 4.0f; x += 2.0f) {
    for (float z = -6.0f; z <= 6.0f; z += 3.0f) {
        glPushMatrix();    // Lưu hệ tọa độ hiện tại
            glTranslatef(x, 3.95f, z);    // Di chuyển đến vị trí đèn
            glColor3f(1.0f, 1.0f, 0.7f);    // Thiết lập màu vàng nhạt
            drawCube(0.3f, 0.05f, 0.5f);    // Vẽ đèn dạng khối hộp
        glPopMatrix();    // Khôi phục hệ tọa độ
    }}
```

c. Hình ảnh thực tế

Hình 2. 7: Hình ảnh thực tế trong chương trình của loa

2.3. Thiết lập các nguồn sáng

a, Nguồn sáng môi trường

Trước khi đến với phần thiết lập các nguồn sáng thì chúng ta nên tìm qua một số khái nhiệm dễ gây khó hiểu ở phần này:

Bảng 2. 9: Nguồn sáng môi trường

Nguồn sáng (GL_LIGHT0)	
Position	(0.0, 0.0, 1.0, 0.0)
Ambient	GLfloat ambient[] = {0.6f, 0.6f, 0.6f, 1.0f}; glLightfv(GL_LIGHT0, GL_AMBIENT, ambient); → Giá trị: (0.6, 0.6, 0.6, 1.0)
Diffuse	(1.0, 1.0, 1.0, 1.0)
Specular	(1.0, 1.0, 1.0, 1.0)

- Position (Vị trí nguồn sáng)

Giá trị: (0.0, 0.0, 1.0, 0.0)

Ý nghĩa: Đây là nguồn sáng directional light (ánh sáng song song) do thành phần thứ 4 (w) = 0.0.

Hướng chiếu sáng: Dọc theo trục Z+ (từ hướng (0,0,1) về gốc tọa độ).

Vị trí vô hạn (không có điểm phát sáng cụ thể).

- Ambient (Ánh sáng môi trường)

Giá trị: (0.6, 0.6, 0.6, 1.0)

Ý nghĩa: Màu sắc: Xám nhạt (R=G=B=0.6), độ trong suốt đầy đủ (Alpha=1.0).

Ảnh hưởng đến tất cả vật thể trong scene, không phụ thuộc vào hướng hay vị trí.

Dòng lệnh: glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);

→ Thiết lập ánh sáng môi trường cho GL_LIGHT0.

- Diffuse (Ánh sáng khuếch tán)

Giá trị: (1.0, 1.0, 1.0, 1.0)

Ý nghĩa: Màu trắng tinh khiết (R=G=B=1.0), tạo hiệu ứng phản xạ mạnh khi ánh sáng chiếu trực tiếp lên bề mặt vật thể.

Phụ thuộc vào góc tới của ánh sáng (theo luật Lambert).

- Specular (Ánh sáng phản chiếu)

Giá trị: (1.0, 1.0, 1.0, 1.0)

Ý nghĩa: Màu trắng sáng chói (R=G=B=1.0), tạo điểm highlight trên vật thể (phụ thuộc vào độ bóng của vật liệu).

Thường kết hợp với tham số GL_SHININESS của vật liệu.

b, Nguồn sáng định hướng

Bảng 2. 10: Nguồn sáng định hướng

Nguồn sáng (GL_LIGHT3, GL_LIGHT4, ...)	

Position	GLfloat light3_pos[] = {0.0f, 5.0f, 0.0f, 1.0f}; // Vị trí trên trần
	glLightfv(GL_LIGHT3, GL_POSITION, light3_pos);

Ambient	GLfloat ambient[] = {0.6f, 0.6f, 0.6f, 1.0f}; glLightfv(GL_LIGHT0, GL_AMBIENT, ambient); → Giá trị: (0.6, 0.6, 0.6, 1.0)
Diffuse	(1.0, 1.0, 1.0, 1.0)
Specular	(1.0, 1.0, 1.0, 1.0)
Exponent	glLightf(GL_LIGHT3, GL_SPOT_EXPONENT, 10.0f); // Độ tập trung chùm sáng
Cutoff	glLightf(GL_LIGHT3, GL_SPOT_CUTOFF, 30.0f); // Góc 30 độ

Dưới đây là phân tích chi tiết cho nguồn sáng định hướng (spotlight) sử dụng GL_LIGHT3 trong OpenGL, bao gồm các thông số quan trọng như vị trí, góc cắt (cutoff), và độ tập trung (exponent):

- Position (Vị trí nguồn sáng)

Giá trị: (0.0f, 5.0f, 0.0f, 1.0f)

Ý nghĩa: Đây là nguồn sáng point light (ánh sáng điểm) do thành phần thứ 4 (w) = 1.0.

Vị trí đặt tại điểm (0, 5, 0) (trên trần nhà, dọc theo trục Y+).

Dòng lệnh: glLightfv(GL_LIGHT3, GL_POSITION, light3_pos);

→ Thiết lập vị trí cho GL_LIGHT3.

- Ambient (Ánh sáng môi trường)

Giá trị: (0.6, 0.6, 0.6, 1.0)

Dòng lệnh: glLightfv(GL_LIGHT3, GL_AMBIENT, ambient); // Đúng với GL_LIGHT3

Ánh sáng môi trường có màu xám nhạt, ảnh hưởng đều đến toàn bộ scene.

- Diffuse & Specular

Diffuse: (1.0, 1.0, 1.0, 1.0)
→ Ánh sáng khuếch tán màu trắng, phản xạ mạnh khi chiếu trực tiếp.

Specular: (1.0, 1.0, 1.0, 1.0)
→ Tạo điểm highlight sáng chói (phụ thuộc vào vật liệu).

- Spotlight Parameters (Đặc tính chùm sáng)

a) Góc cắt (Cutoff)

Giá trị: 30.0f

Dòng lệnh: glLightf(GL_LIGHT3, GL_SPOT_CUTOFF, 30.0f);

Ý nghĩa: Chùm sáng tạo thành góc 30 độ so với hướng chiếu (mặc định hướng theo trục âm Z: (0,0,-1)).

Nếu góc = 180.0, nguồn sáng trở thành omni-directional (chiếu đều mọi hướng).

b) Độ tập trung (Exponent)

Giá trị: 10.0f

Dòng lệnh: glLightf(GL_LIGHT3, GL_SPOT_EXPONENT, 10.0f);

Ý nghĩa: Điều chỉnh độ sắc nét của viền chùm sáng.

Giá trị càng cao, ánh sáng càng tập trung vào trung tâm (tạo hiệu ứng "rọi đèn").

- Hướng chiếu sáng (Spot Direction)

Mặc định: (0.0, 0.0, -1.0) (chiếu dọc theo trục Z-).

Dòng lệnh: GLfloat direction[] = {0.0f, -1.0f, 0.0f}; // Chiếu từ trần xuống (Y-)glLightfv(GL_LIGHT3, GL_SPOT_DIRECTION, direction);

2.4. Xử lý sự kiện bấm phím

a, Bật tắt nguồn sáng trong phòng

Bảng 2. 11: Bật tắt nguồn sáng trong phòng

Phím L	Bật/tắt toàn bộ hệ thống ánh sáng (bao gồm GL_LIGHT0 và GL_LIGHTING)
Hiệu ứng	Khi tắt, scene chuyển sang màu phẳng (flat shading) do tắt GL_LIGHTING. Khi bật, ánh sáng ambient từ GL_LIGHT0 được áp dụng

Code hàm xử lý sự kiện (trong WndProc):	case WM_KEYDOWN:
	if (wParam == 'L') { // Phím L
	lightingEnabled = !lightingEnabled; // Đảo trạng thái
	if (lightingEnabled) {
	glEnable(GL_LIGHTING); // Bật lighting
	glEnable(GL_LIGHT0); // Bật nguồn sáng LIGHT0
	} else {
	glDisable(GL_LIGHTING); // Tắt lighting

	glDisable(GL_LIGHT0);
	// Tắt nguồn sáng LIGHT0
	}
	}
	break;

Dưới đây là phân tích chi tiết xử lý sự kiện bấm phím để bật/tắt hệ thống ánh sáng trong OpenGL, bao gồm logic hoạt động, hiệu ứng visual, và cách triển khai code:

- Mục đích chức năng

Phím tắt (L): Điều khiển toàn bộ hệ thống ánh sáng (bao gồm GL_LIGHTING và nguồn sáng GL_LIGHT0).

Ứng dụng thực tế: Tắt ánh sáng để tiết kiệm tài nguyên hoặc tạo hiệu ứng day/night.

Bật lại ánh sáng để khôi phục shading chuẩn.

- Hiệu ứng khi bật/tắt

Bảng 2. 12: Hiệu ứng khi bật/tắt

Trạng thái	Visual Output	Nguyên nhân kỹ thuật
Khi bật (lightingEnabled = true)	Scene được chiếu sáng bởi GL_LIGHT0 (ambient + diffuse/specular).	GL_LIGHTING và GL_LIGHT0 được kích hoạt → OpenGL áp dụng lighting calculations.
Khi tắt (lightingEnabled = false)	Scene chuyển thành màu phẳng (flat shading).	Tắt GL_LIGHTING → OpenGL ngừng tính toán ánh sáng, chỉ vẽ màu sắc cơ bản của vật thể.

- Phân tích code (trong hàm WndProc)

```
case WM_KEYDOWN:
    if (wParam == 'L') {    // Nhấn phím 'L'
        lightingEnabled = !lightingEnabled;    // Đảo trạng thái (true ↔ false)
        if (lightingEnabled) {
            glEnable(GL_LIGHTING);    // Bật hệ thống lighting
```

```
            glEnable(GL_LIGHT0);        // Bật nguồn sáng LIGHT0
        } else {
            glDisable(GL_LIGHTING); // Tắt toàn bộ lighting
            glDisable(GL_LIGHT0);      // Tắt nguồn sáng LIGHT0
        }
    }
    break;
```

- Chi tiết từng bước:

Kiểm tra phím L: Sử dụng wParam để nhận diện phím được nhấn (trong trường hợp này là 'L').

Đảo trạng thái biến lightingEnabled: Biến boolean này lưu trạng thái hiện tại của hệ thống ánh sáng.

Bật/tắt lighting: glEnable(GL_LIGHTING)/glDisable(GL_LIGHTING): Bật/tắt toàn bộ tính toán ánh sáng trong OpenGL.

glEnable(GL_LIGHT0)/glDisable(GL_LIGHT0): Áp dụng riêng cho nguồn sáng GL_LIGHT0.

b, Thay đổi góc camera

Bảng 2. 13: Thay đổi góc camera

Phím 0	Góc camera tự do (có thể xoay bằng chuột).
Phím 1 đến 4: Chuyển camera đến các góc cố định	1: Camera phía trước (tâm rạp, nhìn thẳng vào màn chiếu). 2: Camera phía sau (nhìn từ sau khán đài). 3: Camera bên trái. 4: Camera bên phải.
Code hàm xử lý sự kiện (trong WndProc):	case WM_KEYDOWN: if (wParam == '0') camPreset = 0;

	// Camera tự do
	else if (wParam == '1') camPreset = 1; // Góc trước
	else if (wParam == '2') camPreset = 2; // Góc sau
	else if (wParam == '3') camPreset = 3; // Góc trái
	else if (wParam == '4') camPreset = 4; // Góc phải

Bảng 2. 14: Tổng quan chức năng

Phím	Góc camera	Mô tả
0	Tự do (Free Camera)	Người dùng có thể xoay camera bằng chuột (hoặc di chuyển tự do).
1	Front View	Camera đặt phía trước rạp, nhìn thẳng vào màn chiếu (góc mặc định).
2	Back View	Camera đặt sau khán đài, nhìn ngược về phía màn chiếu.
3	Left View	Camera đặt bên trái scene, nhìn sang phải.
4	Right View	Camera đặt bên phải scene, nhìn sang trái.

- Phân tích code xử lý sự kiện

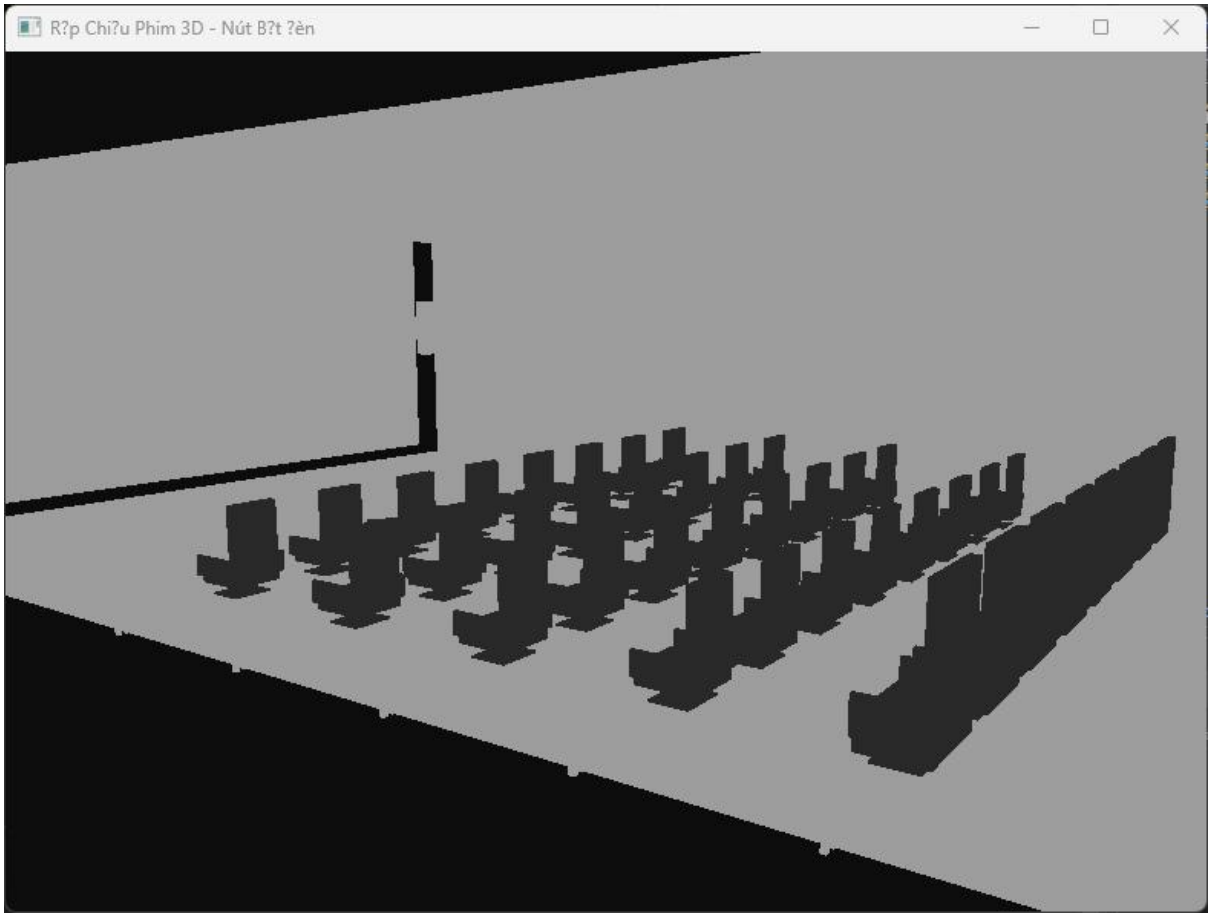
case WM_KEYDOWN:

if (wParam == '0') camPreset = 0; // Camera tự do

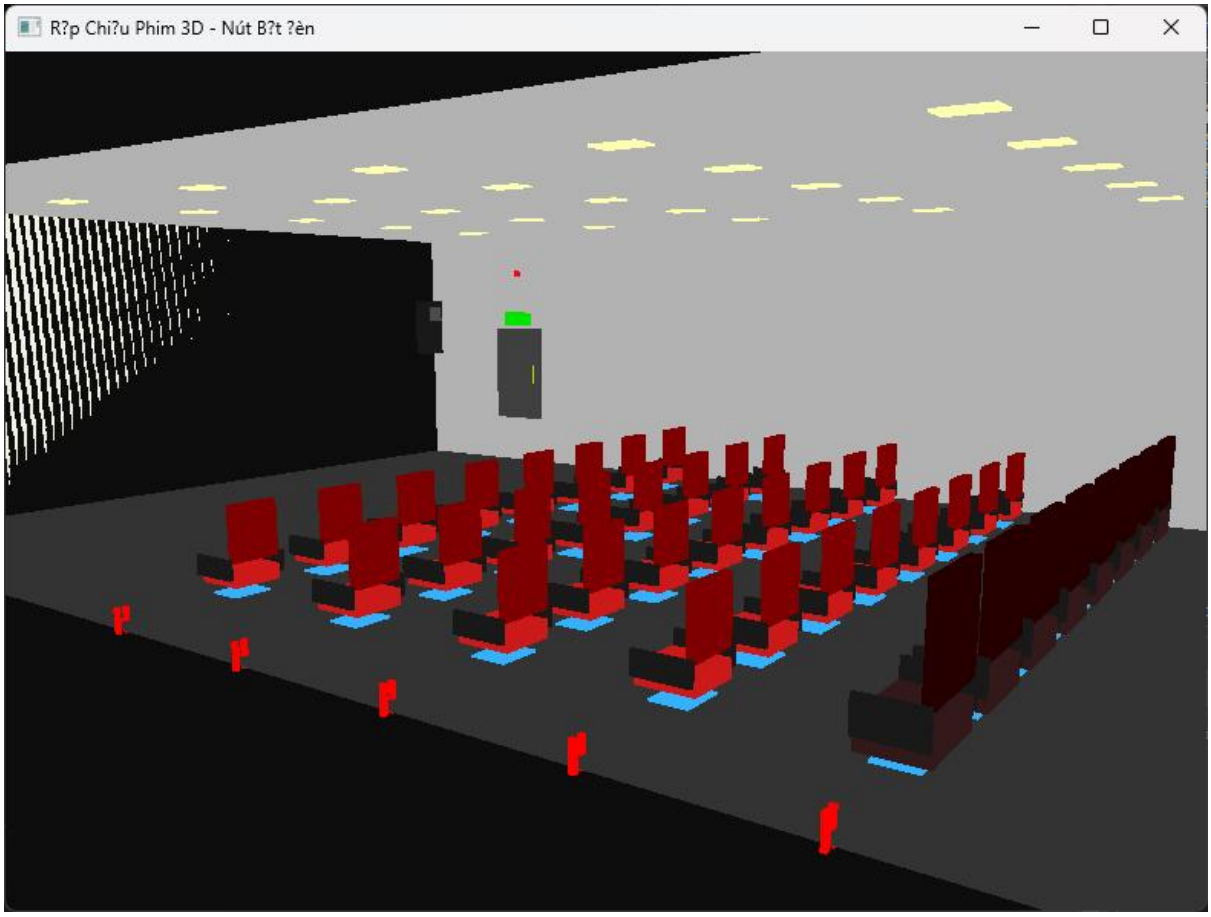
```
else if (wParam == '1') camPreset = 1;    // Góc trước
else if (wParam == '2') camPreset = 2;    // Góc sau
else if (wParam == '3') camPreset = 3;    // Góc trái
else if (wParam == '4') camPreset = 4;    // Góc phải
break;
```

Biến camPreset: Lưu trạng thái góc camera hiện tại (0: tự do, 1-4: góc cố định).

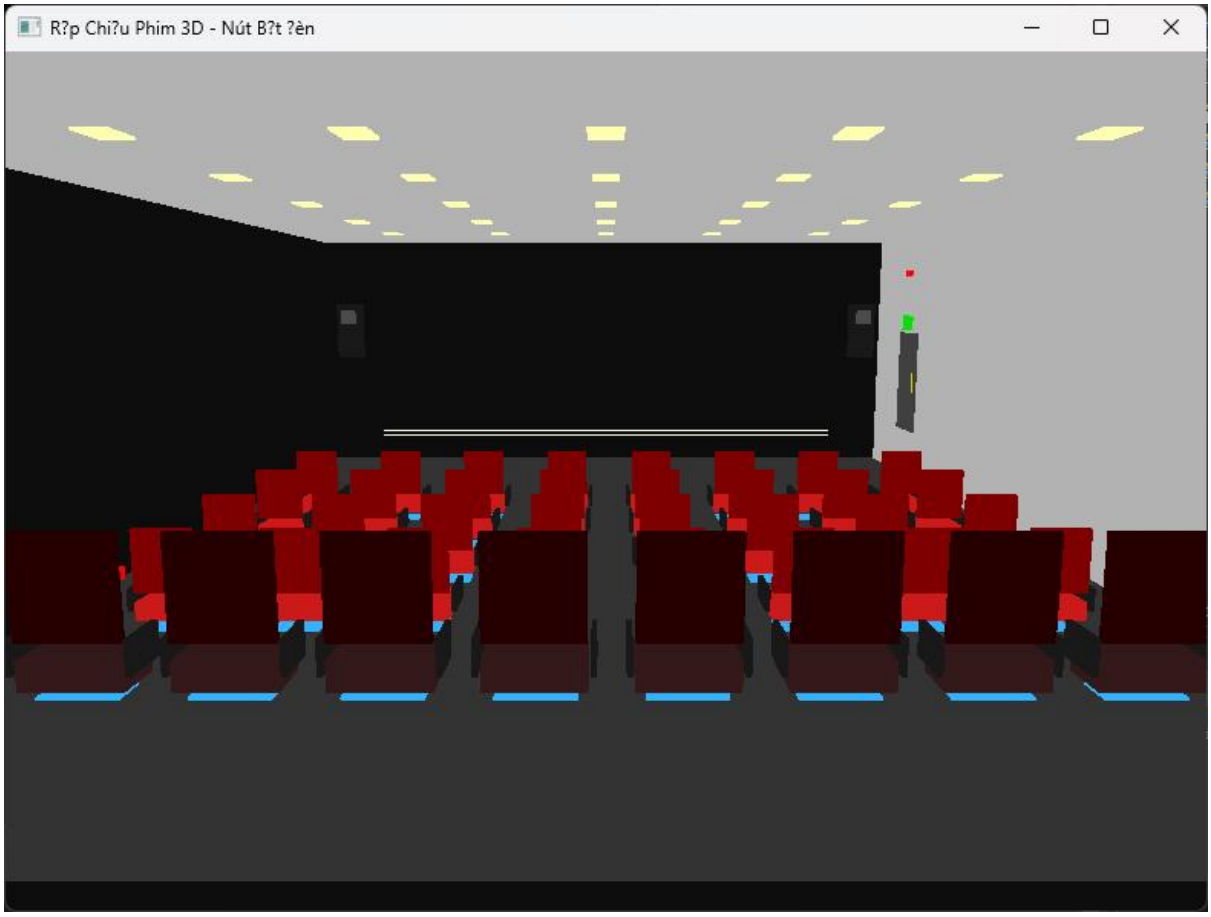
2.5. Hình ảnh thực tế



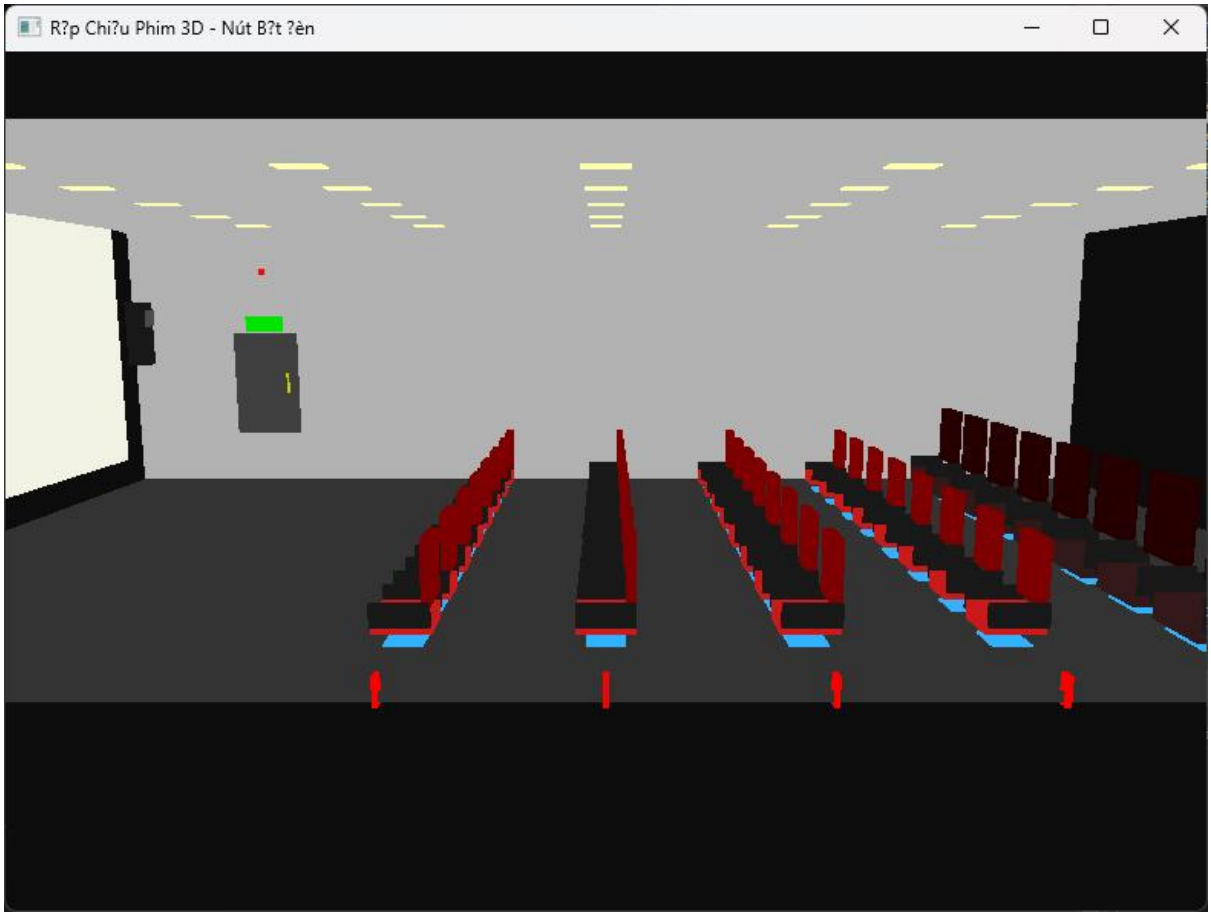
Hình 2. 8: Trạng thái tắt đèn



Hình 2. 9: Trạng thái bật đèn



Hình 2. 10: Góc camera 1



Hình 2. 11: Góc camera 2

Chương 3 : Kết luận

a, Các nội dung đã đạt được

Nhóm đã thiết kế thành công một rạp chiếu phim 3D với đầy đủ các thành phần: ghế thường, ghế VIP, màn hình chiếu, hệ thống loa, đèn trần, cửa thoát hiểm. Các tính năng tương tác bao gồm bật/tắt đèn, xoay góc camera và di chuyển để quan sát toàn bộ khung cảnh. Đồ họa được xây dựng chi tiết với ánh sáng và mô hình 3D chân thực. Nhóm đã xây dựng thành công mô hình 3D một rạp chiếu phim với các yếu tố cơ bản như không gian kiến trúc: Thiết kế layout phòng chiếu với vị trí các dãy ghế, khoảng cách tối ưu giữa ghế và màn hình

Có bố trí ghế ngồi chia làm 2 loại: ghế thường và ghế VIP (được phân biệt bằng màu sắc và kích thước).

Màn hình chiếu phim lớn ở phía trước.

Hệ thống âm thanh gồm 2 loa đặt hai bên phòng.

Đèn trần để điều chỉnh ánh sáng.

Cửa thoát hiểm được thiết kế ở bên hông phải.

- Tính năng tương tác:

Các nút bật/tắt đèn để thay đổi độ sáng phòng.

Các nút xoay camera để thay đổi góc nhìn (view).

Có thể di chuyển camera để quan sát toàn bộ rạp chiếu phim.

b, Các nội dung mong muốn cải tiến

Nhóm mong muốn nâng cấp chất lượng đồ họa (texture, hiệu ứng ánh sáng), bổ sung tính năng tương tác (chọn ghế, thay đổi phim, âm thanh) và tối ưu hiệu suất (giảm tải đa giác, tích hợp camera góc nhìn người xem). Mặc dù đã hoàn thành các yêu cầu cơ bản, nhóm nhận thấy một số điểm có thể nâng cấp trong tương lai:

- Nâng cao chất lượng đồ họa:

Thêm texture chi tiết hơn cho ghế, tường, sàn để tăng tính chân thực.

Cải thiện hiệu ứng ánh sáng (đổ bóng, phản chiếu).

- Tính năng tương tác mở rộng:

Thêm chức năng chọn ghế (click vào ghế để đặt vé).

Thêm hiệu ứng âm thanh (tiếng loa, tiếng khán giả).

Cho phép thay đổi phim trên màn hình.

- Tối ưu hóa hiệu suất:

Giảm tải đa giác (polygon) để mô hình chạy mượt hơn trên các thiết bị yếu.

Tích hợp camera first-person để trải nghiệm như khán giả thật.

c, Bài học kinh nghiệm

Qua dự án, nhóm rút ra kinh nghiệm quan trọng: lập kế hoạch rõ ràng, kiểm thử liên tục, sử dụng thư viện đồ họa hiệu quả và phối hợp nhóm chặt chẽ (dùng GitHub, giao tiếp thường xuyên). Đây là nền tảng để phát triển các dự án đồ họa phức tạp hơn trong tương lai. Qua quá trình thực hiện đồ án, nhóm rút ra một số kinh nghiệm quan trọng:

- Lập kế hoạch trước khi code:

Phân chia công việc rõ ràng (ai làm mô hình, ai làm ánh sáng, ai lập trình tương tác).

Sử dụng bản phác thảo trước khi dựng 3D để tránh sai sót.

- Kiểm thử liên tục:

Sau mỗi thành phần hoàn thành (ghế, đèn, camera), cần test ngay để phát hiện lỗi.

Sử dụng phiên bản phần mềm đồ họa ổn định để tránh crash khi render.

- Tận dụng thư viện đồ họa:

Sử dụng OpenGL/WebGL hoặc Unity/Blender để tiết kiệm thời gian thiết kế.

Tham khảo các mô hình 3D có sẵn (như ghế, loa) để tập trung vào logic chính.

- Làm việc nhóm hiệu quả:

Dùng GitHub/GitLab để quản lý code và merge thay đổi.

Giao tiếp thường xuyên để đồng bộ tiến độ.

TÀI LIỆU THAM KHẢO

[1] https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_h%E1%BB%8Da_m%C3%A1y_t%C3%ADnh

[2] https://glints.com/vn/blog/do-hoa-may-tinh-la-gi/ Người đăng: No Comments | Ngày đăng: 09/09/2023

[3] https://colorme.vn/blog/do-hoa-may-tinh-la-gi-nhung-dieu-nen-biet-ve-do-hoa-may-tinh Người đăng: Thu Trang | Ngày đăng: 2019-12-25