**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA: CÔNG NGHỆ THÔNG TIN**

-----------------

BÁO CÁO BÀI TẬP LỚN THUỘC HỌC PHẦN: CÔNG NGHỆ THỰC TẠI ẢO

**MÔ PHỎNG HOẠT ĐỘNG SĂN BẮT CỦA NGƯỜI TIỀN SỬ**

**GVHD: Ths. Vũ Đức Huy**

**Sinh viên: Nguyễn Duy Long 1141260020**

**Nông Hải Hà 1141260033**

**Lớp: HTTT1 Khóa: 11**

**Hà Nội - 2019**

**LỜI NÓI ĐẦU**

Khi công nghệ thông tin phát triển, đồ họa máy tính cũng ngày càng phát triển và có nhiều ứng dụng rộng rãi và mang tính thực tế cao. Trong những ứng dụng ấy, việc mô phỏng lại thế giới thực bằng cách xây dựng những mô hình “ảo” trên máy tính luôn là một vấn đề lớn mà đồ họa máy tính hướng tới giải quyết. Công nghệ “thực tại ảo” ra đời và phát triển nhanh chóng, với khả năng mô phỏng một cách sống động hầu hết những mô hình trong thế giới thực, từ đơn giản cụ thể cho tới phức tạp trừu tượng. Ngôn ngữ mô hình hóa thực tại ảo (Virtual Reality Modeling Language - VRML) đã được xây dựng từ nhu cầu phát triển của công nghệ “ thực tại ảo” – mà dựa trên đó, con người có thể giải quyết bài toán mô phỏng lại thế giới thực trên máy tính.

Với môn công nghệ thực tại ảo chúng em đã được học, mang lại cho chúng em nhiều kiến thức bổ ích, hiểu rõ về công nghệ thực tại ảo cũng như các ứng dụng rộng rãi của nó, nắm bắt các kiến thức cơ bản về ngôn ngữ mô phỏng thực tại ảo VRML. Để hiểu rõ môn học và nắm vững kiến thức ngôn ngữ VRML chúng em đã thực hiện đề tài “Mô phỏng hoạt động săn bắt của người tiền sử”. Vì kiến thức còn chưa nhiều bài làm còn nhiều sai sót mong cô giáo hướng dẫn và chỉ bảo để bài được hoàn thiẹn hơn. Chúng em xin gửi lời cảm ơn chân thành nhất tới cô giáo **Ths. Vũ Đức Huy** - người đã truyền đạt cho chúng em những kiến thức về môn học để chúng em hoàn thành đề tài này.

MỤC LỤC

[PHẦN 1: MỞ ĐẦU 4](#_Toc27513395)

[1.1 Tên đề tài 4](#_Toc27513396)

[1.2 Nội dung nghiên cứu 4](#_Toc27513397)

[1.2.1. Kiến thức 4](#_Toc27513398)

[1.2.2. Kỹ năng 4](#_Toc27513399)

[PHẦN 2: KẾT QUẢ NGHIÊN CỨU 4](#_Toc27513400)

[2.1. Giới thiệu về thực tại ảo 4](#_Toc27513401)

[2.2. Ngôn ngữ VRML 4](#_Toc27513402)

[2.2.1 Cơ bản về ngôn ngữ VRML 4](#_Toc27513403)

[2.2.2 Các đối tượng hình học cơ bản trong VRML 6](#_Toc27513404)

[*2.2.2.1. Box- Hình hộp chữ nhật*. 6](#_Toc27513405)

[2.2.2.2. Cylinder – Hình trụ 7](#_Toc27513406)

[2.2.2.3. Cone – Hình nón. 7](#_Toc27513407)

[2.2.3. Các kiểu dữ liệu trong VRML 7](#_Toc27513408)

[2.2.4. Các nút trong VRML 9](#_Toc27513409)

[2.2.5. Các nút nội suy 11](#_Toc27513410)

[2.2.6. Các phép biến đổi trong VRML 12](#_Toc27513411)

[2.2.7. Các sự kiện 13](#_Toc27513412)

[2.2.8. Các ROUTER 14](#_Toc27513413)

[2.2.9. Tái sử dụng 14](#_Toc27513414)

[2.3. Chương trình 16](#_Toc27513415)

[PHẦN 3: KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM 19](#_Toc27513416)

[3.1. Kết quả đạt được 19](#_Toc27513417)

[3.2. Hướng phát triển 19](#_Toc27513418)

# **PHẦN 1: MỞ ĐẦU**

## **Tên đề tài**

Mô phỏng hoạt động săn bắt của người tiền sử

## **Nội dung nghiên cứu**

Với đề tài ”mô phỏng hoạt động săn bắt của người tiền sử” nhằm xây dựng ra một thế giới mô phỏng (môi trường ảo) trên máy tính, nhằm đưa người sử dụng vào một thế giới nhân tạo với không gian như ngoài thực tế. Người quan sát có thể quan sát rõ các hoạt động mà không cần phải đi thực tế.

### **1.2.1. Kiến thức**

Hiểu được ý nghĩa của việc sử dụng thực tại ảo.

Hiểu được cách sử dụng phần mềm VRML và thực hiện chạy chương trình.

Nắm bắt được các kiến thức về việc sử dụng các nút, các lệnh điều khiển trong khi thực hiện xây dựng chương trình.

### **1.2.2. Kỹ năng**

Biết sử dụng VRML để tạo các khối hình học cơ bản.

Xác định được các tọa độ của người nhìn.

# **PHẦN 2: KẾT QUẢ NGHIÊN CỨU**

## **2.1. Giới thiệu về thực tại ảo**

**Thực tế ảo** là công nghệ giúp con người có thể “cảm nhận” không gian mô phỏng một cách chân thực hơn nhờ vào một loại kính nhìn 3 chiều ( kính thực tế ảo). Môi trường **3D ảo** này được tạo ra và điều khiển bởi một hệ thống máy tính cấu hình cao.

Đặc biệt, không chỉ đơn thuần hiển thị hình ảnh 3D , một số **hệ thống VR** còn cho phép mô phỏng âm thanh và mùi khá là chân thực!

## **2.2. Ngôn ngữ VRML**

### **2.2.1 Cơ bản về ngôn ngữ VRML**

* *Khái niệm*

VRML (Virtual Reality Modeling Language) là ngôn ngữ mô hình hóa thực tại ảo, một định dạng tập tin được sử dụng trong việc mô tả thế giới thực và các đối tượng đồ họa tương tác ba chiều, sử dụng mô hình phân cấp trong việc thể hiện tương tác với các đối tượng của mô hình, được thiết kế dùng trong môi trường Internet, Intranet và các hệ thống máy khách cục bộ (local client) mà không phụ thuộc vào hệ điều hành.

Các ứng dụng 3D của VRML có thể truyền đi một cách dễ dàng trên mạng với kích thước khá nhỏ so với băng thông, phần lớn giới hạn trong khoảng 100 - 200KB. Nếu HTML là định dạng văn bản thì VRML là định dạng đối tượng 3D có thể tương tác và điều khiển thế giới ảo.

Hiện nay, VRML có lợi thế là sự đơn giản, hỗ trợ dịch vụ Web3D, có cấu trúc chặt chẽ, với khả năng mạnh mẽ, giúp cho việc xây dựng các ứng dụng đồ họa ba chiều một cách nhanh chóng và chân thực nhất.

VRML là một trong những chuẩn trao đổi đa năng cho đồ họa ba chiều tích hợp và truyền thông đa phương tiện, được sử dụng trong rất nhiều lĩnh vực ứng dụng, chẳng hạn như trực quan hóa các khái niệm khoa học và kỹ thuật, trình diễn đa phương tiện, giải trí và giáo dục, hỗ trợ web và chia sẻ các thế giới ảo. Với mục đích xây dựng định dạng chuẩn cho phép mô tả thế giới thực trên máy tính và cho phép chạy trên môi trường web, VRML đã trở thành chuẩn ISO từ năm 1997.

* *Lịch sử ra đời và phát triển*

Năm 1994, lần đầu tiên VRML được thảo luận tại hội nghị WWW, Gieneva, Thụy Sĩ. Tim Berners-Lee và Dave Raggett đã tổ chức ra phiên họp có tên là Birds of a Feather (BOF) để mô tả giao diện thực tại ảo trên WWW. Nhiều thành viên tham dự, phiên họp BOF đã mô tả nhiều dự án thực hiện việc xây dựng các công cụ hiển thị đồ họa 3D cho phép có nhiều thao tác hữu ích trên Web. Những thành viên này đã nhất trí đồng ý sự cần thiết cho các công cụ này có một ngôn ngữ chung, phổ biến cho định dạng, xác định việc mô tả thế giới 3D và các siêu liên kết WWW. Vì thế, cụm từ “the Virtual Reality Markup Language” ra đời, từ “Markup” sau đó đã được đổi thành “Modelling” để phản ánh bản chất tự nhiên của VRML.

Sau phiên họp BOF một thời gian ngắn thì tổ chức WWW-VRML được thành lập để tập trung vào xây dựng phiên bản VRML đầu tiên.

Vào tháng 3/ 1995, Công ty Silicon Graphics cộng tác với hãng Sony Research và Mitra để đưa ra phiên bản mới cho VRML. Bản đệ trình của Silicon Graphics có tên là “Moving Worlds” gửi đến tổ chức Request for Proposals cho việc xây dựng phiên bản mới VRML, bản đệ trình này là một minh chứng cho sự cộng tác thành công của tất cả các thành viên của Silicon Graphics, Sony và Mitra. Năm 1996 tại New Orleans, phiên bản đầu tiên của VRML 2.0 được đưa ra.

Vào tháng 7/1996, tổ chức tiêu chuẩn quốc tế (ISO) đã thống nhất ý kiến lấy phiên bản năm 1996 của VRML 2.0 để đưa ra xem xét vào tháng 4/1997. Sau khi bỏ phiếu về chuẩn ISO thì VRML97 được đưa ra như một chuẩn ISO vào năm 1997.

* *Đặc điểm cơ bản*

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 phần cứng: có thể chạy trên các máy tính do các nhà sản xuất khác nhau chế tạo.
* Có thể mở rộng: có thể chấp nhận các lệnh mới do người sử dụng thêm vào hoặc quy định.
* Thao tác được thế giới ảo thông qua môi trường Internet có băng thông thấp.

VRML được thiết kế dành riêng cho việc hiển thị thế giới 3D và không phải là sự mở rộng của HTML.

* *Công cụ soạn thảo*
* Sử dụng vrmlpad 3.0 để soạn thảo
* Để hiển thị các file VRML, sử dụng trình duyệt Cortona 3D Viewer của hãng Parallel Graphics. Phần mềm này sẽ giúp người dùng thuận tiện hơn khi xem các mô hình ảo trên máy tính một cách trực quan sinh động.

### 2.2.2 Các đối tượng hình học cơ bản trong VRML

Các đối tượng hình học trong thế giới ảo thường được cấu tạo từ các đối tượng hình học cơ bản như hình hộp, hình tròn, hình trụ, hình cầu … Và VRML cung cấp sẵn cho chúng ta một số các đối tượng hình học đơn giản đó dưới dạng các nút như **Box** (hình hộp), **Cone** (hình nón), **Cylinder** (hình trụ tròn), **Sphere** (hình cầu)… **Event Architecture – Kiến trúc sự kiện**

### **2.2.2.1. Box- Hình hộp chữ nhật*.***

Hình hộp chữ nhật tạo ra từ node Box có tâm tại tọa độ ( 0,0,0) và có kích

thước được quy định qua trường “size x y z” với x,y,z lần lượt là kích thước hình hộp theo các trục Ox, Oy, Oz trong hệ trục tọa độ Decac. Nếu để mặc định size của hình hộp (không khai báo giá trị của trường), thì hình hộp có kích thước mặc định từ -1 đến +1.

Shape {

geometry Box{ size 2.0 2.0 2.0 }

}

Sphere – Hình cầu.

Hình cầu được tạo ra từ node Sphere có tâm tại tọa độ (0,0,0) và có bán kính được quy định qua trường “radius”. Giá trị trường radius phải lớn hơn 0.0.

Shape {

geometry Sphere {

radius 1.0

}

}

### **2.2.2.2. Cylinder – Hình trụ**

Hình trụ đuợc tạo ra từ node Cylinder có tâm tại tọa độ (0,0,0), có bán kính quy định qua trường radius và có chiều cao quy định qua trường height.

Ba trường còn lại trong node quy định sự tồn tại (nếu mang giá trị TRUE) hoặc không tồn tại (khi mang giá trị FALSE): tương ứng là mặt đáy trên – top, mặt đáy dưới – bottom, và mặt bên – side.

Các mặt không tồn tại sẽ không được xét tới nếu toàn khối trụ tham gia vào các sự kiện (ví dụ như phát hiện ra va chạm hoặc kích hoạt cảm biến).

Shape {

geometry Cylinder{

height 2.0

radius 1.0

bottom TRUE

top TRUE

side TRUE

}

}

### **2.2.2.3. Cone – Hình nón.**

Hình nón được tạo ra từ node Cone có tâm tại (0,0,0), có bán kính mặt đáy quy định qua trường bottomRadius và có chiều cao quy định qua trường height.

Shape {

geometry Cone {

height 2.0

bottomRadius 1.0

bottom TRUE

side TRUE

}

}

### **2.2.3. Các kiểu dữ liệu trong VRML**

Trong VRML, các nút có thể chứa các nút khác và có thể chứa các trường. Mỗi trường có các kiểu dữ liệu khác nhau. Sau đây là mô tả của chúng:

* **SFBool:** Đây là giá trị kiểu logic, có thể nhận giá trị “TRUE” hay “FALSE”.
* **SFColor & MFColor:** SFColor là trường chứa màu sắc, được tạo nên từ ba số thực từ 0 đến 1 tương ứng với các giá trị màu red, green và blue (RGB) (ví dụ: 0 1 0 là green). MFColor là một tập hợp gồm các màu sắc (ví dụ: [0 1 0, 1 0 0, 0 0 1] là tập hợp gồm ba màu green, red và blue).
* **SFFloat & MFFloat:** SFFloat là giá trị kiểu số thực (ví dụ: 7.5). MFFloat là một tập các giá trị số thực (ví dụ: [1.0, 3.4, 76.54]).
* **SFImage:** SFImage là hình ảnh hai chiều được tạo nên từ các điểm ảnh. Nó bao gồm:
  + - Hai số nguyên, đại diện chiều rộng và chiều cao của hình ảnh cũng chính là số điểm nằm trên chiều rộng và chiều cao của hình.
    - Một số nguyên biểu thị số thành phần của hình ảnh. Số thành phần là 1 tức hình ảnh trắng đen, là 2 cũng tương tự nhưng có thêm độ trong suốt, là 3 thì hình sẽ có màu (RGB), và 4 là hình ảnh có màu kết hợp với độ trong suốt.
    - Sau đó, là các con số được biểu diễn dưới dạng thập lục phân đại diện cho màu sắccủa từng điểm ảnh. Ví dụ 0xFF là màu trắng trong hình ảnh một thành phần và 0xFF00007F sẽ là màu đỏ nhạt trong hình ảnh bốn thành phần.
    - Các điểm được chỉ rõ theo thứ tự từ trái sang phải, từ trên xuống dưới.

Ví dụ: 1 2 1 0xFF 0x00 *# Hình ảnh một thành phần gồm 2 điểm màu trắng và đen.*

* **SFInt32 & MFInt32:** Giá trị kiểu số nguyên hoặc dãy các số nguyên 32 bit có thể theo dạng thập phân hoặc thập lục phân. Các số thập lục phân 32 bit bắt đầu bằng “0x”, ví dụ 0xFF là 255 hệ thập phân.
* **SFNode & MFNode:** SFNode là một nút đơn lẻ và MFNode là danh sách các nút. Trường **children** với nhiều nút là kiểu MFNode.
* **SFRotation & MFRotation:** Các trường này chỉ ra một phép quay xung quanh một trục, nó được tạo nên từ 4 số thực, ba số thực đầu tiên chỉ rõ các toạ độ X, Y và Z cho các vectơ tương ứng với trục để quay, số thứ tư là góc (tính bằng radian) để quay. SFRotation là một tập, MFRotation là một danh sách các tập.
* **SFString & MFString:** Loại này chứa danh sách các kí tự trong tập kí tự **utf-8**, Nói đơn giản thì SFString là một xâu kí tự và MFString là tập các xâu kí tự. Ví dụ SFString là “Hello” (các kí tự nằm trong dấu nháy kép) còn MFString thì tham số truyền vào có dạng là [“Hello” “VRML”].
* **SFTime & MFTime:** Kiểu dữ liệu đặc trưng cho một mốc thời gian hoặc danh sách các mốc thời gian.
* **SFVec2f & MFVec2f:** Các vectơ 2D đơn lẻ hoặc danh sách các vectơ 2D. Một vectơ 2D được tạo bởi một cặp các số thực.
* **SFVec3f & MFVec3f:** Một vectơ 3D hoặc một danh sách các vectơ 3D. Một vectơ 3D là một bộ 3 số thực.

### **2.2.4. Các nút trong VRML**

VRML bao gồm 54 nút khác nhau và được phân loại làm 9 nhóm chính dựa trên chức năng và các hàm của các nút. Bao gồm:

* **Grouping Nodes**: Nhóm các nút nhóm.
* Anchor
* Billboard
* Collision
* Group
* Transform.
* **Special Groups Nodes**: Nhóm các nút nhóm đặc biệt.
* Inline
* LOD
* Switch.
* **Sensors Nodes**: Nhóm các nút cảm biến.
* CylinderSensor
* PlaneSensor
* ProximitySensor
* SphereSensor
* TimeSensor
* TouchSensor
* VisibilitySensor.
* **Geometry Nodes**: Nhóm các nút đối tượng hình học.
* Box
* Cone
* Cylinder
* ElevationGrid
* Extrusion
* IndexedFaceSet
* IndexedLineSet
* PointSet
* Sphere
* Text.
* **Geometry Properties Nodes**: Nhóm các nút thuộc tính hình học.
* Color
* Coordinate
* Normal
* TextureCoordinate.
* **Appearance Nodes**: Nhóm các nút mô tả hiển thị.
* Appearance
* FontStyle
* ImageTexture
* Material
* MovieTexture
* PixelTexture
* TextureTransform.
* **Interpolators Nodes**: Nhóm các nút nội suy.
* ColorInterpolator
* CoordinateInterpolator
* NormalInterpolator
* OrientationInterpolator
* PositionInterpolator
* ScalarInterpolator
* **Bindable Nodes**: Nhóm các nút có thể ghép được
* Background
* Fog
* NavigationInfo
* Viewpoint.

### **2.2.5. Các nút nội suy**

Các nút thuộc nhóm nút **Interpolators**có chức năng giữ các giá trị xen vào các trường khi có sự kiện thay đổi giá trị các trường xảy ra (trừ các trường có giá trị kiểu logic). Các nút thuộc nhóm nút này có cú pháp giống nhau (có cùng các trường) chỉ khác nhau về kiểu dữ liệu.

Trong VRML cung cấp sẵn các nút **Interpolators** sau:

* ColorInterpolator
* CoordinateInterpolator
* NormalInterpolator
* OrientationInterpolator
* PositionInterpolator
* ScalarInterpolator

Mỗi nút đều có cú pháp như sau:

…Interpolator {

key […]

keyValue […]

}

Trong đó **key** là tập các giá trị đầu vào còn **keyValue** là tập các giá trị đầu ra. Trong các nút **Interpolators** đều có một sự kiện đầu vào là **set\_fraction** và sự kiện đầu ra là **value\_changed**, hai sự kiện này được nối với nhau tức là khi các nút này khi nhận được một sự kiện thì nó cũng tạo ra một sự kiện. Sự kiện set\_fraction xác định một giá trị **key** và sự kiện đầu ra xác định một **keyValue** tương ứng với giá trị **key**.

### **2.2.6. Các phép biến đổi trong VRML**

Để tạo nên thế giới sử dụng tất cả mọi thứ, chúng ta cần có thể biến đổi được mọi đối tượng. VRML có 3 loại biến đổi có thể áp dụng cho các đối tượng. Đó là translation, rotations và scales và chúng được sử dụng trong nút transform, Nhưng không phải một nút transform chỉ có phép quay rotation. Các phép biến đổi trong transform có thế áp dụng cho các nút con children của nút. Điều này gọi là một tổ hợp, một nút cha có thé có nhiều nút con. Cú pháp cho vấn đề này được chỉ ra dưới đây, áp dụng cho nút transform

Transform{

Translation 1 1 1

Rotation 0 1 0 0.78

Scale 2 1 2

Children [

USE FBOX

]

}

Một nút transform có thể có các tổ hợp khác bên trong nút con của nó, cho phép bạn thực hiện tuần tự một chuỗi các phép biến đổi. Chú ý rằng trật tự các phép biến đổi là không quan trọng. Một phép biến đổi thực hiện sau phép quay. Bên trong một nút transform đơn lẻ, các phép biến đổi được thực hiện theo một trật tự chặt chẽ: Scale, rotation và translate. Vì vậy nếu bạn muốn thực hiện phép biến đổi sau phép quay, bạn cần tổ hợp các nút transform bên trong các nút khác.

#### **2.2.6.1. Translation and scale (Phép biến đổi và tính tỉ lệ)**

Có 2 phép biến đổi tương tự nhau, cả hai đều nhận ba đối số: giá trị x, y và z.

Phép biến đổi dịch chuyển trung tâm của đổi tượng trong các khoảng cách này theo hướng thích hợp. Tỉ lệ nhân với kích thước của đối tượng bởi các giá trị này theo hướng thích hợp. Một phép biến đổi 0 theo một hướng sẽ loại bỏ các đối tượng không ảnh hưởng đến hướng. Yếu tố tỉ lệ 0 làm cho đối tượng không xác định theo hướng đó là điều bình thường không được mong muốn. Yếu tố 1 yêu cầu mà không có ảnh hưởng nào cả.

Quan trọng chú ý rằng việc tính tỉ lệ là tương đối cho nguyên bản, không phải là trung tâm của đối tượng. Vì vậy để tính tỉ lệ của trung tâm đối tượng, chúng ta cần phải đảm bảo rằng đối tượng được đặt trung tâm tại phần mở đầu. Đó là lý do tại sao việc tính tỉ lệ phải thực hiện trước phép quay và phép biến đổi.

#### **2.2.6.2. Rotation (Phép quay)**

Phép quay có điểm hơi khác so với hai loại ở trên. Nó nhận 4 tham số, đầu tiên là 3 tọa độ xác định trục của phép quay và cuối cùng là góc quay tính bằng radian. Ví dụ để quay 1 radian quanh trục Y bạn phải viết:

Transform{

Rotate 0 1 0 1

Childen [

USE FBOX

]

}

Chiều dài của trục quay bất kỳ, không cần thiết là 1. Bạn có thể sử dụng giá trị Y là 50 nếu bạn thích nhưng nó không thực hiện điều gì khác với giá trị Y là 1. Trục quay hoàn toàn là tùy ý, bạn có thể quay quanh bất kỳ trục nào bạn thích. Ví dụ trục 1 0.3 2.45 là hoàn toàn hợp lý. Thật là khó khăn để tạo các phép quay theo cách này, tuy nhiên chúng ta có thể khó khăn để quan sát, đặc biệt nếu bạn không có bất kỳ luyện tập nào. Chính vì điều đó Vapour Technology đã tạo ra một cặp công cụ Dizzy và Twister. Twister tạo phép quay vuông góc với trục từ một laoij đơn giản hơn và Dizzy kết hợp nhân các phép quay vuông góc với trục để tạo một loại đơn giản hơn với hiệu quả tương tự. Bạn có thể sử dụng các điều này để dễ dàng hơn trong việc quay các đối tượng trong thế giới.

### **2.2.7. Các sự kiện**

Một số trường hợp, hầu như các nút đều có chứa các sự kiện. Có hai loại sự kiện: eventIn và eventOut. Các eventOut là các sự kiện sắp đi ra, phát sinh các thông tin như thay đổi 1 giá trị hoặc thời gian kích chuột. Các eventIn là các sự kiện sắp đến, chấp nhận thông tin từ bên ngoài của nút và thực hiện một vài điểu với nó. Một sự kiện có một loại dữ liệu thích hợp với chúng.

Một vài nút có các trường được trưng bày. Điều này có nghĩa là nút có hai trường định nghĩa cho trường đó set\_fieldname và fieldname\_changed. Có eventIn và eventOut đối với trường có thể sử dụng để thiết lập các giá trị của nó và thông báo cho thế giói bên ngoài có khi có sự thay đổi. Nếu bạn sử dụng set\_fieldname để thiết lập giá trị của trường, nút sẽ phát sinh một sự kiện fieldname\_changed. Để dễ dàng sử dụng, các thành phần set\_ và \_changed của sự kiện có thể nghiêng về phía trái và browser sẽ làm việc khi sự kiện được sử dụng. Nếu một trường không được phơi bày ra, nó không thể thay đổi do các sự kiện và giá trị trong trường được sử dụng trong suốt các khoảng thời gian. Để xem xét trường nào được bày ra đối với mỗi nút, hãy giữ lấy một tham chiếu từ một vị trí từ trang liên kết hoặc nhận lấy tham chiếu nút trong phụ lục.

### **2.2.8. Các ROUTER**

Để thực hiện hiệu quả mọi thứ với các sự kiện, chúng ta cần nối chúng với nhau. Việc kết nối được biết đến như ROUTER. Ví dụ để dẫn từ touchTime eventOut tới startTime eventIn, chúng ta sẽ dẫn sự kiện như sau:

ROUTE SSENSOR.touchTime TO SOUND.startTime

Vì vậy, bít này của mã sẽ dẫn tới sự kiện touchTime từ TouchSensor (sẽ đề cập tới ở phần sau) tới sự kiện startTime trong nút sound (cũng được để cập ở phần sau). Do đó khi TouchSensor bị kích vào âm thanh được phát ra. Bạn cần sử dụng DEF cho mỗi nút mà bạn dẫn tới hoặc từ đó sao cho nó có một tên riêng lẻ. Vì vậy các nút TouchSensor và Sound sẽ được định nghĩa:

 DEF SENSOR TouchSensor{

}

DEF SOUND Sound {

}

Ngoại trừ các trường hợp bên trong chúng. Nếu bạn có một số các đối tượng với cùng một tên (do sử dụng USE), và dẫn tới hoặc từ chúng, tất cả các đối tượng bị ảnh hưởng, vì vậy nếu bạn chỉ muốn một đối tượng bị ảnh hưởng, đưa cho nó một tên duy nhất hoặc sử dụng PROTO (được giải thích sau).

### **2.2.9. Tái sử dụng**

Nếu bạn có nhiều đối tượng giống hệt nhau, thường rất khó khăn để duy trì việc viết chính xác nhiều đối tượng theo cùng một loại. Do đó, bạn có thể sử dụng các định nghĩa ở phần trước. Sử dụng hộp này, bạn có thể định nghĩa nó để có tên FBOX. Mỗi khi bạn muốn sử dụng lại hộp, bạn có thể chỉ gõ USE FBOX thay cho toàn bộ định nghĩa. Ví dụ:

DEF FBOX Shape {

apprearance Apprearance {

material Material {

}

}

geometry Box {

}

}

USE FBOX

Đây không chỉ làm một ví dụ đơn giản, khi tạo hai hộp chính xác tại cùng một vị trí, không chỉ là các thứ mà bạn muốn. Thỉnh thoảng, là định nghĩa USE tại một vài nơi có ý nghĩa hơn. Đồng thời bạn có thể DEF/USE bất kỳ loại nút nào vì vậy bạn muốn sử dụng lại Appearance của một đối tượng, bạn có thể thực hiện tốt điểu đó.

Shape {

appearance DEF APP1 Appearance {

material Material {

}

}

geometry Box {

}

}

Shape {

appearance USE APP1

geometry Box {

}

}

Một lần nữa bạn có thể thấy ở đây ta có thể tạo hai hộp ở cùng một vị trí trong thế giới. Một cách khác để sử dụng lại mã VRML là sử dụng các nút Inline. Điều này cho phép nhận dữ liệu từ các file bên ngoài và chèn nó vào file của bạn. Vì vậy nếu bạn có mô hình chiếc ghế gọi là chair.wrl bạn có thể chèn nó vào quang cảnh của bạn:

Inline {

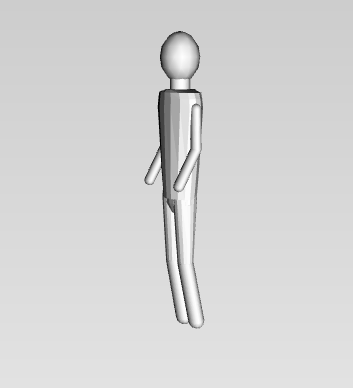
url “cay.wrl”

}

File bạn đang kết hợp theo các này phải là VRML hợp lệ, vì vậy nó phải là các header và có mọi thứ cần thiết cho một file VRML.

## **2.3. Chương trình**

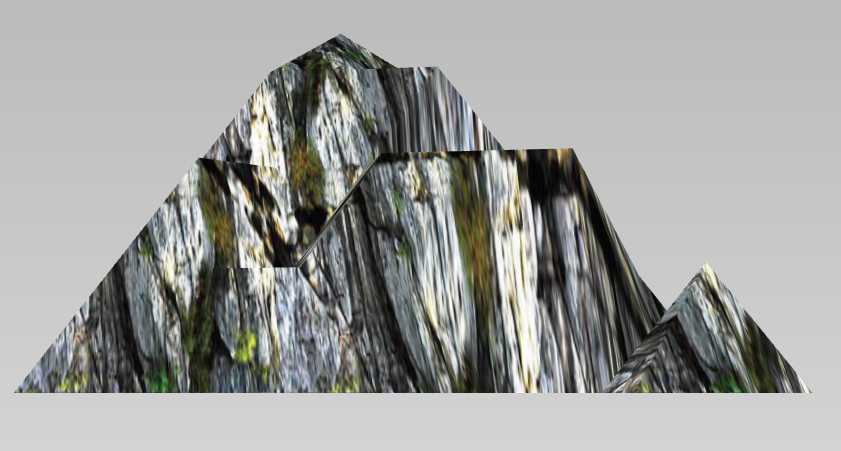
* Người di chuyển



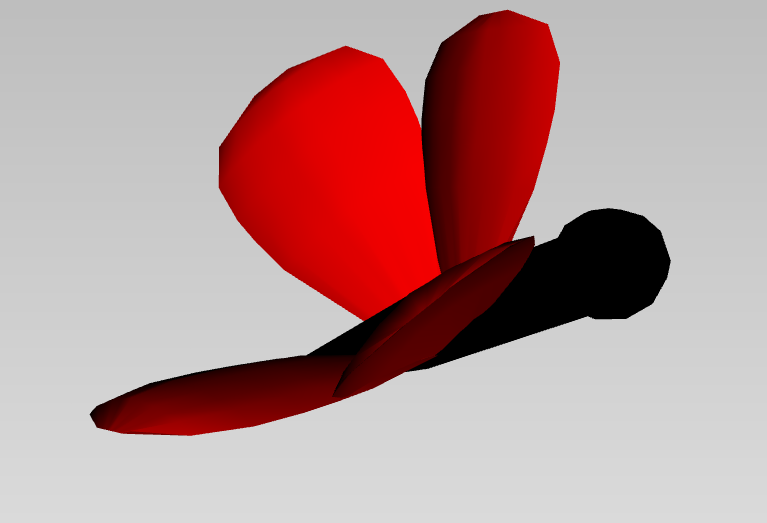
* Cây



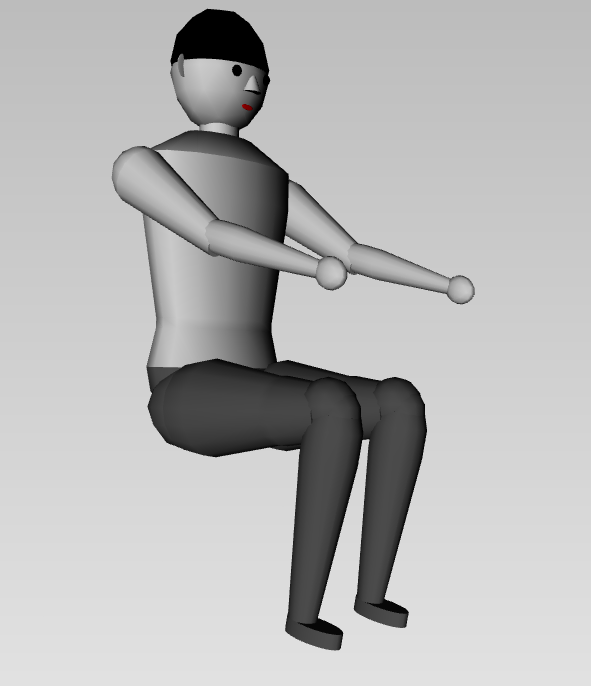
* Núi



* Con bướm



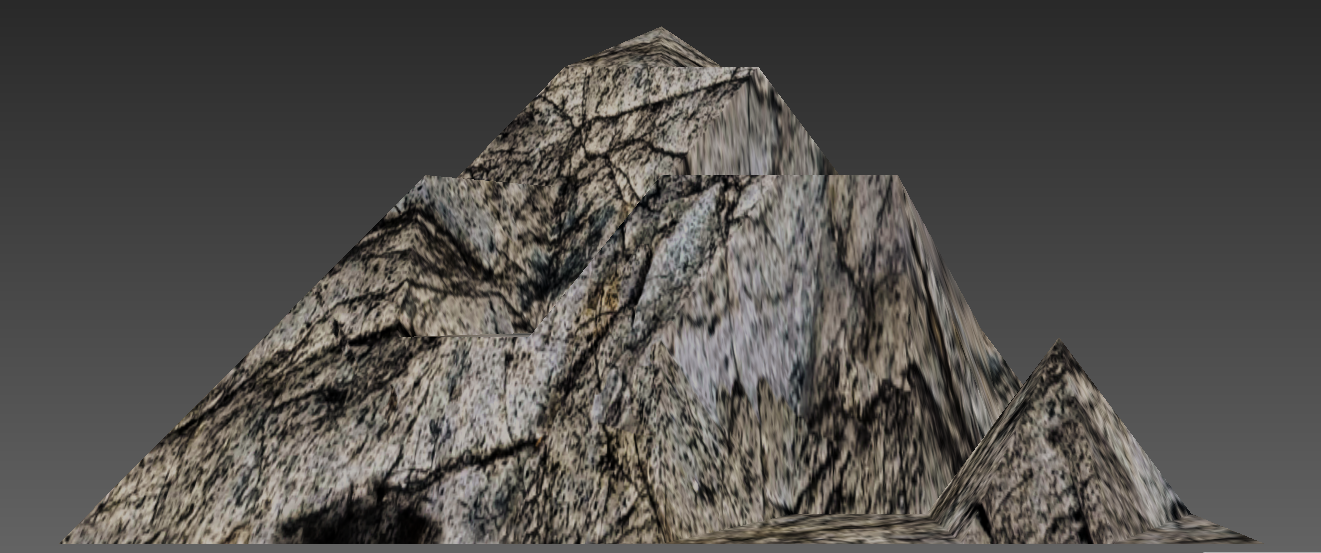
* Người ngồi



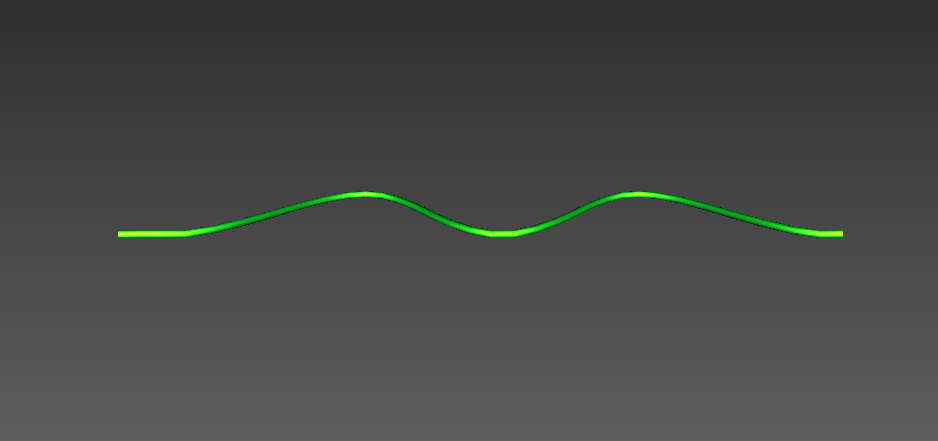
* Bụi cây



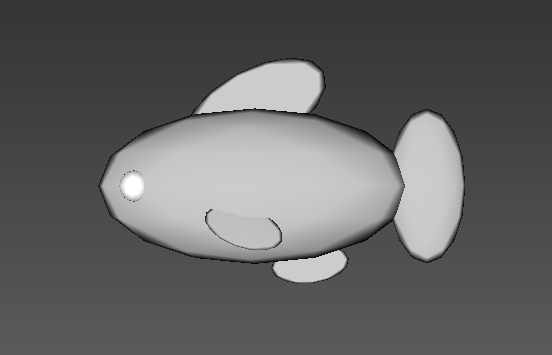
* Núi



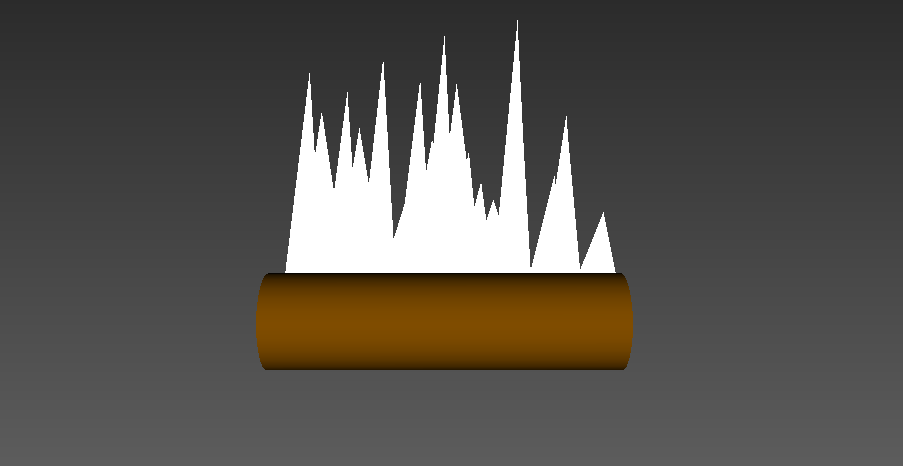
* Rắn



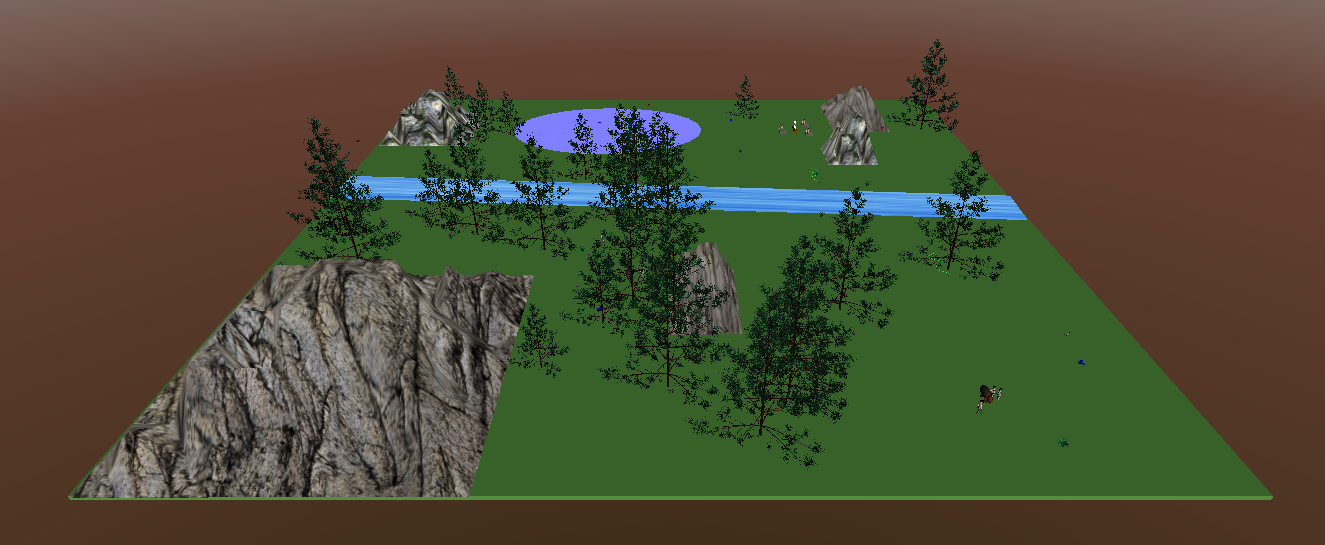
* Cá



* Đống lửa



* Quang cảnh



# PHẦN 3: KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM

## 3.1. Kết quả đạt được

Sau một thời giam tìm hiểu thảo luận với nhau, nhóm chúng em đã hoàn thành bài làm của mình. Do thời gian không cho phép nên nhóm chúng em đã không thể làm bài với các chi tiết được đẹp mong thầy thông cảm. Chúng em cảm ơn thầy vì đã tận tình giúp đỡ chúng em để chúng em có thể hoàn thành tốt bài làm của mình, và cho chúng em hiểu được nội dung của môn học cũng như kết hộp được các thành phần với nhau để tạo ra sự sinh động. Và sự kết hợp các sự kiện với nhau để tạo nên hành động cho các đối tượng. Chúng em cảm ơn thầy rất nhiều.

## 3.2. Hướng phát triển

Trong tương lai chúng em sẽ cố gắng tìm hiểu và học hỏi thêm nhiều công nghệ về phần mềm để phát triển thêm về môn học và cũng như phát triển những thiếu sót còn có trong bài.

TÀI LIỆU THAM KHẢO

1. Giáo trình ebook VRML, Trường đại học công nghiệp Hà Nội

[1] <http://123doc.org/document/100256-do-hoa-may-tinh-va-hien-thuc-ao.htm>l

[2] <http://www.ebook.edu.vn/?page=1.39&view=21190>

[3] <http://khotailieu.com/luan-van-do-an-bao-cao/van-hoa-nghe-thuat/thiet-ke-do-hoa/do-hoa-may-tinh-va-hien-thuc-ao.html>