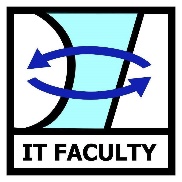
ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

Tel. (+84.0236) 3736949, Fax. (84-511) 3842771

Website: http://dut.udn.vn/khoacntt, E-mail: cntt@dut.udn.vn



BÁO CÁO MÔN HỌC

ĐỒ HỌA MÁY TÍNH

**ĐỀ TÀI :**

XÂY DỰNG CHƯƠNG TRÌNH OPENGL GLUT ĐỌC FILE VRML

VÀ HIỂN THỊ ĐỐI TƯỢNG 3D

|  |  |  |
| --- | --- | --- |
| HỌ TÊN SINH VIÊN | MÃ SINH VIÊN | NHÓM |
| Hoàng Minh Đức | 123200134 | 20.89 |
| Huỳnh Minh Trí | 123200129 | 20.89 |
| Phan Nguyên Đạt | 123200096 | 20.89 |

CBHD : PGS.TS.Nguyễn Tấn Khôi

Đà nẵng, 12/2023

MỤC LỤC

[CHƯƠNG 1:CƠ SỞ LÝ THUYẾT 3](#_Toc155187796)

[1.1. THỰC TẠI ẢO 3](#_Toc155187797)

[1.2. NGÔN NGỮ THỰC TẠI ẢO VRML 3](#_Toc155187798)

[1.3. PHÁT BIỂU BÀI TOÁN 5](#_Toc155187799)

[1.4. KẾT CHƯƠNG 5](#_Toc155187800)

[CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 6](#_Toc155187801)

[2.1. PHÁT BIỂU BÀI TOÁN 6](#_Toc155187802)

[2.2. PHÂN TÍCH CHỨC NĂNG 6](#_Toc155187803)

[2.2.1. Các bước thực hiện: 6](#_Toc155187804)

[2.2.2. Chương trình và phân tích chức năng: 6](#_Toc155187805)

[2.3. KẾT CHƯƠNG 11](#_Toc155187806)

[CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 12](#_Toc155187807)

[3.1. KẾT QUẢ THỰC NGHIỆM 12](#_Toc155187808)

[3.1.1. File VRML hiển thị đối tượng 3D hình nón : 12](#_Toc155187809)

[3.1.2. File VRML hiển thị đối tượng 3D hình hộp: 12](#_Toc155187810)

[3.2. NHẬN XÉT VÀ ĐÁNH GIÁ KẾT QUẢ 13](#_Toc155187811)

[3.3. KẾT CHƯƠNG 13](#_Toc155187812)

DANH SÁCH HÌNH ẢNH

***Hình ảnh 3.1: Hiển thị đối tượng 3D hình nón từ file VRML***

***Hình ảnh 3.2: Hiển thị đối tượng 3D hình hộp từ file VRML***

DANH SÁCH TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ viết tắt** | **Diễn giải** |
| VR | Virtual reality |
| VRML | Virtual reality modeling language |
| IDE | Integrated Development Environment |

MỞ ĐẦU

**1.**  **Tổng quan về đề tài**

Ngày nay trước sự phát triển mạnh của các ngành khoa học kỹ thuật và những  
ứng dụng của khoa học kỹ thuật vào cuộc sống đã mang lại hiệu quả to lớn đặc biệt là Công nghệ thông tin. Bất kỳ đâu và bất kể lĩnh vực nào của cuộc sống Công nghệ  
thông tin đều có mặt, hiệu quả mà Công nghệ thông tin mang lại thật to lớn và là điều được tất cả mọi người quan tâm. Hiện nay, Công nghệ thông tin ở nước ta đã phát triển mạnh mẽ, hiệu quả kinh tế mà Công nghệ thông tin mang lại cũng rất to lớn. Công nghệ thông tin phát triển theo chiều sâu, phục vụ các ngành công nghiệp sản xuất như: Tự động hóa, khai thác, thăm dò, hàng không, quân sự. Vì vậy chúng ta cần phải có những nghiên cứu và phát triển Công nghệ thông tin theo chiều sâu hoặc theo khía cạnh nào đó.  
 Thực tại ảo là một lĩnh vực đem lại những ứng dụng và hiệu quả là vô cùng to lớn ví dụ chúng ta có thể điều khiển một cái máy bay mà không cần có bất kỳ một chiếc máy bay nào hay là thực tập một ca mổ mà không cần phải sử dụng đến các đối tượng thực.

**2. Mục đích và ý nghĩa của đề tài**

***2.1. Mục đích***

Nghiên cứu và phát triển một chương trình sử dụng OpenGL và GLUT để đọc và hiển thị các đối tượng 3D từ file VRML. Mục tiêu chính là tạo ra một công cụ hữu ích giúp người dùng có thể dễ dàng tương tác, hiển thị các mô hình 3D từ các tệp VRML một cách hiệu quả và trực quan.

***2.2. Ý nghĩa***

* Tăng cường trải nghiệm người dùng: Chương trình cung cấp một giao diện đơn giản để người dùng có thể xem và kiểm tra các mô hình 3D từ file VRML, từ đó tăng cường trải nghiệm và khả năng tiếp cận của người dùng với công nghệ 3D.
* Phát triển công cụ hỗ trợ đồ họa 3D: Việc này góp phần phát triển các công cụ và ứng dụng hỗ trợ đồ họa 3D, đặc biệt là trong lĩnh vực giáo dục, thiết kế, và giải trí, nơi mà việc hiển thị và tương tác với các mô hình 3D là rất quan trọng.
* Nâng cao kiến thức và kỹ năng lập trình: Qua việc nghiên cứu và phát triển chương trình, người thực hiện có cơ hội nâng cao kiến thức và kỹ năng về lập trình đồ họa máy tính, thao tác với thư viện OpenGL, cũng như hiểu biết sâu sắc hơn về cấu trúc và đặc điểm của file VRML.

**3. Phương pháp thực hiện**

Phương án thực hiện cho việc nghiên cứu và triển khai Xây dựng chương trình OpenGL GLUT đọc file VRML và hiển thị đối tượng 3D có thể bao gồm các bước sau:

1. Nghiên cứu lý thuyết: hiểu Tìm hiểu về cấu trúc cơ bản của file VRML và ngôn ngữ VRML. Đọc tài liệu và hướng dẫn liên quan đến VRML để hiểu cách mô tả đối tượng 3D.
2. Xây dựng thuật toán: Sử dụng thư viện VRML hoặc tự viết mã để đọc và phân tích file VRML. Chuyển đổi dữ liệu VRML thành đối tượng 3D sử dụng cấu trúc dữ liệu phù hợp.
3. Thực nghiệm và đánh giá: Kiểm tra ứng dụng để đảm bảo rằng nó hoạt động đúng và hiệu quả. Tối ưu hóa mã nguồn để cải thiện hiệu suất nếu cần.
4. Cải Tiến và Phát Triển: Dựa trên kết quả thực nghiệm và phản hồi, tiếp tục cải tiến và phát triển chương trình để nó trở nên hiệu quả hơn.

**4. Bố cục của đồ án**

Báo cáo bao gồm các nội dung sau:

Mở đầu

Chương 1: **Cơ sở lý thuyết**

Chương 2: **Phân tích thiết kế hệ thống**

Chương 3: **Triển khai và đánh giá kết quả**

Kết luận và hướng phát triển.

# CHƯƠNG 1:CƠ SỞ LÝ THUYẾT

## THỰC TẠI ẢO

Thực tại ảo hay còn gọi là thực tế ảo (tiếng Anh là virtual reality, viết tắt là VR) là thuật ngữ miêu tả một môi trường mô phỏng bằng máy tính, trong môi trường mô phỏng đó con người có thể quan sát và thực hiện những thao tác mà mình mong muốn. Với tất cả những gì có trong môi trường mô phỏng thông qua thiết bị đầu vào, kết quả trả lại là những sự thay đổi của môi trường đó mà con người có thể quan sát, hay cảm nhận được thông qua các thiết bị đầu ra.

Ngoài việc tạo ra không gian ảo, công nghệ thực tế ảo VR còn có thể tương tác thực tế với người dùng qua cử chỉ và nhiều giác quan khác nhau như: Thính giác, khứu giác và xúc giác. Ngày nay, VR đã được ứng dụng vào nhiều lĩnh vực như: quân sự, hàng không vũ trụ, y học, giáo dục,…Tuy nhiên, để có được mô hình thực tại ảo như trên thì cần phải có một ngôn ngữ xây dựng chúng, tiếp theo ta tìm hiểu ngôn ngữ VRML để xây dựng mô hình thực tại ảo.

## NGÔN NGỮ THỰC TẠI ẢO VRML

Ngôn ngữ mô hình hoá thực tại ảo (VRML- virtual reality modeling language) là ngôn ngữ sử dụng mô hình phân cấp trong việc thể hiện các tương tác với các đối tượng của mô hình, VRML được sử dụng để phát triển những hình ảnh 3 chiều và quang cảnh trên World Wide Web. Các tập tin VRML có kích thước nhỏ, thường không quá 1MB. Tập tin nhỏ cho phép người dùng tương tác với quang cảnh bằng cách di chuyển "qua" hình ảnh.... Ví dụ, người dùng có thể "bước vào" một tòa nhà và đi từ phòng này qua phòng khác để nhìn tòa nhà từ mọi góc độ.

**Đặc điểm cơ bản của VRML:**

Tiêu chuẩn cho việc xác định đối tượng 3D, quang cảnh và cho sự liên kết các mô hình với nhau là:

* Không phụ thuộc nền nghĩa là chạy trên các máy tính do các nhà sản xuất khác nhau chế tạo.
* Có thể mở rộng nghĩa là có thể chấp nhận các lệnh mới do người sử dụng thêm vào
* Thao tác được thế giới ảo thông qua môi trường Internet có băng thông thấp

**VRML được thiết kế nhằm đáp ứng các yêu cầu sau:**

* Sự uỷ nhiệm (Authorability): cho phép phát triển các chương trình máy tính có khả năng tạo, chỉnh sửa và bảo trì các tập tin VRML. Bên cạnh đó, người dùng có thể viết các chương trình dịch tự động cho phép chuyển đổi các định dạng tập tinba chiều thông dụng sang các tập tin định dạng VRML.
* Có khả năng kết hợp hay cấu thành tổng thể (Composability): Hỗ trợ khả năng sử dụng và kết hợp các đối tượng ba chiều động bên trong một tổng thể và do đó cho phép tái sử dụng.
* Có thể mở rộng (Extensibility): cung cấp khả năng chèn thêm các kiểu đối tượng mới chưa được định nghĩa sẵn trong VRML.
* Có nhiều khả năng cài đặt (Be capable of implementation): VRML có thể được cài đặt trên một lượng lớn các hệ thống máy tính.
* Sự thể hiện (Performance): Có được sự thể hiện tương tác, co giãn trên nhiều loại nền khác nhau.
* Hiệu suất cao: nhấn mạnh hiệu suất khả co, tương tác trên một lượng lớn các hệ nền tính toán.
* Tính khả co: cho phép tạo ra các thế giới ba chiều động có độ lớn tùy ý.

**Tập tin của VRML:** Cấu trúc một tập tin VRML gồm có các thành phần: header, scene graph, prototype, và event routing.

1. *Header*: dùng để nhận dạng tập tin VRML và cách mã hóa. Header của VRML file bắt đầu bằng dấu # ý nghĩa đánh dấu những gì theo sau nó là phần chú thích. File tiêu đề của VRML có dạng :

#VRML V1.0 ascii dành cho phiên bản VRML 1.0, #VRML V2.0 utf-8 dành cho phiên bản VRML 2.0

1. *Scene graph*: chứa những nút mô tả các đối tượng và những thuộc tính đi kèm. Nó gần như một cây phả hệ gồm các nhóm đối tượng (hình học).
2. *Prototype*: cho phép một tập các nút kiểu VRML được mở rộng bởi người sử dụng. Các định danh kiểu này có thể được bao hàm trong file (mà chúng được sử dụng ) hay định nghĩa ở bên ngoài (file đó).
3. *Event routing*: một số nút có thể phát sinh những sự kiện đáp trả những thay đổi môi trường do tương tác phía người dùng.

**Các thẻ (node) cơ bản WorldInfo, Shape, Appearance trong VRML:**

* *Thẻ WorldInfo* : Đây là nút chứa thông tin chung của File VRML, được hiển thị trong thanh tiêu đề của cửa sổ trình duyệt.
* *Thẻ Shape*: Tất cả các đối tượng trong thẻ Shape đều nhìn thấy được. Thẻ Shape có hai thuộc tính: appearance và geometry. Thuộc tính appearance cho biết thẻ Appearance được dùng để xác định màu và các thao tác như quay, co giãn, dịch chuyển đối tượng hình học. Thuộc tính geometry chỉ rõ dạng hình học được vẽ ra như thế nào.
* *Thẻ Appearance* xác định hình dạng của một đối tượng hình học nào đó. Thẻ này chỉ ở bên trong thẻ Shape. Thẻ Appearance có các thuộc tính: material, texture và textureTransform.

Ngoài ra, một số thẻ như:

* Thẻ Box cho phép xây dựng một hình hộp chữ nhật
* Thẻ Cone cho phép chúng ta xác định độ cao, bán kính của hình nón.
* Thẻ Cylinder cho phép chúng ta xác định độ cao, bán kính của hình  
  trụ
* Thẻ Sphere có một trường cho phép xác định bán kính của hình cầu.

## PHÁT BIỂU BÀI TOÁN

Các giải pháp hiện có thường là các ứng dụng hoặc công cụ chuyên nghiệp như các phần mềm đồ họa 3D, trình xem VRML chuyên dụng, hoặc các thư viện lập trình phức tạp. Việc phát triển một giải pháp đơn giản, dễ sử dụng nhưng vẫn đủ mạnh mẽ để hiển thị chính xác và đẹp mắt các mô hình 3D từ tệp VRML là cần thiết và có ý nghĩa thiết thực.

Dựa trên những phân tích trên, báo cáo này sẽ tập trung vào việc nghiên cứu và phát triển một chương trình sử dụng OpenGL và GLUT để đọc và hiển thị các đối tượng 3D từ file VRML.

## KẾT CHƯƠNG

Chương này trình bày tổng quan về khái niệm VR, VRML, đặc điểm, cấu trúc cơ bản của một file VRML.

# CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 2.1. PHÁT BIỂU BÀI TOÁN

Chương trình này được phát triển sử dụng thư viện đồ họa OpenGL và giao diện lập trình ứng dụng GLUT với mục đích chính là đọc và trực quan hóa các đối tượng ba chiều được mô tả trong các file VRML. Mục tiêu cụ thể của chương trình bao gồm:

* Đầu vào: File VRML, định dạng chuẩn cho mô tả cấu trúc và thuộc tính của các đối tượng 3D.
* Đầu ra: Hiển thị hình ảnh của đối tượng 3D trên cửa sổ ứng dụng, cho phép người dùng quan sát từ nhiều góc độ khác nhau thông qua các thao tác điều khiển như xoay, phóng to, thu nhỏ.

## 2.2. PHÂN TÍCH CHỨC NĂNG

### 2.2.1. Các bước thực hiện:

Việc xây dựng chương trình OpenGL Glut để đọc file VRML và hiển thị đối tượng 3D bao gồm:

* Phân tích cú pháp file VRML: đọc dữ liệu từ file, phân tích cấu trúc và các thành phần của nó như nút (nodes), thuộc tính, và hình dạng.
* Khởi tạo và cấu hình môi trường đồ họa OpenGL: cấu hình các thông số như chiếu sáng, vật liệu, camera, và các thuộc tính khác liên quan đến cách đối tượng 3D được hiển thị và tương tác.
* Xử lý dữ liệu đối tượng và tạo các mô hình 3D tương ứng: Các đối tượng đọc được từ file VRML được chuyển đổi thành các đối tượng mà OpenGL có thể hiểu và hiển thị.
* Hiển thị và tương tác: Sau khi mô hình 3D đã được tạo, chúng được hiển thị trên màn hình sử dụng OpenGL. Có thể tương tác với mô hình thông qua các phím điều khiển để xoay, phóng to, thu nhỏ hoặc di chuyển camera quan sát.

### 2.2.2. Chương trình và phân tích chức năng:

* **Khai báo các class:** đại diện cho các hình dạng cụ thể như Box, Cone, Sphere, Cylinder. Lưu trữ thông tin và triển khai hàm draw().

// Khai báo các lớp con tạo ra các hình dạng cụ thể như Box, Cone, Sphere, và Cylinder

// Mỗi lớp con này sẽ triển khai hàm draw() để vẽ hình dạng đó

class Box : public Object3D {

public:

    float width, height, depth;

    Box(float w, float h, float d) : width(w), height(h), depth(d) {}

    void draw() const override {

        glPushMatrix();

        glScalef(width, height, depth);

        glutSolidCube(1.0);

        glPopMatrix();

    }

};

class Cone : public Object3D {

public:

    float bottomRadius;

    float height;

    Cone(float br, float h) : bottomRadius(br), height(h) {}

    void draw() const override {

        glColor3f(1.0, 1.0, 1.0);

        glutSolidCone(bottomRadius, height, 20, 20);

    }

};

class Sphere : public Object3D {

public:

    float radius;

    Sphere(float r) : radius(r) {}

    void draw() const override {

        glColor3f(1.0, 1.0, 1.0); //

        glutSolidSphere(radius, 20, 20);

    }

};

class Cylinder : public Object3D {

public:

    float radius;

    float height;

    Cylinder(float r, float h) : radius(r), height(h) {}

    void draw() const override {

        glColor3f(1.0, 1.0, 1.0);

        glutSolidCone(radius, height, 20, 20);

        glTranslatef(0.0, 0.0, height);

        glutSolidSphere(radius, 20, 20);

    }

};

* **Hàm readNode():**được thiết kế để đọc và phân tích từng nút (node) trong file VRML. Mỗi nút có thể đại diện cho một đối tượng 3D cụ thể như hộp (Box), hình cầu (Sphere), hình nón (Cone), hay hình trụ (Cylinder). Hàm này đảm nhận việc nhận dạng loại nút và thu thập thông tin cần thiết để tạo ra đối tượng 3D tương ứng

// Hàm đọc và phân tích cú pháp Node từ file VRML

void readNode(stringstream& ss) {

    string token;

    while (ss >> token) {

        if (token == "Box") {

            float w = 1, h = 1, d = 1;

            string subToken;

            while (ss >> subToken) {

                if (subToken == "size") {

                    ss >> w >> h >> d;

                    break;

                }

            }

            objects.push\_back(new Box(w, h, d));

        } else if (token == "Cone") {

            float bottomRadius, height;

            bool radiusFound = false, heightFound = false;

            if (ss.peek() == '{') {

                ss.ignore();

            }

            while (ss >> token) {

                if (token == "bottomRadius") {

                    ss >> bottomRadius;

                    radiusFound = true;

                } else if (token == "height") {

                    ss >> height;

                    heightFound = true;

                } else if (token == "}") {

                    break;

                }

            }

            if (radiusFound && heightFound) {

                objects.push\_back(new Cone(bottomRadius, height));

            }

        } else if (token == "Sphere") {

            float radius;

            bool radiusFound = false;

            if (ss.peek() == '{') {

                ss.ignore();

            }

            while (ss >> token) {

                if (token == "radius") {

                    ss >> radius;

                    radiusFound = true;

                } else if (token == "}") {

                    break;

                }

            }

            if (radiusFound) {

                objects.push\_back(new Sphere(radius));

            }

        } else if (token == "Cylinder") {

            float radius, height;

            bool radiusFound = false, heightFound = false;

            if (ss.peek() == '{') {

                ss.ignore();

            }

            while (ss >> token) {

                if (token == "radius") {

                    ss >> radius;

                    radiusFound = true;

                } else if (token == "height") {

                    ss >> height;

                    heightFound = true;

                } else if (token == "}") {

                    break;

                }

            }

            if (radiusFound && heightFound) {

                objects.push\_back(new Cylinder(radius, height));

            }

        }

    }

}

**- Hàm readVRML ():**

Hàm readVRML() dựa trên đường dẫn được cung cấp. Nếu file không tồn tại hoặc không thể mở, hàm sẽ thông báo lỗi và dừng quá trình đọc file.

* Đọc toàn bộ nội dung file vào một đối tượng stringstream. Điều này cho phép xử lý nội dung file như một luồng dữ liệu liên tục.
* Gọi hàm readNode() để phân tích cú pháp từ luồng stringstream. Hàm readNode() sau đó sẽ xác định các đối tượng 3D dựa trên các token được đọc từ file VRML và tạo ra các thực thể đối tượng 3D tương ứng.

// Hàm đọc file VRML, mở file và gọi readNode() để xử lý

void readVRML(const string& filePath) {

    ifstream file(filePath.c\_str());

    if (!file.is\_open()) {

        cerr << "Error: Could not open file " << filePath << endl;

        return;

    }

    stringstream ss;

    ss << file.rdbuf();

    file.close();

    readNode(ss);

}

**- Hàm main()**

* glutInit(&argc, argv); khởi tạo thư viện GLUT với các tham số dòng lệnh.
* glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH); đặt chế độ hiển thị với đệm kép (cho phép hiển thị mượt mà hơn), ..
* glutInitWindowSize(800, 600); và glutCreateWindow("OpenGL VRML Viewer"); tạo cửa sổ ứng dụng với kích thước và tiêu đề được chỉ định.
* glEnable(GL\_DEPTH\_TEST); cho phép OpenGL hiểu và xử lý độ sâu của các đối tượng trong không gian 3D.
* setupView(800, 600); cài đặt máy chiếu phối cảnh và camera quan sát.
* glClearColor(1.0, 1.0, 1.0, 1.0); đặt màu nền cho cửa sổ là màu trắng.
* glutDisplayFunc(display); hàm display() để xử lý việc vẽ cảnh 3D.
* glutReshapeFunc(reshape); hàm reshape() để xử lý việc cửa sổ thay đổi kích thước.
* glutKeyboardFunc(keyboard); hàm keyboard() để xử lý các sự kiện từ bàn phím.
* atexit(cleanup); hàm cleanup() để được gọi khi chương trình kết thúc.
* readVRML(filePath); đọc và phân tích cú pháp file VRML, tạo các đối tượng 3D tương ứng.
* Kiểm tra và xử lý lỗi: Nếu không có đối tượng nào được tạo (objects.empty()), thông báo lỗi và thoát.
* glutMainLoop(); bắt đầu vòng lặp sự kiện của GLUT, nơi mà tất cả các hàm callback được gọi khi cần thiết.

// Hàm main()

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

        // Khởi tạo GLUT và cấu hình cửa sổ

    // Ðang ký các hàm callback và bắt đầu vòng lặp sự kiện

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

    glutInitWindowSize(800, 600);

    glutCreateWindow("OpenGL VRML Viewer");

    glEnable(GL\_DEPTH\_TEST);

    setupView(800, 600);

    glClearColor(1.0, 1.0, 1.0, 1.0);  // R, G, B, A

    glutDisplayFunc(display);

    glutReshapeFunc(reshape);

    glutKeyboardFunc(keyboard);

    atexit(cleanup);

    string filePath = "path/file.wrl";// Ðường dẫn file VRML

    readVRML(filePath); // Ðọc file VRML

    if (objects.empty()) {

        cerr << "No objects to display. Exiting." << endl;

        return 1;

    }

    glutMainLoop();

    return 0;

}

Ngoài ra, chương trình còn có các hàm keyboard (): điều chỉnh góc nhìn, zoom đối tượng; hàm drawGrid(): vẽ lưới,…

## 2.3. KẾT CHƯƠNG

Chương này trình bày về phát biểu bài toán, phân tích chức năng của các hàm chính trong chương trình OpenGL Glut đọc file VRML hiển thị đối tượng 3D .

# CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

## 3.1. KẾT QUẢ THỰC NGHIỆM

### 3.1.1. File VRML hiển thị đối tượng 3D hình nón :

- Ngôn ngữ sử dụng: C++

- IDE: Dev-C++ 5.7.1   
Các bước thực hiện:

1. Nhập đường dẫn chứa file VRML hình nón
2. Chạy chương trình, ta được kết quả :

A black drop of water on a grid

Description automatically generated

***Hình ảnh 3.1: Hiển thị đối tượng 3D hình nón từ file VRML***

### 3.1.2. File VRML hiển thị đối tượng 3D hình hộp:

Các bước thực hiện:

1. Nhập đường dẫn chứa file VRML hình hộp chữ nhật
2. Chạy chương trình, ta được kết quả:

A black box on a grid

Description automatically generated

***Hình ảnh 3.1: Hiển thị đối tượng 3D hình hộp từ file VRML***

## 3.2. NHẬN XÉT VÀ ĐÁNH GIÁ KẾT QUẢ

Qua kết quả thực nghiệm, về cơ bản đã xây dựng chương trình OpenGL GLUT đọc file VRML và hiển thị được đối tượng 3D cơ bản. Bên cạnh đó, còn một số hạn chế của chương trình đó là các file VRML có thể chứa cấu trúc dữ liệu rất phức tạp, bao gồm nhiều node, thuộc tính, và tương tác. Điều này đòi hỏi phải có cơ chế phân tích cú pháp và xử lý dữ liệu tinh vi hơn để đảm bảo hiển thị chính xác và hiệu quả.

## 3.3. KẾT CHƯƠNG

Chương này trình bày kết quả thực nghiệm của chương trình đối với một số file VRML.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. **KẾT QUẢ ĐẠT ĐƯỢC**

Trong thời gian tìm hiểu, nghiên cứu cơ sở lý thuyết và triển khai ứng dụng công nghệ, báo cáo đã đạt được những kết quả sau:

*Về mặt lý thuyết,* báo cáo đã khám phá và hiểu sâu sắc cơ chế của đồ họa máy tính, đặc biệt là ứng dụng của OpenGL và GLUT trong việc tạo ra và hiển thị các đối tượng 3D từ file VRML.

*Về mặt thực tiễn*, đã triển khai thành công chương trình đọc và hiển thị đối tượng 3D từ file VRML. Chương trình có khả năng xử lý và hiển thị các đối tượng cơ bản như Box, Cone, Sphere, và Cylinder.

Kết quả đóng góp của báo cáo được thể hiện như sau:.

* Phát triển một thuật toán hiệu quả: đã xây dựng thành công một chương trình linh hoạt có khả năng đọc và hiển thị nhiều loại đối tượng 3D khác nhau từ file VRML.
* Xây dựng và chứng minh mô hình ứng dụng: Hiểu biết sâu về xử lý đồ họa 3D: đã phát triển được kiến thức chuyên sâu về lập trình đồ họa 3D, cũng như cách thức tích hợp và sử dụng thư viện OpenGL/GLUT, là nền tảng quan trọng cho những nghiên cứu và phát triển sau này.
* Nhận diện và giải quyết thách thức cụ thể: Đồ án đã chỉ ra các hạn chế và thách thức khi làm việc với file VRML và đồ họa 3D, từ đó đề xuất các giải pháp và cải tiến để nâng cao hiệu quả và chất lượng hiển thị

Tuy nhiên, vẫn còn các vấn đề cần giải quyết như tăng cường khả năng tương thích với nhiều định dạng VRML khác nhau, cải thiện hiệu suất hiển thị đối với các mô hình 3D phức tạp, và tăng cường tính tương tác và giao diện người dùng.

1. **KIẾN NGHỊ VÀ HƯỚNG PHÁT TRIỂN**

Một số số hướng nghiên cứu và phát triển của đề tài như sau:

* Tiếp tục phát triển chương trình bằng cách bổ sung các chức năng mới, hỗ trợ nhiều loại đối tượng và định dạng VRML khác nhau..
* Đánh giá hiệu năng trên các môi trường khác nhau để tối ưu hóa.
* Kiểm thử kỹ lưỡng các chức năng của chương trình.
* Tiếp tục nghiên cứu để cải thiện và mở rộng ứng dụng của hệ thống trong thực tế.

TÀI LIỆU THAM KHẢO

**Internet**

1. https://www.codeproject.com/Articles/82/A-small-VRML-viewer-using-OpenGL-and-MFC
2. http://doc.edu.vn/tai-lieu/do-an-tim-hieu-ve-ngon-ngu-thuc-tai-ao-vrml-va-ung-dung-87396/
3. https://cs.lmu.edu/~ray/notes/vrmlexamples/
4. http://www.brunel.ac.uk/~eestjpc/vrmlcourse/toc.htm
5. Thư mục tài liệu môn học (.\PUBLICDoHoaMayTinh\SOURCE) và các tài liệu liên quan khác.

**PHỤ LỤC**

#include <GL/glut.h>

#include <fstream>

#include <sstream>

#include <vector>

#include <cmath>

#include <cstdlib>

#include <iostream>

using namespace std;

//Khoi tao các bien de diêu chinh góc nhìn camera

float cameraAngleX = 0.0f;

float cameraAngleY = 0.0f;

float angleIncrement = 1.0f;

// Class co ban dai dien cho mot doi tuong 3D

class Object3D {

public:

    virtual void draw() const = 0; // Hàm ao de ke thua và trien khai trong các lop con

    virtual ~Object3D() {}

};

// Khai báo các lớp con tạo ra các hình dạng cụ thể như Box, Cone, Sphere, và Cylinder

// Mỗi lớp con này sẽ triển khai hàm draw() để vẽ hình dạng đó

class Box : public Object3D {

public:

    float width, height, depth;

    Box(float w, float h, float d) : width(w), height(h), depth(d) {}

    void draw() const override {

        glPushMatrix();

        glScalef(width, height, depth);

        glutSolidCube(1.0);

        glPopMatrix();

    }

};

class Cone : public Object3D {

public:

    float bottomRadius;

    float height;

    Cone(float br, float h) : bottomRadius(br), height(h) {}

    void draw() const override {

        glColor3f(1.0, 1.0, 1.0);

        glutSolidCone(bottomRadius, height, 20, 20);

    }

};

class Sphere : public Object3D {

public:

    float radius;

    Sphere(float r) : radius(r) {}

    void draw() const override {

        glColor3f(1.0, 1.0, 1.0); // Set the color to yellow

        glutSolidSphere(radius, 20, 20);

    }

};

class Cylinder : public Object3D {

public:

    float radius;

    float height;

    Cylinder(float r, float h) : radius(r), height(h) {}

    void draw() const override {

        glColor3f(1.0, 1.0, 1.0); // Set the color to yellow

        glutSolidCone(radius, height, 20, 20);

        glTranslatef(0.0, 0.0, height); // Move to the top of the cone

        glutSolidSphere(radius, 20, 20); // Draw a sphere to close the top

    }

};

// Mang luu tru các doi tuong 3D

vector<Object3D\*> objects;

// Hàm đọc và phân tích cú pháp Node từ file VRML

void readNode(stringstream& ss) {

    string token;

    while (ss >> token) {

        if (token == "Box") {

            float w = 1, h = 1, d = 1;

            string subToken;

            while (ss >> subToken) {

                if (subToken == "size") {

                    ss >> w >> h >> d;

                    break;

                }

            }

            objects.push\_back(new Box(w, h, d));

        } else if (token == "Cone") {

            float bottomRadius, height;

            bool radiusFound = false, heightFound = false;

            if (ss.peek() == '{') {

                ss.ignore();

            }

            while (ss >> token) {

                if (token == "bottomRadius") {

                    ss >> bottomRadius;

                    radiusFound = true;

                } else if (token == "height") {

                    ss >> height;

                    heightFound = true;

                } else if (token == "}") {

                    break;

                }

            }

            if (radiusFound && heightFound) {

                objects.push\_back(new Cone(bottomRadius, height));

            }

        } else if (token == "Sphere") {

            float radius;

            bool radiusFound = false;

            if (ss.peek() == '{') {

                ss.ignore();

            }

            while (ss >> token) {

                if (token == "radius") {

                    ss >> radius;

                    radiusFound = true;

                } else if (token == "}") {

                    break;

                }

            }

            if (radiusFound) {

                objects.push\_back(new Sphere(radius));

            }

        } else if (token == "Cylinder") {

            float radius, height;

            bool radiusFound = false, heightFound = false;

            if (ss.peek() == '{') {

                ss.ignore();

            }

            while (ss >> token) {

                if (token == "radius") {

                    ss >> radius;

                    radiusFound = true;

                } else if (token == "height") {

                    ss >> height;

                    heightFound = true;

                } else if (token == "}") {

                    break;

                }

            }

            if (radiusFound && heightFound) {

                objects.push\_back(new Cylinder(radius, height));

            }

        }

    }

}

// Hàm đọc file VRML, mở file và gọi readNode() để xử lý

void readVRML(const string& filePath) {

    ifstream file(filePath.c\_str());

    if (!file.is\_open()) {

        cerr << "Error: Could not open file " << filePath << endl;

        return;

    }

    stringstream ss;

    ss << file.rdbuf();

    file.close();

    readNode(ss);

}

// Cài đặt chế độ xem cho cửa sổ OpenGL

void setupView(int width, int height) {

    glViewport(0, 0, width, height);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluPerspective(60.0, (double)width / (double)height, 0.1, 100.0);

    glMatrixMode(GL\_MODELVIEW);

}

// Vẽ lưới dễ quan sát không gian 3D

void drawGrid(int size, int step) {

    glColor3f(1.0, 1.0, 1.0);

    for (int i = -size; i <= size; i += step) {

        glBegin(GL\_LINES);

        glVertex3f(i, 0, -size);

        glVertex3f(i, 0, size);

        glEnd();

        glBegin(GL\_LINES);

        glVertex3f(-size, 0, i);

        glVertex3f(size, 0, i);

        glEnd();

    }

}

// Biến kiểm soát phóng to thu nhỏ

float zoomFactor = 1.0f; // Tỉ lệ zoom mặc định

// Hàm display() được gọi mỗi khi cần vẽ lại cửa sổ

void display() { // Xóa buffer, đặt camera, vẽ lưới và các đối tượng 3D

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    glLoadIdentity();

    gluLookAt(0.0, 0.0, 20.0 / zoomFactor, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

    glRotatef(cameraAngleX, 1.0, 0.0, 0.0);

    glRotatef(cameraAngleY, 0.0, 1.0, 0.0);

    glEnable(GL\_LIGHTING);

    glEnable(GL\_LIGHT0);

    GLfloat light\_position[] = {1.0, 1.0, 1.0, 0.0};

    glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

    drawGrid(10, 1);

    for (size\_t i = 0; i < objects.size(); ++i) {

        const Object3D\* obj = objects[i];

        glPushMatrix();

        glScalef(zoomFactor, zoomFactor, zoomFactor);

        obj->draw();

        glPopMatrix();

    }

    glutSwapBuffers();

}

// Xử lý sự kiện bàn phím

void keyboard(unsigned char key, int x, int y) {

    switch (key) {

        case 'a':

            cameraAngleY -= angleIncrement;

            break;

        case 'd':

            cameraAngleY += angleIncrement;

            break;

        case 'w':  // Quay lên

            cameraAngleX += angleIncrement;

            break;

        case 's':  // Quay xu?ng

            cameraAngleX -= angleIncrement;

            break;

        case '+': // Phím '+' d? zoom in

            zoomFactor += 0.1f;

            break;

        case '-': // Phím '-' d? zoom out

            zoomFactor -= 0.1f;

            if (zoomFactor < 0.1f) {

                zoomFactor = 0.1f;

            }

            break;

        case 27:  // ESC key

            exit(0);

            break;

    }

    glutPostRedisplay();

}

// Cài đặt lại viewport và máy chiếu khi cửa sổ thay đổi kích thước

void reshape(int w, int h) {

    setupView(w, h);

}

// Hàm dọn dẹp, giải phóng các đối tượng 3D khi chuong trình kết thúc

void cleanup() {

    for (size\_t i = 0; i < objects.size(); ++i) {

        delete objects[i];

    }

    objects.clear();

}

// Hàm main()

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

        // Khởi tạo GLUT và cấu hình cửa sổ

    // Ðang ký các hàm callback và bắt đầu vòng lặp sự kiện

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

    glutInitWindowSize(800, 600);

    glutCreateWindow("OpenGL VRML Viewer");

    glEnable(GL\_DEPTH\_TEST);

    setupView(800, 600);

    glClearColor(1.0, 1.0, 1.0, 1.0);  // R, G, B, A

    glutDisplayFunc(display);

    glutReshapeFunc(reshape);

    glutKeyboardFunc(keyboard);

    atexit(cleanup);

    string filePath = "path/file.wrl";// Ðường dẫn file VRML

    readVRML(filePath); // Ðọc file VRML

    if (objects.empty()) {

        cerr << "No objects to display. Exiting." << endl;

        return 1;

    }

    glutMainLoop();

    return 0;

}