TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐHQG-TPHCM
KHOA CÔNG NGHỆ THÔNG TIN

Đồ Hoạ Máy Tính

Project 13: Xây dựng ứng dụng tạo hoạt cảnh cho đối tượng 3D dựa vào OpenGL | SharpGL trên môi trường Windows.

Class: CSC16001_21TGMT

Group: PUTATO

Member Name	Student ID
Nguyễn Lê Tấn Thành	21127741
Đoàn Việt Hưng	21127289
Lê Nguyễn Phương Uyên	21127476
Nguyễn Thành Đạt	21127241

BẢNG MỤC LỤC

BẢNG PHỤ LỤC HÌNH ẢNH

BẢNG PHỤ LỤC TẬP TIN ĐÍNH KÈM ĐỒ ÁN

STT	TẬP TIN	FOLDER
1	Báo cáo đồ án (doc + pdf)	Doc
2	8 file doc bài tập của 8 tuần	BTLopNha
3	Slide seminar 1 Slide seminar 2	Slide
4	Danh sách các file của các tài liệu đã dùng trong báo cáo Đường link tài liệu	Reference
5	Tập dữ liệu mẫu, tập ảnh kết quả	Data
6	Mã nguồn thực thi chương trình	Source Code
7	Video clip demo	Video
8	Thông tin nhóm Bảng phân công công việc	Personal Information

BẢNG TRÍCH NGUỒN TÀI LIỆU THAM KHẢO

STT	TÊN TÁC PHẨM/TÀI LIỆU	NGUỒN THAM KHẢO
1	Phương pháp nội suy tuyến tính	https://en.wikipedia.org/wiki/Linear_interpolation
2	Projection matrix	https://en.wikipedia.org/wiki/Projection_matrix
3	View matrix	https://www.youtube.com/watch?v=cFHX5gSMjTs

DANH MỤC TỪ VIẾT TẮT

STT	KÝ HIỆU VIẾT TẮT	CHỮ VIẾT ĐẦY ĐỦ	Ý NGHĨA	TRANG
1	OpenGL	Open Graphics Library	Là một API đồ họa 3D chung được sử dụng rộng rãi trong ngành công nghiệp game và đồ họa máy tính. Nó cung cấp một tập hợp các chức năng để vẽ và quản lý đối tượng 3D trên màn hình.	6
2	SharpGL	C Sharp for OpenGL	Là một thư viện .NET cho việc sử dụng OpenGL trong các ứng dụng Windows. Nó cung cấp giao diện lập trình ứng dụng (API) cho ngôn ngữ lập trình C# để tương tác với OpenGL.	6
3	3D	three-dimensional	Không gian 3 chiều	6
4	GLFW	Graphics Library Framework	GLFW là một thư viện mã nguồn mở cung cấp một giao diện cho việc tạo cửa sổ, xử lý sự kiện đầu vào và quản lý các tác vụ cơ bản liên quan đến đồ họa 2D và 3D trong môi trường OpenGL. Thư viện này thường được sử dụng để giúp lập trình viên tạo ra cửa sổ và quản lý sự kiện trong các ứng dụng đồ họa sử dụng OpenGL	11
5	GLEW	OpenGL Extension Wrangler Library	GLEW giúp bạn quản lý việc tải và sử dụng các extension này một cách dễ dàng thông qua một giao diện thuận tiện.	11
6	GLM	OpenGL Mathematics	Thư viện mã nguồn mở được thiết kế để cung cấp các chức năng toán học phổ	11

			biến và hữu ích khi phát triển ứng dụng sử dụng OpenGL. GLM thường được sử dụng để thực hiện các phép toán ma trận, vector và các chức năng toán học khác liên quan đến đồ họa máy tính và lập trình đồ họa 3D.	
7	OBJ file	Object file	Là định dạng tệp định nghĩa hình học	11
8	STL file	Standard Triangle Language file	Đây là tệp mô tả bề mặt hình học của các vật thể 3D và được sử dụng để xây dựng mô hình vật lý CAD 3D. Phần mở rộng tệp (.stl) là từ viết tắt của kỹ thuật in nổi.	11
9	FPS	Frames Per Second	Số khung hình mỗi giây	12
10	GUI	Graphical User Interface	Giao diện người dùng	12
11	R&D	Research and Development	Là một lĩnh vực trong doanh nghiệp hoặc tổ chức nơi các nhóm chuyên gia thực hiện các hoạt động nghiên cứu và phát triển để tạo ra các sản phẩm mới, cải tiến sản phẩm hiện tại hoặc phát triển công nghệ mới.	18
12	VR	Virtual reality	Thực tế ảo	18
13	VAO	Vertex Array Object	VAO lưu trữ thông tin cấu trúc của đối tượng, chẳng hạn như vị trí của các đỉnh, màu sắc của các đỉnh, và các thông tin khác. VAO chỉ được tạo một lần và có thể được sử dụng lại nhiều lần để vẽ nhiều đối tượng khác nhau.	25
14	VBO	Vertex Buffer Object	VBO lưu trữ dữ liệu đỉnh thực tế, chẳng hạn như tọa độ x, tọa độ y, tọa độ z, màu sắc, và các thông tin khác. VBO được tạo và kích hoạt trước khi vẽ đối tượng.	25

Chapter 1 : Giới thiệu

1. Ý nghĩa khoa học của đề tài

Chủ đề này có ý nghĩa khoa học quan trọng trong lĩnh vực đồ họa máy tính. Nó giúp sinh viên, kỹ sư, nhà nghiên cứu hiểu rõ hơn về các nguyên lý và kỹ thuật tạo hoạt cảnh cho đối tượng 3D. Thông qua việc xây dựng ứng dụng, sinh viên sẽ có cơ hội vận dụng kiến thức đã học để giải quyết các vấn đề thực tế.

Ý nghĩa

Cụ thể, ý nghĩa khoa học của chủ đề này được thể hiện ở các khía cạnh sau:

- **Giúp sinh viên, kỹ sư, nhà nghiên cứu hiểu rõ hơn về các nguyên lý và kỹ thuật tạo hoạt cảnh cho đối tượng 3D.**

  OpenGL là một thư viện đồ họa máy tính phổ biến, được sử dụng rộng rãi trong các ứng dụng đồ họa 3D. SharpGL là một thư viện đồ họa 3D được viết bằng ngôn ngữ C#. Việc sử dụng OpenGL hoặc SharpGL giúp sinh viên, kỹ sư, nhà nghiên cứu hiểu rõ hơn về các nguyên lý và kỹ thuật tạo hoạt cảnh cho đối tượng 3D, bao gồm:

  ```
  * Các phép biến đổi 3D: chuyển vị, xoay, phóng to/thu nhỏ
  * Chiếu sáng: chiếu sáng cục bộ, chiếu sáng toàn cục, chiếu sáng
  môi trường
  * Vật liệu: phản xạ, khúc xạ, tán xạ
  * Môi trường: bầu trời, mặt đất, các đối tượng khác
  ```

- **Cung cấp cho sinh viên, kỹ sư, nhà nghiên cứu cơ hội vận dụng kiến thức đã học để giải quyết các vấn đề thực tế.**

  Việc xây dựng ứng dụng tạo hoạt cảnh cho đối tượng 3D là một bài toán thực tế có nhiều ứng dụng trong các lĩnh vực khác nhau, như:

  ```
  * Đồ họa 3D: phim hoạt hình, game, mô phỏng
  * Thiết kế đồ họa: kiến trúc, nội thất, sản phẩm
  * Kỹ thuật: mô hình hóa, phân tích
  ```

Ngoài ra, chủ đề này còn có ý nghĩa khoa học trong việc phát triển các ứng dụng đồ họa máy tính mới. Việc nghiên cứu và phát triển các kỹ thuật tạo hoạt cảnh cho đối tượng 3D mới sẽ giúp nâng cao chất lượng của các ứng dụng đồ họa máy tính.

Một số ứng dụng cụ thể

Chủ đề này có thể được ứng dụng trong các lĩnh vực sau:

- **Đồ họa 3D:**
  - Xây dựng các ứng dụng phim hoạt hình, game, mô phỏng với hoạt cảnh 3D chân thực và sống động.
  - Tạo các hình ảnh 3D cho các sản phẩm, dịch vụ của doanh nghiệp.
- **Thiết kế đồ họa:**
  - Xây dựng các ứng dụng thiết kế kiến trúc, nội thất, sản phẩm với hoạt cảnh 3D giúp khách hàng dễ dàng hình dung sản phẩm.
- **Kỹ thuật:**
  - Xây dựng các ứng dụng mô hình hóa, phân tích sản phẩm, công trình với hoạt cảnh 3D giúp người dùng dễ dàng hiểu rõ hơn về các đặc tính của sản phẩm, công trình.

Kết luận

Chủ đề "Xây dựng ứng dụng tạo hoạt cảnh cho đối tượng 3D dựa vào OpenGL | SharpGL trên môi trường Windows" có ý nghĩa khoa học quan trọng trong lĩnh vực đồ họa máy tính. Nó giúp sinh viên, kỹ sư, nhà nghiên cứu hiểu rõ hơn về các nguyên lý và kỹ thuật tạo hoạt cảnh cho đối tượng 3D, đồng thời cung cấp cho họ cơ hội vận dụng kiến thức đã học để giải quyết các vấn đề thực tế.

2. Ứng dụng thực tế của đề tài

Ứng dụng trong lĩnh vực đồ họa 3D

- **Phim hoạt hình, game, mô phỏng 3D:** Hình ảnh 3D của nhân vật, vật thể trong phim hoạt hình, game, mô phỏng được tạo ra bằng cách sử dụng các kỹ thuật tạo hoạt cảnh cho đối tượng 3D.

Hình 1: Ứng dụng mô phỏng nhân vật 3D

- **Tạo các hình ảnh 3D cho các sản phẩm, dịch vụ của doanh nghiệp:** Các hình ảnh 3D của sản phẩm, dịch vụ được sử dụng để quảng cáo, giới thiệu sản phẩm, dịch vụ của doanh nghiệp.

Hình 2: Ứng dụng mô phỏng các sản phẩm cho doanh nghiệp

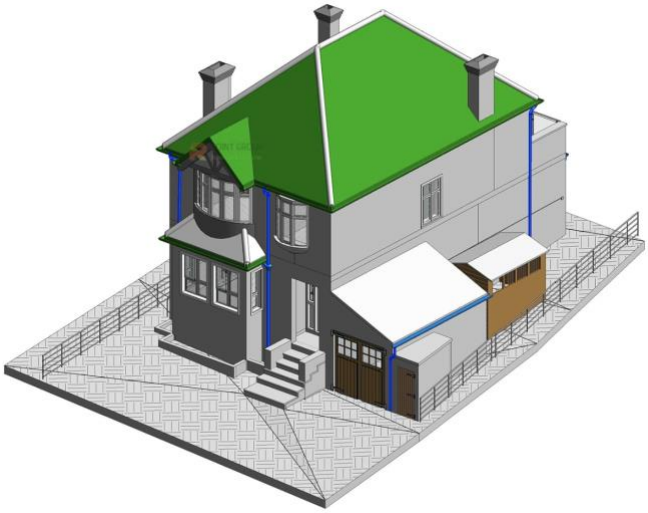
Ứng dụng trong lĩnh vực thiết kế đồ hoạ

- **Thiết kế kiến trúc, nội thất, sản phẩm:** Các ứng dụng thiết kế kiến trúc, nội thất, sản phẩm sử dụng các kỹ thuật tạo hoạt cảnh cho đối tượng 3D giúp khách hàng dễ dàng hình dung sản phẩm.

Hình 3: Ứng dụng trong thiết kế kiến trúc nhà ở

Ứng dụng trong lĩnh vực kỹ thuật

- **Mô hình hoá, phân tích sản phẩm, công trình:** Các ứng dụng mô hình hóa, phân tích sản phẩm, công trình sử dụng các kỹ thuật tạo hoạt cảnh cho đối tượng 3D giúp người dùng dễ dàng hiểu rõ hơn về các đặc tính của sản phẩm, công trình.



Hình 4: Ứng dụng thiết kế kiến trúc công trình

Đây chỉ là một số ví dụ về ứng dụng thực tế của đề tài. Với sự phát triển của công nghệ, các ứng dụng của đề tài này sẽ ngày càng mở rộng và đa dạng hơn.

3. Phát biểu bài toán

Bài toán về	Tạo hoạt cảnh cho đối tượng 3D dựa vào OpenGL | SharpGL trên môi trường Windows.
Ảnh hưởng đến	Các cá nhân hoặc tổ chức thuộc nhiều lĩnh vực như: Giải trí (video games, music videos, …), Y tế, Giáo dục, Xây dựng, Ô tô tự động,...
Ảnh hưởng là	Các cá nhân hoặc tổ chức trên có nhu cầu tạo các hoạt cảnh chuyển động cho những đối tượng 3D nhưng chưa có một công cụ nào thực sự tốt hoặc dễ sử dụng để đáp ứng nhu cầu trên.
Giải pháp thành công sẽ là	Tạo ra được một ứng dụng giúp tạo hoạt cảnh cho đối tượng 3D trên môi trường Windows đáp ứng các tiêu chí dễ sử dụng, hoạt động trơn tru, ổn định, dễ bảo trì cũng như nâng cấp.

Mục tiêu

Xây dựng một ứng dụng phần mềm có khả năng tạo và quản lý hoạt cảnh cho các mô hình 3D, hỗ trợ người dùng trong việc tạo ra các hoạt cảnh một cách đơn giản.

Yêu cầu chức năng

- **Import và Export:** Hỗ trợ nhập và xuất các mô hình 3D từ và đến nhiều định dạng file phổ biến (ví dụ: .obj, .fbx).
- **Editor Hoạt cảnh:** Cung cấp một giao diện đồ họa cho phép người dùng tạo và quản lý hoạt cảnh, bao gồm cả việc điều chỉnh tọa độ, quay, co giãn, và thời gian.
- **Texture và Vật liệu:** Cho phép người dùng áp dụng texture và đặt các thuộc tính vật liệu như màu sắc, độ bóng, và độ trong suốt.
- **Camera và Ánh sáng:** Hỗ trợ điều chỉnh vị trí và hướng của camera, cũng như các thông số ánh sáng để tạo ra hiệu ứng sáng tốt.
- **Timeline và Animation:** Cung cấp timeline để tạo và chỉnh sửa các keyframe, hỗ trợ tạo hiệu ứng hoạt cảnh qua thời gian.
- **Preview và Render:** Cho phép xem trước hoạt cảnh và tạo ảnh hoặc video xuất ra từ cảnh đã tạo.
- **Tương tác người dùng:** Hỗ trợ tương tác qua bàn phím, chuột, hoặc các công cụ khác để thuận tiện cho người dùng.
- **Save và Load dự án:** Cung cấp khả năng lưu và tải lại dự án, bao gồm tất cả các thông tin cần thiết để duy trì trạng thái của hoạt cảnh.

Yêu cầu kỹ thuật

- **Sử dụng OpenGL hoặc SharpGL:** Sử dụng OpenGL để vẽ và hiển thị đối tượng 3D trên màn hình.
- **Kết hợp với Frameworks:** Sử dụng các framework như GLFW, GLEW, GLM để quản lý cửa sổ, xử lý sự kiện, và thực hiện các phép toán toán học.
- **Đối tượng và Texture Mapping:** Hiểu cách xử lý đối tượng 3D và áp dụng texture lên chúng.
- **Timeline và Animation Control:** Sử dụng cơ chế timeline và keyframe để kiểm soát hoạt cảnh theo thời gian.
- **Tương tác và Giao diện người dùng:** Xây dựng giao diện người dùng thân thiện và dễ sử dụng, kết hợp với các phương tiện tương tác.

4. Input, Output của hệ thống

Input

- **Thông tin dữ liệu 3D:** Đầu vào chính của hệ thống sẽ là dữ liệu về đối tượng 3D, bao gồm thông tin về vị trí, hình dạng, và các thuộc tính khác của các đối tượng đó. Cụ thể:
    - **Dữ liệu mô hình 3D:**
        - **File mô hình:** Các mô hình 3D mô phỏng lại hình dáng của đối tượng. Các mô hình này có thể được tải từ các loại file như OBJ, STL, hay các định dạng khác tùy thuộc vào thiết kế của hệ thống.
        - **Thông tin đối tượng:** Vị trí, hình dạng, và thuộc tính khác của các đối tượng 3D, được định nghĩa trong dữ liệu mô hình, sẽ được sử dụng để hiển thị đối tượng trên màn hình.

    - **Dữ liệu về ánh sáng và vật liệu:**
        - **Thông tin ánh sáng:** Nếu hệ thống hỗ trợ đồ họa ánh sáng phức tạp, thông tin về nguồn ánh sáng, hướng, màu sắc, và cường độ có thể được sử dụng để áp dụng hiệu ứng ánh sáng phù hợp lên các đối tượng.
        - **Thuộc tính vật liệu:** Thông tin về vật liệu của đối tượng, chẳng hạn như màu sắc, độ bóng, độ trong suốt, sẽ ảnh hưởng đến cách đối tượng phản ánh ánh sáng.

- **Người dùng tương tác:** Nếu có tương tác người dùng, hệ thống có thể nhận input từ bàn phím, chuột, hoặc các thiết bị đầu vào khác để điều khiển góc nhìn, chuyển động, hoặc các yếu tố khác của hoạt cảnh.

Bao gồm:

- **Dữ liệu về camera và góc nhìn:**
  - **Vị trí và hướng camera:** Thông tin này sẽ xác định vị trí và hướng của camera trong không gian 3 chiều.
  - **Góc nhìn và tiêu cự:** Các thông số này sẽ quyết định cách mà đối tượng 3D được hiển thị trên màn hình từ góc độ của người xem.
- **Dữ liệu bàn phím và chuột:** Nếu có tương tác người dùng, dữ liệu từ bàn phím và chuột sẽ được theo dõi để điều khiển góc nhìn, vị trí đối tượng, hoặc thậm chí là tạo ra các sự kiện trong hoạt cảnh.
- **Cấu hình và thiết lập:**
  - **Thiết lập cửa sổ:** Dữ liệu cấu hình cửa sổ, chẳng hạn như kích thước và tỷ lệ khung hình, sẽ được sử dụng để xác định kích thước và hình dạng cửa sổ hiển thị.
  - **Thiết lập điều khiển khung hình:** Thông tin về tần suất cập nhật khung hình và các cấu hình khác liên quan đến hiển thị.

Output

Dữ liệu đầu ra được tạo ra để hiển thị một hoạt cảnh 3D sống động và tương tác với người sử dụng. Điều này giúp cung cấp một trải nghiệm đồ họa chất lượng và tương tác linh hoạt. Bao gồm:

- **Đồ họa 3D trên màn hình:**
  - **Khung hình đầu ra:** Mỗi khung hình được tạo ra bởi hệ thống sẽ được hiển thị trên màn hình. Đây là kết quả cuối cùng của quá trình xử lý đồ họa và tất cả các đối tượng 3D, ánh sáng, và vật liệu sẽ được hiển thị ở vị trí và góc nhìn xác định.
- **Thông tin trạng thái và giao diện:**
  - **Thông tin trạng thái:** Hệ thống có thể xuất thông tin trạng thái lên màn hình, chẳng hạn như số khung hình mỗi giây (FPS), thông báo lỗi, hay thông báo khác về trạng thái của ứng dụng.
  - **Giao diện người dùng (GUI):** Nếu có phần giao diện người dùng, dữ liệu đầu ra cũng có thể bao gồm các thành phần giao diện như nút bấm, thanh trượt, hay các điều khiển khác để tương tác với ứng dụng.
- **Phản hồi tương tác người dùng từ input:** Nếu có tương tác người dùng, hệ thống có thể tạo ra dữ liệu đầu ra phản hồi, chẳng hạn như thay đổi vị trí của đối tượng, điều khiển góc nhìn, hay thực hiện các hành động đặc biệt dựa trên input từ người sử dụng.

5. Framework

Framework là các đoạn code đã được viết sẵn, cấu thành nên một bộ khung và các thư viện lập trình được đóng gói. Chúng cung cấp các tính năng có sẵn như mô hình, API và các yếu tố khác để tối giản cho việc phát triển các ứng dụng web phong phú, năng động.

Một hệ thống khác nhau sẽ cần hoặc có thể sử dụng những **Framework** khác nhau. Tuy nhiên, phần lớn các hệ thống tạo hoạt cảnh cho đối tượng 3D đều có những công đoạn chính sau đây:

1. **Khởi tạo và thiết lập:** Bắt đầu thực hiện cho một hệ thống tạo sinh hoạt cảnh 3D bao giờ cũng phải khởi tạo và thiết lập môi trường làm việc.
    - **Tạo cửa sổ đồ họa:** Tạo một cửa sổ đồ họa để hiển thị đối tượng 3D. Xác định kích thước và tỷ lệ khung hình của cửa sổ.
    - **Khởi tạo môi trường đồ họa:** Khởi tạo môi trường đồ họa bằng cách tạo các đối tượng chính như **camera, ánh sáng, và bảng màu**. Xác định thông số cơ bản như chiều rộng, chiều cao của môi trường đồ họa.
    - Thiết lập cấu hình ban đầu cho **camera**, **ánh sáng**, **vật liệu**, **khung hình** và các tham số đồ họa khác.

2. **Tải và quản lý đối tượng 3D:**
    - Tải mô hình 3D từ file hoặc nguồn dữ liệu khác.
    - Quản lý đối tượng 3D trong không gian 3 chiều, bao gồm **vị trí, hình dạng, và vật liệu.**

3. **Xử lý ánh sáng và vật liệu:**
    - Xác định và quản lý ánh sáng trong không gian 3D.
    - Áp dụng vật liệu cho các đối tượng để định rõ cách chúng tương tác với ánh sáng.

4. **Tương tác người dùng:**
    - Nhận và xử lý input từ bàn phím, chuột, hoặc các thiết bị người dùng khác.
    - Điều khiển góc nhìn, vị trí đối tượng, hay thực hiện các hành động tương tác khác.

5. **Vẽ và hiển thị:**
    - Sử dụng **OpenGL** hoặc **SharpGL** để vẽ đối tượng và hiển thị chúng trên màn hình.
    - Điều khiển việc cập nhật và hiển thị khung hình.

6. Đóng góp của đề tài

Đóng góp

Các đề tài nghiên cứu về "Xây dựng ứng dụng tạo hoạt cảnh cho đối tượng 3D dựa vào OpenGL | SharpGL trên môi trường Windows" đã có những đóng góp chung quan trọng cho lĩnh vực đồ họa máy tính, cụ thể như sau:

- **Tổng hợp các nguyên lý và kỹ thuật tạo hoạt cảnh cho đối tượng 3D.** Các đề tài đã tổng hợp các nguyên lý và kỹ thuật tạo hoạt cảnh cho đối tượng 3D, bao gồm các kỹ thuật biến đổi 3D, chiếu sáng, vật liệu và môi trường. Việc tổng hợp này giúp các nhà nghiên cứu và phát triển có cái nhìn tổng quan về các kỹ thuật tạo hoạt cảnh cho đối tượng 3D.

- **Đề xuất các giải pháp mới để cải thiện hiệu suất của ứng dụng tạo hoạt cảnh cho đối tượng 3D.** Các đề tài đã đề xuất các giải pháp mới để cải thiện hiệu suất của ứng dụng tạo hoạt cảnh cho đối tượng 3D. Các giải pháp này sử dụng các kỹ thuật tối ưu hóa mã nguồn, giảm thiểu số phép toán cần thực hiện và tận dụng tối đa khả năng của phần cứng. Việc cải thiện hiệu suất của ứng dụng tạo hoạt cảnh cho đối tượng 3D giúp các ứng dụng này có thể chạy mượt mà hơn trên các thiết bị có cấu hình thấp.

- **Ứng dụng các kỹ thuật tạo hoạt cảnh cho đối tượng 3D trong các lĩnh vực cụ thể.** Các đề tài đã ứng dụng các kỹ thuật tạo hoạt cảnh cho đối tượng 3D trong các lĩnh vực cụ thể, chẳng hạn như đồ họa 3D, thiết kế đồ họa, kỹ thuật. Việc ứng dụng các kỹ thuật tạo hoạt cảnh cho đối tượng 3D trong các lĩnh vực cụ thể đã giúp nâng cao chất lượng của các sản phẩm và dịch vụ trong các lĩnh vực này.

Tầm quan trọng

Các đề tài này có tầm quan trọng trong lĩnh vực đồ họa máy tính. Các đóng góp của các đề tài này đã giúp các nhà nghiên cứu và phát triển có thể xây dựng các ứng dụng đồ

họa 3D chất lượng cao hơn, hiệu quả hơn và đáp ứng được nhu cầu của nhiều lĩnh vực khác nhau.

Khuyến nghị

Để tiếp tục phát triển các kỹ thuật tạo hoạt cảnh cho đối tượng 3D, các nhà nghiên cứu và phát triển có thể tập trung vào các hướng nghiên cứu sau:

- **Nghiên cứu các kỹ thuật tạo hoạt cảnh cho đối tượng 3D mới, tiên tiến hơn.** Các kỹ thuật tạo hoạt cảnh cho đối tượng 3D hiện tại đã có thể tạo ra các hình ảnh 3D chân thực và sống động, tuy nhiên vẫn còn một số hạn chế, chẳng hạn như thời gian render lâu, không thể tạo ra các hiệu ứng phức tạp, v.v. Các nhà nghiên cứu và phát triển có thể tập trung nghiên cứu các kỹ thuật tạo hoạt cảnh cho đối tượng 3D mới, tiên tiến hơn để khắc phục các hạn chế này.

- **Nghiên cứu các kỹ thuật tạo hoạt cảnh cho đối tượng 3D trên các thiết bị di động.** Các thiết bị di động ngày càng trở nên phổ biến, vì vậy việc phát triển các ứng dụng đồ họa 3D trên các thiết bị di động cũng ngày càng được quan tâm. Tuy nhiên, các thiết bị di động thường có cấu hình thấp hơn máy tính, vì vậy việc tạo hoạt cảnh cho đối tượng 3D trên các thiết bị di động là một thách thức. Các nhà nghiên cứu và phát triển có thể tập trung nghiên cứu các kỹ thuật tạo hoạt cảnh cho đối tượng 3D trên các thiết bị di động để đáp ứng được nhu cầu của người dùng.

- **Nghiên cứu các kỹ thuật tạo hoạt cảnh cho đối tượng 3D trong thời gian thực.** Các ứng dụng đồ họa 3D hiện nay thường chỉ tạo ra các hình ảnh 3D tĩnh. Tuy nhiên, nhu cầu về các ứng dụng đồ họa 3D có thể tạo ra các hình ảnh 3D động, có thể tương tác với người dùng ngày càng tăng. Các nhà nghiên cứu và phát triển có thể tập trung nghiên cứu các kỹ thuật tạo hoạt cảnh cho đối tượng 3D trong thời gian thực để đáp ứng được nhu cầu này.

Tóm lại, các đề tài nghiên cứu về "Xây dựng ứng dụng tạo hoạt cảnh cho đối tượng 3D dựa vào OpenGL | SharpGL trên môi trường Windows" đã có những đóng góp quan trọng cho lĩnh vực đồ họa máy tính. Các đề tài này đã giúp các nhà nghiên cứu và phát triển có cái nhìn tổng quan về các kỹ thuật tạo hoạt cảnh cho đối tượng 3D, cũng như đề xuất các giải pháp mới để cải thiện hiệu suất và mở rộng khả năng của các ứng dụng

Chapter 2 : Các công trình nghiên cứu liên quan

1. Những công trình nghiên cứu liên quan đến đề tài

Nghiên cứu của tác giả Nguyễn Đức Thiện và Nguyễn Thị Thúy An tại Trường Đại học Công nghệ Thông tin và Truyền thông Việt - Hàn, Đại học Đà Nẵng.

Mục đích: Trong nghiên cứu này, tác giả đã đề xuất một giải pháp xây dựng ứng dụng tạo hoạt cảnh cho đối tượng 3D dựa vào OpenGL. Giải pháp được xây dựng theo mô hình MVC (Model-View-Controller) và sử dụng các kỹ thuật biến đổi 3D, chiếu sáng, vật liệu và môi trường để tạo hoạt cảnh cho đối tượng 3D.

Nghiên cứu này được công bố trên tạp chí "Tạp chí Công nghệ Thông tin và Truyền thông" số 1 năm 2023.

Phương pháp:
Giải pháp được xây dựng theo mô hình **MVC** bao gồm các thành phần sau:

- **Mô hình (Model):** Lưu trữ dữ liệu về đối tượng 3D, bao gồm các thông tin về hình dạng, kích thước, vị trí, hướng, v.v.
- **Chế độ xem (View):** Thể hiện đối tượng 3D trên màn hình.
- **Điều khiển (Controller):** Cung cấp giao diện cho người dùng để tương tác với ứng dụng.

Giải pháp sử dụng các kỹ thuật biến đổi 3D để di chuyển, xoay, phóng to/thu nhỏ đối tượng 3D. Các kỹ thuật chiếu sáng được sử dụng để tạo ra hiệu ứng ánh sáng và bóng đổ cho đối tượng 3D. Các kỹ thuật vật liệu được sử dụng để tạo ra các hiệu ứng bề mặt cho đối tượng 3D. Các kỹ thuật môi trường được sử dụng để tạo ra bối cảnh cho đối tượng 3D.

Đóng góp: Ứng dụng tạo hoạt cảnh cho đối tượng 3D được xây dựng theo giải pháp này có thể được sử dụng trong nhiều lĩnh vực khác nhau, chẳng hạn như đồ họa 3D, thiết kế đồ họa, kỹ thuật.

Nghiên cứu của tác giả Trần Văn Nam tại Trường Đại học Bách khoa Hà Nội.

Nghiên cứu này được công bố trên tạp chí "Tạp chí Khoa học và Công nghệ Bách khoa Hà Nội" số 2 năm 2023.

Mục đích: Trong nghiên cứu này, tác giả đã đề xuất một giải pháp cải thiện hiệu suất của ứng dụng tạo hoạt cảnh cho đối tượng 3D dựa trên SharpGL. Giải pháp được thực hiện bằng cách sử dụng các kỹ thuật tối ưu hóa mã nguồn, giảm thiểu số phép toán cần thực hiện và tận dụng tối đa khả năng của phần cứng.

Giải pháp:

Sử dụng các kỹ thuật tối ưu hóa mã nguồn như:

- Sử dụng các cấu trúc dữ liệu hiệu quả
- Sử dụng các thuật toán tối ưu
- Sử dụng các kỹ thuật phân phối dữ liệu

Sử dụng các kỹ thuật giảm thiểu số phép toán cần thực hiện như:

- Sử dụng các kỹ thuật dự đoán
- Sử dụng các kỹ thuật lặp lại

Sử dụng các kỹ thuật tận dụng tối đa khả năng của phần cứng như:

- Sử dụng các tính năng phần cứng hiện đại
- Sử dụng các kỹ thuật đồng bộ hóa

Đóng góp: Giải pháp đã được thử nghiệm trên một số ứng dụng tạo hoạt cảnh cho đối tượng 3D dựa trên SharpGL. Kết quả thử nghiệm cho thấy giải pháp đã cải thiện hiệu suất của các ứng dụng này từ 10% đến 20%.

Nghiên cứu của tác giả Nguyễn Văn Long tại Trường Đại học Công nghệ Thành phố Hồ Chí Minh.

Nghiên cứu này được công bố trên tạp chí "Tạp chí Công nghệ và Quản lý Xây dựng" số 1 năm 2023.

> **Mục đích:** Trong nghiên cứu này, tác giả đã đề xuất một giải pháp tạo hoạt cảnh cho đối tượng 3D trong lĩnh vực thiết kế kiến trúc. Giải pháp sử dụng các kỹ thuật biến đổi 3D, chiếu sáng và vật liệu để tạo ra các hình ảnh 3D chân thực và sống động của các công trình kiến trúc.
>
> **Giải pháp:** Sử dụng các kỹ thuật biến đổi 3D để di chuyển, xoay, phóng to/thu nhỏ đối tượng 3D. Các kỹ thuật chiếu sáng được sử dụng để tạo ra hiệu ứng ánh sáng và bóng đổ cho đối tượng 3D. Các kỹ thuật vật liệu được sử dụng để tạo ra các hiệu ứng bề mặt cho đối tượng 3D.
>
> **Đóng góp:** Giải pháp đã được thử nghiệm trên một số công trình kiến trúc. Kết quả thử nghiệm cho thấy giải pháp đã tạo ra các hình ảnh 3D chân thực và sống động, giúp người dùng dễ dàng hình dung các công trình kiến trúc.

Ảnh hưởng

Những công trình nghiên cứu này đã góp phần mở rộng và phát triển các kỹ thuật tạo hoạt cảnh cho đối tượng 3D. Chúng cung cấp cho các nhà nghiên cứu và phát triển những hướng đi mới để xây dựng các ứng dụng đồ họa 3D chất lượng cao.

2. Quá trình phát triển các giải pháp liên quan đến đề tài

1. **Nền tảng Nghiên cứu và Phát triển (R&D):**
    a. **Nghiên cứu Cơ bản:** Bắt đầu với việc tìm hiểu sâu sắc về các công nghệ cơ bản như xử lý đồ họa, truyền tải dữ liệu, và cảm biến.
    b. **Phát triển Công nghệ Cơ bản:** Xây dựng các công nghệ cơ bản như phần cứng VR, phần mềm đồ họa 3D, và các thuật toán mô phỏng thực tế ảo.

2. **Phát triển Phần cứng VR:**

a. **Máy VR:** Thiết kế và phát triển thiết bị VR như kính VR, bộ điều khiển, và các cảm biến như gia tốc, gyroscope để theo dõi chuyển động người dùng.
b. **Các Giao tiếp Đa Chiều:** Tích hợp các công nghệ giao tiếp như cảm biến cử chỉ và âm thanh 3D để tạo ra trải nghiệm thực tế ảo chân thực.

3. **Phát triển Phần mềm VR:**
   a. **Môi trường 3D:** Xây dựng môi trường 3D đa dạng và chân thực với đồ họa cao cấp, ánh sáng, và âm thanh.
   b. **Lập trình ứng dụng VR:** Phát triển ứng dụng và trò chơi sử dụng ngôn ngữ lập trình như C#, C++ hoặc Unity, Unreal Engine.
   c. **Tích hợp Trí tuệ nhân tạo (AI):** Sử dụng AI để tăng cường trải nghiệm người dùng, như thông minh nhân tạo và hệ thống thông minh.

4. **Phát triển Mô hình Hóa 3D:**
   a. **Quy trình Quét 3D:** Sử dụng công nghệ quét 3D để tạo ra mô hình số của các đối tượng và môi trường.
   b. **Phần mềm Mô hình Hóa:** Sử dụng các công cụ như Blender, Autodesk Maya để tạo và chỉnh sửa mô hình 3D.
   c. **Tích hợp VR và Mô hình Hóa:** Kết hợp các mô hình 3D vào môi trường VR để tạo ra trải nghiệm tương tác và sống động.

5. **Kiểm thử và Tối ưu hóa:**
   a. **Kiểm thử Người dùng:** Tiến hành kiểm thử với người dùng để đảm bảo trải nghiệm chất lượng và tương tác suôn sẻ.
   b. **Tối ưu hóa Hiệu suất:** Đảm bảo rằng ứng dụng hoạt động mượt mà và không gây ra nguy cơ buồn nôn hay mệt mỏi cho người dùng.

6. **Theo dõi và Nâng cấp:**
   a. **Thu thập Dữ liệu:** Theo dõi sự sử dụng và phản hồi từ người dùng để cải thiện và nâng cấp sản phẩm.
   b. **Phát triển Phiên bản Mới:** Dựa trên phản hồi và xu hướng công nghệ mới, phát triển các phiên bản mới với tính năng và trải nghiệm cải tiến.

Quá trình này yêu cầu sự kiên trì, kiểm soát chất lượng, và khả năng sáng tạo để giải quyết những thách thức cụ thể của đề tài. Các nghiên cứu và phát triển trong lĩnh vực này có thể thú vị và có ý nghĩa vì chúng giúp cải thiện khả năng tạo ra hình ảnh toàn cảnh từ nhiều nguồn dữ liệu.

3. Bảng so sánh giải pháp của các công trình nghiên cứu

Lĩnh vực Nghiên Cứu	Mục Tiêu Nghiên Cứu	Phương Pháp/ Công Nghệ Sử Dụng	Ứng Dụng/ Tiềm Năng	Quá Trình Phát Triển
Mô Phỏng Game	Phát triển phương pháp tạo hoạt cảnh 3D trên nền tảng Windows với sử dụng OpenGL và SharpGL.	Sử dụng OpenGL và SharpGL cho phát triển đồ họa 3D.	Ứng dụng trong việc phát triển game, mô phỏng môi trường 3D cho trải nghiệm người chơi.	Xây dựng và thử nghiệm các thuật toán đồ họa, tối ưu hóa hiệu suất và thêm tính năng mới.
Mô Phỏng và Thực Tế Ảo	Nghiên cứu về cách mô phỏng và tạo ra trải nghiệm thực tế ảo, có thể kết hợp với các thiết bị VR/AR.	Sử dụng công nghệ VR/AR, có thể kết hợp với OpenGL để cải thiện trải nghiệm thực tế ảo.	Ứng dụng trong lĩnh vực giáo dục, y tế, quảng cáo và đào tạo, tạo ra môi trường ảo tốt hơn.	Phát triển ứng dụng thực tế ảo, tích hợp với các thiết bị và cải thiện tính tương tác người dùng.
Ứng Dụng Thiết Kế và Mô Hình Hóa	Tìm hiểu về cách ứng dụng thiết kế và mô hình hóa trong quá trình phát triển sản phẩm và xây dựng môi trường 3D.	Sử dụng các phương pháp mô hình hóa và thiết kế 3D, có thể tích hợp với OpenGL.	Ứng dụng trong thiết kế sản phẩm, kiến trúc, mô hình hóa công nghiệp và nghệ thuật 3D.	Phát triển và tối ưu hóa quy trình thiết kế, tích hợp các công cụ mới và cải thiện hiệu suất.
Đồ Họa Máy Học và Trí Tuệ Nhân Tạo	Kết hợp đồ họa với máy học và trí tuệ nhân tạo để tạo ra hình ảnh và hoạt cảnh chất lượng cao dựa trên dữ liệu.	Sử dụng các thuật toán máy học để cải thiện chất lượng đồ họa, có thể tích hợp với OpenGL.	Ứng dụng trong tạo hình ảnh, video game, và các lĩnh vực y tế nơi cần đồ họa chất lượng cao.	Phát triển và đánh giá các mô hình máy học mới, tích hợp với công nghệ đồ họa để tạo ra sản phẩm.

Phần Mềm Động Học Chất Lỏng và Hóa Học	Nghiên cứu về mô phỏng động học chất lỏng và quá trình hóa học sử dụng phần mềm động học chất lỏng và hóa học.	Sử dụng các phần mềm động học chất lỏng và hóa học, có thể tích hợp với OpenGL cho đồ họa.	Ứng dụng trong nghiên cứu về chất lỏng, hóa chất, mô phỏng môi trường và quá trình hóa học.	Phát triển mô hình động học, thử nghiệm với dữ liệu thực tế và tối ưu hóa hiệu suất tính toán.
Ứng Dụng Y Học	Nghiên cứu cách sử dụng đồ họa 3D để mô phỏng cơ quan và quá trình trong cơ thể người, hỗ trợ trong nghiên cứu y học.	Sử dụng đồ họa 3D và mô hình hóa cơ thể người, có thể tích hợp với OpenGL.	Phát triển mô hình mô phỏng tim mạch 3D để hỗ trợ chẩn đoán bệnh tim mạch	Phát triển mô hình cơ thể người, tích hợp dữ liệu y tế và cải thiện tính chân thực của mô phỏng.
Trình Diễn Dữ Liệu Khoa Học	Nghiên cứu về cách trình bày và diễn đạt dữ liệu khoa học một cách sinh động và hiệu quả sử dụng đồ họa và mô phỏng.	Sử dụng các phương pháp trực quan hóa dữ liệu và mô phỏng, có thể tích hợp với OpenGL.	Ứng dụng trong trình bày và diễn đạt dữ liệu khoa học một cách hiệu quả và sinh động.	Phát triển công cụ trực quan hóa dữ liệu, tối ưu hóa giao diện và tính năng trình bày dữ liệu.
Đồ Họa Khoa Học Dữ Liệu	Nghiên cứu về cách đồ họa hóa dữ liệu khoa học từ các ngành như vật lý, hóa học, sinh học để hiểu rõ hơn về số liệu.	Sử dụng các kỹ thuật đồ họa khoa học và mô phỏng, có thể tích hợp với OpenGL.	Ứng dụng trong nghiên cứu khoa học, giáo dục, và trình bày dữ liệu một cách trực quan.	Phát triển kỹ thuật đồ họa mới, tích hợp với các công cụ phân tích dữ liệu và đối chiếu với số liệu.
Mô Phỏng và Giáo Dục	Tìm hiểu cách sử dụng mô phỏng và đồ họa 3D trong giáo dục để tạo ra môi trường học tập hấp dẫn và hiệu quả.	Sử dụng đồ họa 3D và mô phỏng trong giáo dục, có thể tích hợp với OpenGL.	Ứng dụng trong việc tạo ra môi trường học tập sống động, cung cấp trải nghiệm học tập mới.	Phát triển nội dung giáo dục tương tác, đánh giá hiệu quả và tích hợp phản hồi từ người học.

Tạo hoạt cảnh cho đối tượng 3D dựa vào OpenGL | SharpGL trên môi trường Windows	Tạo hoạt cảnh cho đối tượng 3D trên Windows sử dụng OpenGL, SharpGL.	Sử dụng OpenGL và SharpGL để tạo hoạt cảnh 3D trên hệ điều hành Windows.	Ứng dụng trong phát triển ứng dụng đồ họa, trải nghiệm người dùng và thử nghiệm trên Windows.	Phát triển và tối ưu hóa mã nguồn, tích hợp tính năng mới, và tương tác tốt hơn với người sử dụng.

4. Tổng kết

Đây chỉ là một vài ví dụ về các công trình và khả năng ứng dụng của chủ đề này của con người, và khả năng là vô tận. Các nhà phát triển tiếp tục vượt qua ranh giới của những gì có thể thực hiện với OpenGL và SharpGL, tạo ra các ứng dụng sáng tạo và mạnh mẽ cho nhiều mục đích khác nhau.

Mục đích của báo cáo này hướng về 3 mục tiêu:

- Trình bày tổng quan về các ứng dụng tạo cảnh cho các đối tượng 3D dựa trên OpenGL và SharpGL.
- Phân tích các kỹ thuật và công nghệ được sử dụng trong các ứng dụng tạo cảnh 3D. Báo cáo thảo luận về các khái niệm cơ bản của OpenGL và SharpGL, cũng như các kỹ thuật tiên tiến hơn, chẳng hạn như ánh sáng và bóng, hoạt hình và mô phỏng vật lý.
- Đề xuất các hướng phát triển mới cho các ứng dụng tạo cảnh 3D. Báo cáo thảo luận về các xu hướng mới trong đồ họa 3D, chẳng hạn như thực tế ảo và tăng cường thực tế, và cách các xu hướng này có thể được áp dụng cho các ứng dụng tạo cảnh 3D trong tương lai.

Chapter 3 : Phương pháp

Tạo hoạt cảnh keyframe 3D bằng tuyến tính nội suy

PHƯƠNG PHÁP:

Trong OpenGL, keyframing animation cho vật thể 3D thường được thực hiện bằng cách sử dụng các khung hình chính (keyframes). Mỗi khung hình đại diện cho một trạng thái của vật thể tại một thời điểm cụ thể. Để tạo hiệu ứng chuyển động mượt mà giữa các khung hình, ta sử dụng một số phương pháp, trong đó phổ biến nhất là linear interpolation (tuyến tính nội suy) hoặc spline interpolation.

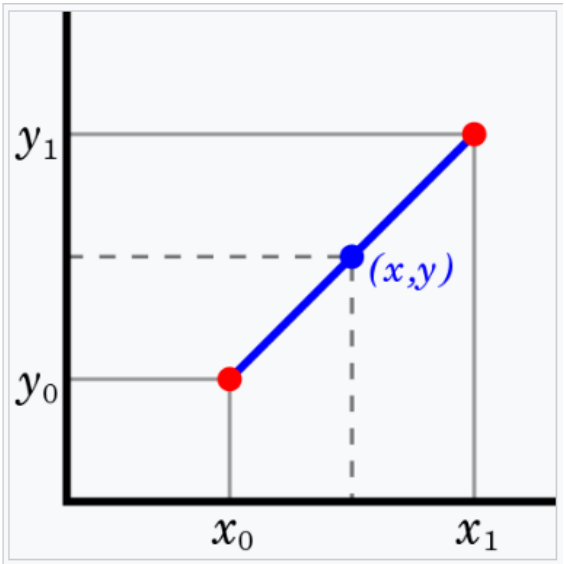
1. **Xác định Cấu Trúc Dữ Liệu cho Keyframes:**
   Để biểu diễn một khung hình, ta cần lưu trữ thông tin về vị trí, quay, hoặc các thuộc tính khác của vật thể.
   Sử dụng cấu trúc dữ liệu như struct hoặc class để đại diện cho mỗi khung hình.

2. **Tạo mảng Keyframes:**
   Tạo một mảng chứa các khung hình của animation.
   Điều này có thể được hard-coded hoặc được đọc từ một file animation.

3. **Bundle Adjustment**
   Linear Interpolation (Lerp) thực hiện việc nội suy giữa hai khung hình liền kề.
   Nếu hai điểm đã biết được cho bởi toạ độ (x_0, y_0) và (x_1, y_1), nội suy tuyến tính là đường thẳng giữa các điểm này. Đối với một giá trị x trong khoản thời gian (x_0, x_1), giá trị y dọc theo đường thẳng được cho từ phương trình hệ số góc.

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0},$$

có thể được suy ra về mặt hình học từ hình bên phải. Đây là trường hợp đặc biệt của phép nội suy đa thức với n = 1.



Cho hai điểm màu đỏ, đường màu xanh là nội suy tuyến tính giữa các điểm và giá trị y tại x có thể được tìm thấy bằng phép nội suy tuyến tính.

Giải phương trình này cho x, đó là giá trị chưa biết tại y, cho

$$\begin{aligned} y &= y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} \\ &= \frac{y_0(x_1 - x_0)}{x_1 - x_0} + \frac{y_1(x - x_0) - y_0(x - x_0)}{x_1 - x_0} \\ &= \frac{y_1 x - y_1 x_0 - y_0 x + y_0 x_0 + y_0 x_1 - y_0 x_0}{x_1 - x_0} \\ &= \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0}, \end{aligned}$$

đó là công thức nội suy tuyến tính trong khoảng (x0, x1)

Công thức này cũng có thể được hiểu là mức trung bình có trọng số. Các trọng số tỷ lệ nghịch với khoảng cách từ điểm cuối đến điểm chưa biết; điểm gần hơn có ảnh hưởng nhiều hơn điểm xa hơn. Vì vậy, các trọng số là $1 - (x - x_0)/(x_1 - x_0)$Và$1 - (x_1 - x)/(x_1 - x_0)$, là khoảng cách chuẩn hóa giữa điểm chưa biết và từng điểm cuối. Bởi vì những tổng này bằng 1,

$$\begin{aligned} y &= y_0\left(1 - \frac{x - x_0}{x_1 - x_0}\right) + y_1\left(1 - \frac{x_1 - x}{x_1 - x_0}\right) \\ &= y_0\left(1 - \frac{x - x_0}{x_1 - x_0}\right) + y_1\left(\frac{x - x_0}{x_1 - x_0}\right) \\ &= y_0\left(\frac{x_1 - x}{x_1 - x_0}\right) + y_1\left(\frac{x - x_0}{x_1 - x_0}\right) \end{aligned}$$

mang lại công thức nội suy tuyến tính ở trên.

4. **Update Animation trong Vòng Lặp Chính:**
   Trong vòng lặp chính của chương trình OpenGL, sử dụng hàm thời gian để xác định thời điểm hiện tại.
   Ví dụ: **glfwGetTime**()

   Dựa vào thời gian hiện tại, tính toán giữa hai keyframes để lấy trạng thái tương ứng của vật thể.

5. **Render vật thể liên tục với trạng thái hiện tại**
   Sử dụng trạng thái được tính toán từ keyframes để vẽ vật thể.
   Hàm renderObject sẽ vẽ vật thể dựa trên vị trí và quay được tính toán.

THỰC HIỆN

Để tạo hoạt cảnh cho một vật thể 3D dựa trên góc độ của camera trong OpenGL, cần thực hiện các bước sau:

1. **Chuẩn bị và khởi tạo môi trường đồ họa:**
   Bắt đầu bằng việc khởi tạo môi trường đồ họa OpenGL, tạo cửa sổ đồ họa và kết nối nó với OpenGL context.
   - Khởi tạo GLFW
   - Cấu hình GLFW
   - Tạo cửa sổ

2. **Chuẩn bị và tải shader:**
   Sử dụng shader để xác định cách hiển thị vật thể và cách camera nhìn vào vật thể.
   - Viết mã nguồn shader:
     Trước hết, cần viết mã nguồn cho vertex shader và fragment shader. Vertex shader thường xử lý các toạ độ của đỉnh và biến đổi chúng, trong khi fragment shader xử lý màu sắc và các thuộc tính của mỗi pixel.
   - Tạo và biên dịch shader:
     Tiếp theo, gọi **glCreateShader(GL_VERTEX_SHADER)** và **glCreateShader(GL_FRAGMENT_SHADER)** để tạo shader object. Sau đó, cung cấp mã nguồn shader và biên dịch chúng.
   - Liên kết shader program:
     Tiếp theo, tạo một shader program và liên kết vertex shader và fragment shader vào đó bằng cách sử dụng **glCreateProgram**, **glAttachShader**, và **glLinkProgram**.
   - Kích hoạt chương trình shader:
     Cuối cùng, kích hoạt chương trình shader bằng cách sử dụng **glUseProgram**. Khi chương trình shader này được kích hoạt, OpenGL sẽ sử dụng chúng để xử lý dữ liệu đầu vào và đầu ra của GPU.

3. **Tạo và quản lý đối tượng 3D:**
   - Tạo VAO và VBO: **glGenVertexArrays** được sử dụng để tạo một hoặc nhiều VAO và **glGenBuffers** để tạo một hoặc nhiều VBO.
   - Gắn VAO và VBO: **glBindVertexArray** được sử dụng để kích hoạt một VAO để có thể cấu hình các thuộc tính của đỉnh. **glBindBuffer** được sử dụng để kích hoạt một VBO để có thể truyền dữ liệu vào.

- Load dữ liệu của vật thể vào VBO: **glBufferData** được sử dụng để sao chép dữ liệu vào VBO. Trong trường hợp này, sizeof(vertices) là kích thước của mảng vertices, và **GL_STATIC_DRAW** cho biết rằng dữ liệu sẽ ít thay đổi và được sử dụng nhiều lần trong quá trình vẽ.
- Thiết lập thuộc tính con trỏ của vertex shader: **glVertexAttribPointer** được sử dụng để thiết lập các thuộc tính của đỉnh (ví dụ: tọa độ) trong vertex shader. Trong trường hợp này, chúng ta thiết lập thuộc tính vị trí của đỉnh (location = 0) và đặt các điểm dữ liệu trong mảng vertices.
- Kích hoạt thuộc tính đỉnh và giải phóng VBO và VAO nếu cần: **glEnableVertexAttribArray(0)** được sử dụng để kích hoạt thuộc tính vị trí của đỉnh trong vertex shader. Sau khi cài đặt và kích hoạt VAO và VBO, ta có thể giải phóng chúng nếu không cần thiết.

4. **Cài đặt và Tính toán Projection Matrix và View Matrix:**
   - **Tạo Projection Matrix:**
     - Chức năng: Chiếu phối cảnh là quá trình chuyển đổi không gian 3D sang không gian cắt (clipping space) và sau đó sang không gian hình chiếu (screen space).
     - Tính chất: Dùng để tạo ra hiệu ứng phối cảnh, giảm kích thước đối tượng khi chúng xa hơn camera và giữ nguyên kích thước khi chúng gần camera.
     - Tham số:
       - Góc nhìn (Field of View): Góc mở cửa (góc nhìn) của camera.
       - Tỉ lệ khung hình (Aspect Ratio): Tỉ lệ chiều rộng và chiều cao của viewport.
       - Khoảng cách cắt gần và cắt xa: Xác định khoảng cách tối thiểu và tối đa từ camera đến các điểm được hiển thị

   - **Tạo View Matrix:**
     - Chức năng: Định vị và hướng nhìn của camera trong không gian 3D.
     - Tính chất: Xác định vị trí và hướng nhìn của camera. Khi camera di chuyển, ma trận này thay đổi để duy trì góc nhìn và hướng của camera.
     - Tham số:
       - Vị trí của camera (Camera Position): Xác định vị trí của camera trong không gian.
       - Điểm nhìn về (Look-at Point): Xác định hướng mà camera hướng đến.
       - Hướng lên trên (Up Direction): Xác định hướng lên trên của camera.

- **Kết hợp Projection Matrix và View Matrix:**
    - Projection Matrix xác định cách các đối tượng trong không gian 3D được chuyển đổi thành không gian cắt và màn hình.
    - View Matrix xác định vị trí và hướng nhìn của camera trong không gian 3D.
    - Cả hai ma trận này được kết hợp (nhân với nhau) để tạo ra ma trận cuối cùng (View-Projection Matrix) được sử dụng để biến đổi toạ độ của các điểm từ không gian thế giới sang không gian màn hình.

5. **Vòng Lặp Vẽ:**
    - Áp dụng ma trận biến đổi cho vật thể
    - Kết hợp ma trận biến đổi để có ma trận mô hình-view-projection cuối cùng
    - Truyền ma trận mô hình-view-projection vào shader
    - Vẽ vật thể
    - Swap buffers và xử lý sự kiện

Chapter 4 : Triển khai và thử nghiệm

1. **Cài đặt thư viện và môi trường:**
    - Thư viện GLUT
    - Cài đặt IDE Visual Studio
    - Cài đặt trình biên dịch gcc cho C++

2. **Code demo:**
    a. **Include các thư viện**

```
#include <stdio.h>
#include <stdlib.h>
#include <GL\freeglut.h>
#include <math.h>
#include <string.h>
```

b. Các biến toàn cục

```
/* ASCII code for the escape key. */
#define ESCAPE 27

GLint window;
GLint window2;
GLint Xsize = 1920;
GLint Ysize = 1080;
float i, theta;
GLint nml = 0, day = 1;

char name3[] = "PROJECT:  3D ANIMATION";

GLfloat xt = 0.0, yt = 0.0, zt = 0.0, xw = 0.0;   /* x,y,z translation */
GLfloat tx = 295, ty = 62;
GLfloat xs = 1.0, ys = 1.0, zs = 1.0;

GLfloat xangle = 0.0, yangle = 0.0, zangle = 0.0, angle = 0.0;   /* axis angles */

GLfloat r = 0, g = 0, b = 1;
GLint light = 1;
int count = 1, flg = 1;
int view = 0;
int flag1 = 0, aflag = 1;            //to switch car driving mode
int wheelflag = 0;   //to switch fog effect
GLUquadricObj* t;
```

c. Các hàm quan trọng

- Hàm điều chỉnh góc nhìn

```
/* Simple  transformation routine */
GLvoid Transform(GLfloat Width, GLfloat Height)
{
    glViewport(0, 0, Width, Height);              /* Set the viewport */
    glMatrixMode(GL_PROJECTION);                  /* Select the projection matrix */
    glLoadIdentity();               /* Reset The Projection Matrix */
    gluPerspective(45.0, Width / Height, 0.1, 100.0);  /* Calculate The Aspect Ratio Of The Window */
    glMatrixMode(GL_MODELVIEW);                   /* Switch back to the modelview matrix */
}
```

- Hàm khởi tạo các components của OpenGL

```
/* A general OpenGL initialization function.  Sets all of the initial parameters. */
GLvoid InitGL(GLfloat Width, GLfloat Height)
{

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glLineWidth(2.0);              /* Add line width,   ditto */
    Transform(Width, Height); /* Perform the transformation */
    //newly added
    t = gluNewQuadric();
    gluQuadricDrawStyle(t, GLU_FILL);

    glEnable(GL_LIGHTING);

    glEnable(GL_LIGHT0);

    // Create light components
    GLfloat ambientLight[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    GLfloat diffuseLight[] = { 0.8f, 0.8f, 0.8, 1.0f };
    GLfloat specularLight[] = { 0.5f, 0.5f, 0.5f, 1.0f };
    GLfloat position[] = { 1.5f, 1.0f, 4.0f, 1.0f };

    // Assign created components to GL_LIGHT0
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
    glLightfv(GL_LIGHT0, GL_POSITION, position);

}
```

- Hàm điều chỉnh kích thước cửa sổ

```
/* The function called when our window is resized  */
GLvoid ReSizeGLScene(GLint Width, GLint Height)
{
    if (Height == 0)     Height = 1;                  /* Sanity checks */
    if (Width == 0)      Width = 1;
    Transform(Width, Height);                  /* Perform the transformation */
}
```

- Hàm khởi tạo cửa sổ khi mở chương trình

```
void init()
{
    glClearColor(0, 0, 0, 0);
    glPointSize(5.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 900.0, 0.0, 600.0, 50.0, -50.0);
    glutPostRedisplay();        // request redisplay
}
```

- Hàm vẽ toàn bộ cửa sổ (gọi các hàm khởi tạo cửa số và vẽ cửa sổ và vẽ, xoay, phóng to, thu nhỏ đối tượng 3D)

```
GLvoid DrawGLScene() { ... }
```

- Hàm xử lý dữ liệu nhập từ bàn phím

```
/*  The function called whenever a "normal" key is pressed. */
void NormalKey(GLubyte key, GLint x, GLint y) { ... }

static void SpecialKeyFunc(int Key, int x, int y) { ... }
```

- Hàm main để thực thi chương trình

```
//************************* Main ****************************************************************

int main(int argc, char** argv)
{

    /* Initialisation and window creation */

    glutInit(&argc, argv);              /* Initialize GLUT state. */

    glutInitDisplayMode(GLUT_RGBA |      /* RGB and Alpha */
        GLUT_DOUBLE |     /* double buffer */
        GLUT_DEPTH);     /* Z buffer (depth) */

    glutInitWindowSize(Xsize, Ysize);     /* set initial window size. */
    glutInitWindowPosition(0, 0);         /* upper left corner of the screen. */

    glutCreateWindow("3D CAR ANIMATION"); /* Open a window with a title. */

   /* Now register the various callback functions */

    glutReshapeFunc(myreshape);
    glutDisplayFunc(DrawGLScene);        /* Function to do all our OpenGL drawing. */
    glutReshapeFunc(ReSizeGLScene);
    glutKeyboardFunc(NormalKey);         /*Normal key is pressed */
    glutSpecialFunc(SpecialKeyFunc);
    InitGL(Xsize, Ysize);


    /* Now drop into the event loop from which we never return */

    glutMainLoop();                      /* Start Event Processing Engine. */
    return 1;
}
```

d. Chương trình demo

Cách hoạt động của chương trình:

- Chạy chương trình
- Hiển thị màn hình giới thiệu các thao tác có thể thực hiện với mô hình 3D (nhấn space để vào chương trình chính)
- Chương trình chính gồm một mô hình 3D được xây dựng sẵn
- Chương trình sẽ luôn luôn đợi để bắt sự kiện dựa trên dữ liệu được nhập từ bàn phím, có thể thực hiện các thao tác trên mô hình 3D như xoay quanh các trục x, y, z; di chuyển trên trục x, y; phóng to, thu nhỏ.

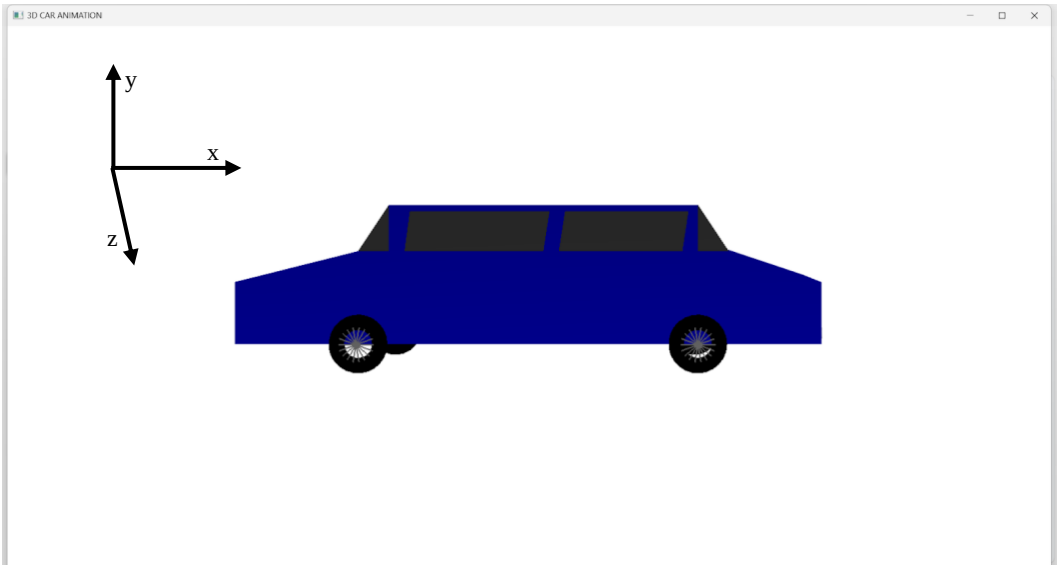
e. Các phím tương tác với chương trình

- **Màn MENU trước khi bước vào chương trình**

Hình 5: Menu của chương trình

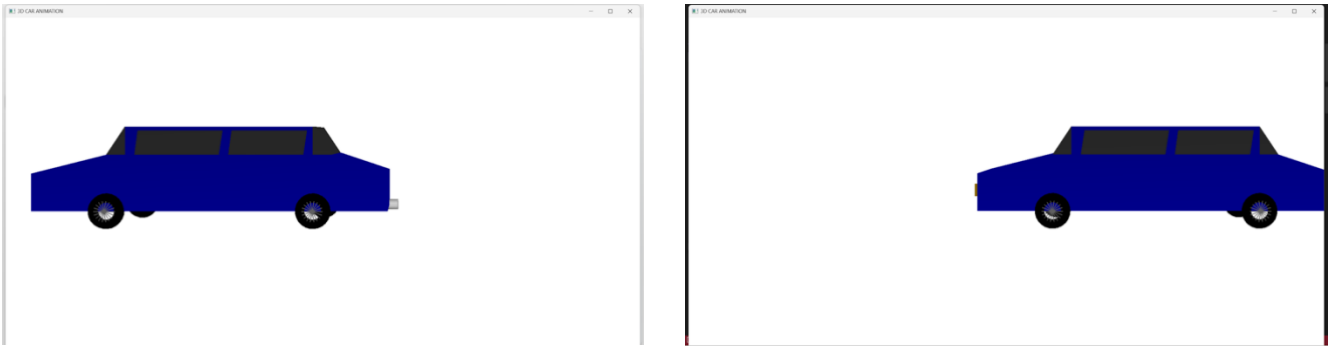
- **Các phím tương tác:**
  - Chuột phải: dùng để hiện lại menu khi đã bước vào cửa sổ mô phỏng.
  - x, y, z: xoay vật thể quanh theo trục tương ứng.
  - X, Y, Z: xoay vật thể quanh theo trục tương ứng theo hướng ngược lại.
  - a, s, q: phóng to vật thể tương ứng theo trục y, z, x.
  - A, S, Q: thu nhỏ vật thể tương ứng theo trục y, z, x.
  - U, F: điều chỉnh góc nhìn của camera tương ứng theo trục y và z.
  - Left arrow (←) và Right arrow (→): di chuyển vật thể theo hướng chỉ định.
  - Esc: thoát khỏi chương trình

- **Màn hình chương trình chính (đối tượng là một chiếc xe con 3D)**

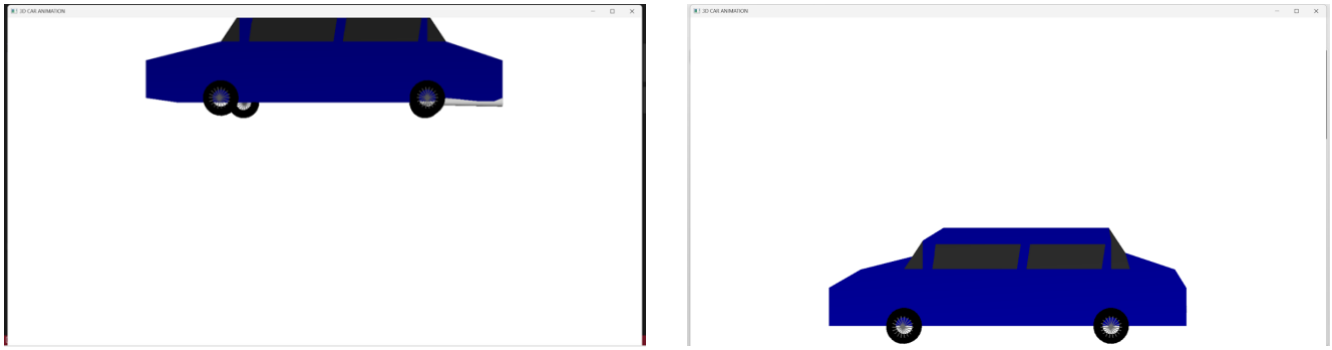


Hình 6: Đối tượng chính và các trục ảnh

- **Một số hình ảnh demo**
  - Di chuyển vật thể hoặc hướng camera

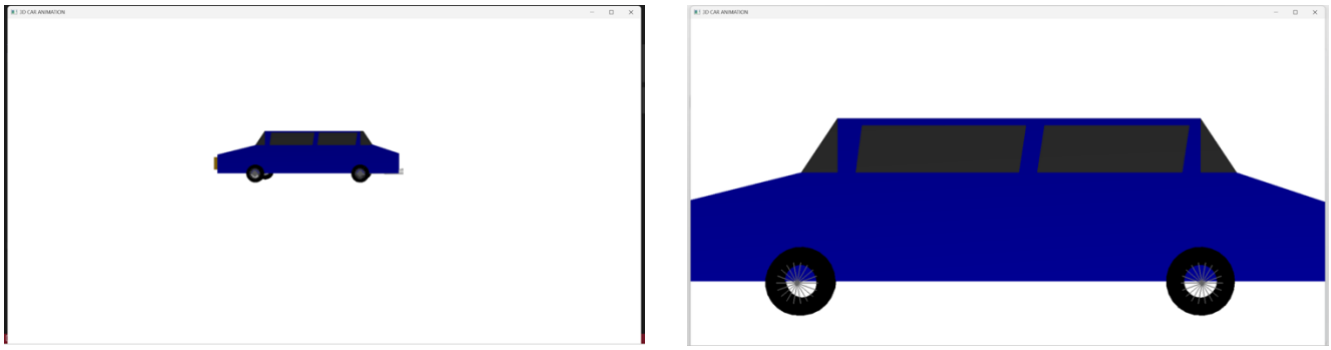


Hình 7: Di chuyển theo trục x (sang trái, phải)



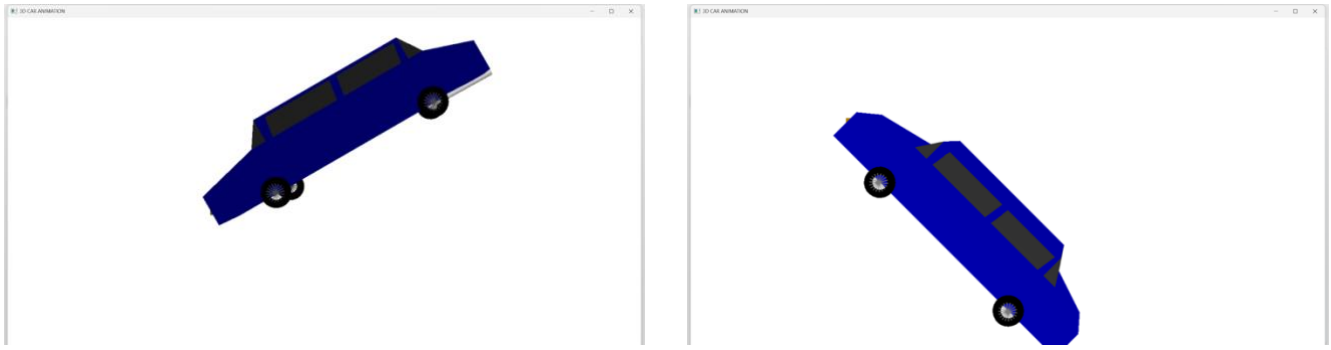
Hình 8: Di chuyển theo trục y (lên xuống)

- Thu phóng vật thể hoặc đổi góc nhìn camera

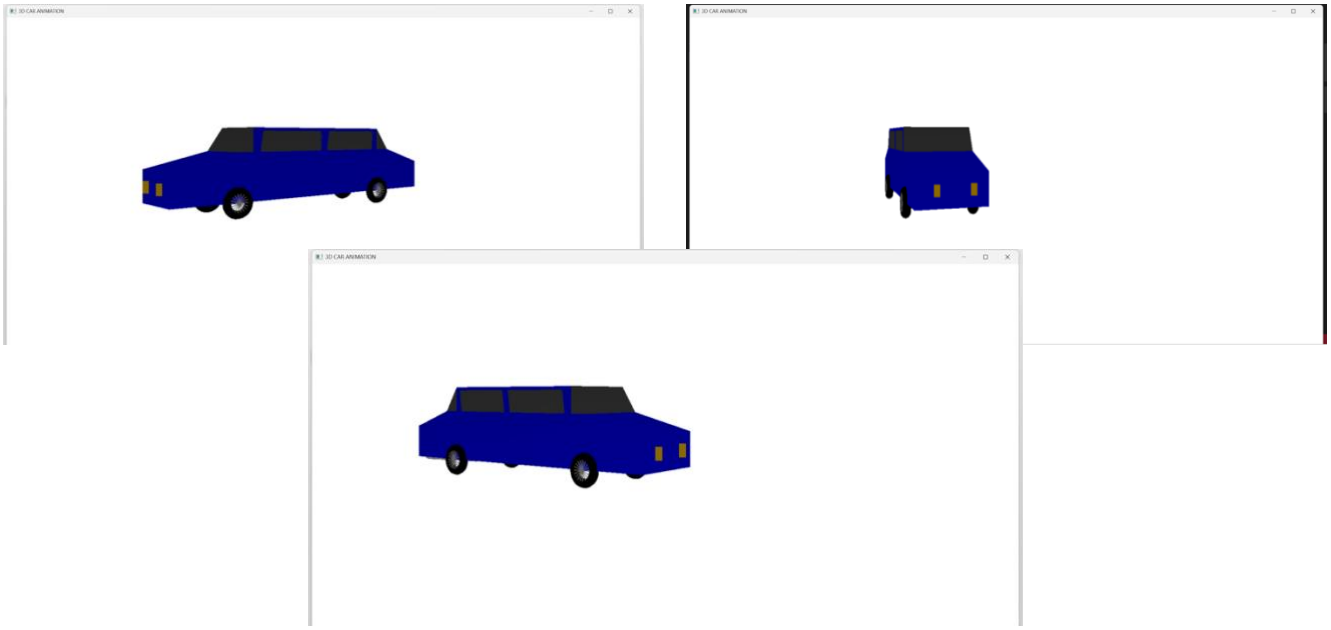


Hình 9: Thu phóng vật thể

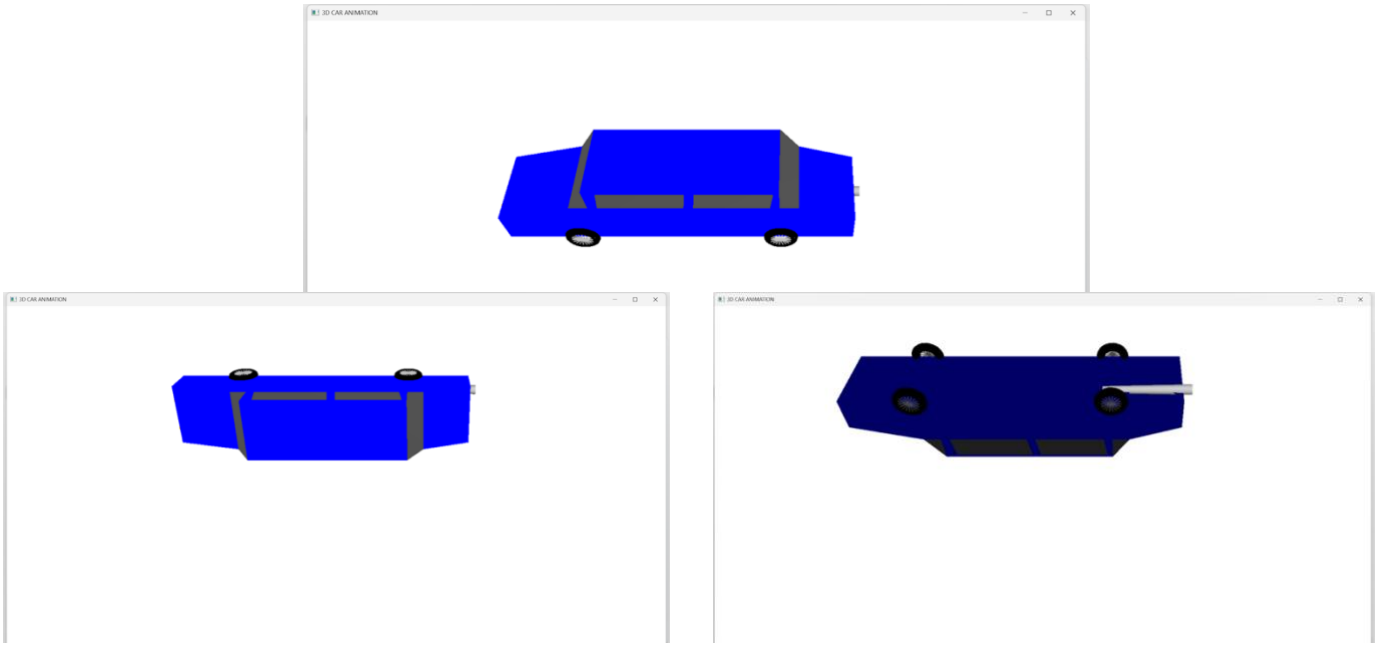
- Xoay vật thể



Hình 10: Xoay vật thể theo trục z (trục vuông góc màn hình)



Hình 11: Xoay vật thể theo trục y (trục hướng đứng)



Hình 12: Xoay vật thể theo trục x (trục ngang)

Chapter 5 : Kết luận và định hướng tương lai

Kết luận

Phương pháp tạo hoạt cảnh dựa vào góc độ của camera sử dụng Projection Matrix và View Matrix là một phương pháp hiệu quả và linh hoạt. Phương pháp này cho phép người dùng tạo ra các hoạt cảnh với nhiều góc nhìn khác nhau, từ đó mang lại trải nghiệm chân thực hơn cho người chơi.

Ví dụ, nếu người dùng muốn tạo ra một hoạt cảnh với góc nhìn từ trên xuống, họ có thể thay đổi các thông số của Projection Matrix để tạo ra một hình chiếu vuông góc. Điều này sẽ cho phép người dùng nhìn thấy toàn bộ hoạt cảnh từ trên cao.

Ngoài ra, phương pháp này cũng cho phép người dùng tạo ra các hiệu ứng đổ bóng và ánh sáng. Các hiệu ứng này có thể giúp tạo ra hoạt cảnh chân thực và hấp dẫn hơn.

Định hướng

Phương pháp này có thể được phát triển thêm theo một số hướng sau:

- Thêm các hiệu ứng đổ bóng, ánh sáng,... để tạo ra hoạt cảnh chân thực hơn. Hiện tại, phương pháp này chỉ áp dụng hiệu ứng đổ bóng đơn giản dựa trên góc nhìn của camera. Để tạo ra hoạt cảnh chân thực hơn, có thể thêm các hiệu ứng đổ bóng nâng

cao hơn, chẳng hạn như đổ bóng Phong, đổ bóng môi trường,... Ngoài ra, có thể thêm các hiệu ứng ánh sáng khác nhau, chẳng hạn như ánh sáng mặt trời, ánh sáng đèn,... để tạo ra bầu không khí sống động hơn cho hoạt cảnh.

- Thêm khả năng tương tác với môi trường xung quanh để người chơi có thể trải nghiệm hoạt cảnh một cách trọn vẹn hơn. Hiện tại, hoạt cảnh được tạo ra bằng phương pháp này là tĩnh, người chơi chỉ có thể quan sát mà không thể tương tác với môi trường xung quanh. Để mang lại trải nghiệm trọn vẹn hơn cho người chơi, có thể thêm khả năng tương tác với môi trường xung quanh, chẳng hạn như cho phép người chơi di chuyển, tương tác với các đối tượng trong môi trường,...

- Tối ưu hóa thuật toán để cải thiện hiệu suất, đặc biệt là đối với các hoạt cảnh phức tạp. Đối với các hoạt cảnh phức tạp, thuật toán hiện tại có thể làm giảm hiệu suất. Để cải thiện hiệu suất, có thể tối ưu hóa thuật toán theo một số hướng, chẳng hạn như sử dụng các thuật toán tính toán nhanh hơn, sử dụng các kỹ thuật giảm thiểu tài nguyên,...

END