

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**- - -🙟🕮🙝- - -**

****

**BÁO CÁO BÀI TẬP LỚN**

**MÔN: CÔNG NGHỆ THỰC TẠI ẢO**

**Đề tài: *Mô phỏng va chạm giao thông***

**Giảng viên HD : Th.S. Vũ Đức Huy**

**Lớp : Đại học Kĩ thuật phần mềm 1 – K11**

**Sinh viên TH : Nhóm 03**

**HÀ NỘI – 2019**

**MỤC LỤC**

[LỜI NÓI ĐẦU 3](#_Toc7906518)

[CHƯƠNG 1: KHẢO SÁT, XÂY DỰNG KỊCH BẢN 4](#_Toc7906519)

[1.1. Xây dựng kịch bản 4](#_Toc7906520)

[1.1.1 Yêu cầu bài toán 4](#_Toc7906521)

[1.2. Khảo sát và thiết kế hình ảnh va chạm giao thông: 4](#_Toc7906522)

[1.3 Các phần mềm hỗ trợ 6](#_Toc7906523)

[CHƯƠNG 2: CÁC KỸ THUẬT SỬ DỤNG TRONG VRML 7](#_Toc7906524)

[**2.1. Các thành phần một hệ thống VR** 7](#_Toc7906525)

[**2.1.1    Phần cứng (Hardware)** 7](#_Toc7906526)

[**2.1.2    Phần mềm (Software)** 7](#_Toc7906527)

[2.2. Giới thiệu các ngôn ngữ VRML 8](#_Toc7906528)

[2.2.1. Định nghĩa VRML 8](#_Toc7906529)

[2.2.2. Đặc điểm cơ bản 8](#_Toc7906530)

[2.2.3. Các đặc tính chính của VR 9](#_Toc7906531)

[2.2.4. Một số ứng dụng chính của VR 9](#_Toc7906532)

[2.3. Tập tin của VRML 13](#_Toc7906533)

[2.4. Các nút trong VRML 13](#_Toc7906534)

[2.4.1. Các đối tượng hình học cơ bản 16](#_Toc7906535)

[2.4.2. Các phép biến đổi 18](#_Toc7906536)

[2.4.3. Màu sắc và hình ảnh 21](#_Toc7906537)

[2.4.4. Âm thanh, ánh sáng, camera và phong cảnh 24](#_Toc7906538)

[2.4.5. Nhóm đối tượng 25](#_Toc7906539)

[2.4.6. Tái sử dụng đối tượng 27](#_Toc7906540)

[2.4.7. Nội suy và cảm biến 28](#_Toc7906541)

[CHƯƠNG 3: HÌNH ẢNH SẢN PHẨM SAU KHI MÔ PHỎNG MÔ HÌNH VA CHẠM GIAO THÔNG 31](#_Toc7906542)

[3.1. Hình ảnh tổng quan mô hình 31](#_Toc7906543)

[3.2. Các thành phần trong mô hình 32](#_Toc7906544)

[3.2.1. Tổng quan mô hình va chạm 32](#_Toc7906545)

[3.2.2. Một số vật thể trong mô hình 36](#_Toc7906546)

[TÀI LIỆU THAM KHẢO 39](#_Toc7906547)

# LỜI NÓI ĐẦU

Thực tại ảo là một thuật ngữ mới xuất hiện khoảng đầu thập kỷ 90, nhưng ở Mỹ và châu Âu thực tại ảo (Virtual Reality) đã và đang trở thành một công nghệ mũi nhọn nhờ khả năng ứng dụng rộng rãi trong mọi lĩnh vực (nghiên cứu và công nghiệp, giáo dục và đào tạo, thương mại và giải trí,…) và tiềm năng kinh tế, cũng như tính lưỡng dụng (trong dân dụng và quân sự) cuả nó. Tại Việt Nam, tuy là một lĩnh vực mới nhưng đã có những công trình rất hữu ích như: tái hiện lại con Sao La hay một Văn Miếu Quốc Tử Giám ảo mà ta có thể đi lại quan sát trong đó.

Chính vì tầm quan trọng cũng như khả năng ứng dụng to lớn đó nên việc nghiên cứu về thực tại ảo là vô cùng cần thiết. Và trên cơ sở đó có thể xây dựng một ứng dụng thực tại ảo hoàn chỉnh.

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 va chạm giao thông”. Bài tập lớn này gồm có phần mở đầu, kết luận và nội dung:

Chương 1: Khảo sát, xây dựng kịch bản

Chương 2: Các kỹ thuật sử dụng trong VRML

Chương 3: Kết quả mô phỏng Va chạm giao thông

Chúng em xin gửi lời cảm ơn chân thành nhất tới thầy Vũ Đức Huy - người đã truyền đạt cho chúng em những kiến thức về môn học để chúng em hoàn thành bài báo cáo này.

Nhóm 3

# CHƯƠNG 1: KHẢO SÁT, XÂY DỰNG KỊCH BẢN

## Xây dựng kịch bản

**Đề tài**: Mô phỏng Va chạm giao thông

### 1.1.1 Yêu cầu bài toán

Với những công năng to lớn mà ngôn ngữ thực tại ảo đem lại và nhằm mô phỏng lại các tình huống xảy ra tai nạn giao thông tại các ngã tư đường không có đèn tín hiệu: tai nạn giữa 2 xe ô tô, giũa xe ô tô và xe máy và tai nạn giữa 2 xe máy với nhau. Việc mô phỏng giúp cho chúng ta cân nhắc kĩ việc di chuyển phương tiện một cách cẩn thận, vì trường hợp xấu nào cũng có thể xảy ra khi tham gia giao thông.

Từ những vấn đề trên, nhóm chúng tôi đã bắt tay vào tìm hiểu và xây dựng mô hình: **“*****Mô phỏng Va chạm giao thông“***.

## Khảo sát và thiết kế hình ảnh va chạm giao thông:

Mẫu va chạm số 1



Mẫu va chạm số 2

 Mẫu va chạm số 3

*Hình ảnh* *được tham khảo qua trang tìm kiếm:* [*https://www.google.com.vn*](https://www.google.com.vn) *với từ khóa: “Va chạm giao thông”*

## 1.3 Các phần mềm hỗ trợ

**- VRMLPAD 3.0  
 - Autodesk 3ds Max 2016**

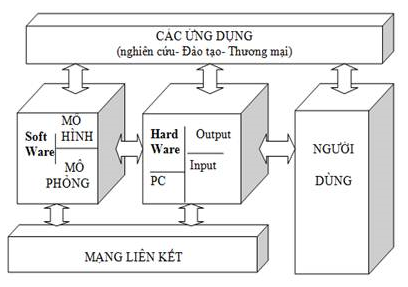
**- Cortona 3D**

**- Adobe Photoshop CC**

**- Paint**

# CHƯƠNG 2: CÁC KỸ THUẬT SỬ DỤNG TRONG VRML

**2.1. Các thành phần một hệ thống VR**

Tổng quát một hệ thống VR bao gồm những thành phần sau:

*Hình 1.1 Các thành phần của một hệ thống VR*

**2.1.1    Phần cứng (Hardware)**

Phần cứng của một VR bao gồm:

* Máy tính (PC hay Workstation với cấu hình đồ họa mạnh).
* Các thiết bị đầu vào (Input devices): Bộ dò vị trí (position tracking) để xác định vị trí quan sát. Bộ giao diện định vị (Navigation interfaces) để di chuyển vị trí người sử dụng. Bộ giao diện cử chỉ (Gesture interfaces) như găng tay dữ liệu (data glove) để người sử dụng có thể điều khiển đối tượng.
* Các thiết bị đầu ra (Output devices): gồm hiển thị đồ họa (như màn hình, HDM,..) để nhìn được đối tượng 3D nổi. Thiết bị âm thanh (loa) để nghe được âm thanh vòm (như Hi-Fi, Surround,...). Bộ phản hồi cảm giác (Haptic feedback như găng tay,...) để tạo xúc giác khi sờ, nắm đối tượng. Bộ phản hồi xung lực (Force Feedback) để tạo lực tác động như khi đạp xe, đi đường xóc,...

**2.1.2    Phần mềm (Software)**

Phần mềm luôn là linh hồn của VR cũng như đối với bất cứ một hệ thống máy tính hiện đại nào. Về mặt nguyên tắc có thể dùng bất cứ ngôn ngữ lập trình hay phần mềm đồ họa nào để mô hình hóa (modelling) và mô phỏng (simulation) các đối tượng của VR.

Ví dụ như các ngôn ngữ (miễn phí) OpenGL, C++, Java3D, VRML, X3D,... hay các phần mềm thương mại như WorldToolKit, PeopleShop,... Phần mềm của bất kỳ VR nào cũng phải bảo đảm 2 công dụng chính: Tạo hình vào Mô phỏng. Các đối tượng của VR được mô hình hóa nhờ chính phần mềm này hay chuyển sang từ các mô hình 3D (thiết kế nhờ các phần mềm CAD khác như AutoCAD, 3D Studio,...). Ngoài ra, phần mềm VR cần có khả năng mô phỏng động học, động lực học, và mô phỏng phản ứng của đối tượng.

## 2.2. Giới thiệu các ngôn ngữ VRML

### 2.2.1. Định nghĩa VRML

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.

### 2.2.2. Đặc điểm cơ bản

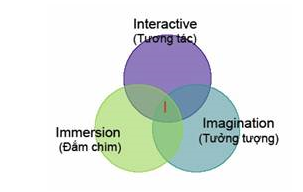
Tiêu chuẩn cho việc xác định đối tượng 3D, quang cảnh và cho sự liên kết các mô hình với nhau là:

* 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.

### 2.2.3. Các đặc tính chính của VR

Đặc tính chính của VR là Tương tác (Interactive) và Đắm chìm (Immersion). Tuy nhiên, VR không chỉ là một hệ thống tương tác Người - Máy, mà các ứng dụng của nó còn liên quan tới việc giải quyết các vấn đề thật trong kỹ thuật, y học, quân sự,... Các ứng dụng này phụ thuộc rất nhiều vào khả năng tưởng tượng (Imagination) của con người. Do đó có thể coi VR là tổng hợp của 3 yếu tố: Tương tác - Đắm chìm - Tưởng tượng (“3I” trong tiếng Anh: Interactive – Immersion - Imagination).



*Hình 1.2 Các đặc tính chính của VR*

### 2.2.4. Một số ứng dụng chính của VR

Tại các nước phát triển, VR được ứng dụng trong mọi lĩnh vực: Khoa học kỹ thuật, kiến trúc, quân sự, giải trí,... và đáp ứng mọi nhu cầu: Nghiên cứu - Giáo dục - Thương mại. Y học là lĩnh vực ứng dụng truyền thống của VR.

Ngoài ra, VR cũng đã được ứng dụng trong giáo dục, nghệ thuật, giải trí. Đặc biệt trong lĩnh vực quân sự, VR đã được ứng dụng rất nhiều ở các nước phát triển hiện nay. Bên cạnh các ứng dụng truyền thống ở trên, cũng có một số ứng dụng mới nổi lên trong thời gian gần đây của VR như: ứng dụng trong sản xuất, ứng dụng trong ngành robot, ứng dụng trong hiển thị thông tin (thăm dò dầu mỏ, hiển thị thông tin khối, ...). VR có tiềm năng ứng dụng vô cùng lớn, hầu hết các lĩnh vực “có thật” trong cuộc sống đều có thể ứng dụng “thực tại ảo” để nghiên cứu và phát triển hoàn thiện hơn.

**-    Trong y học**

Thực tại ảo giải quyết được rất nhiều vấn đề trong y học: cung cấp môi trường thực hành cho nghiên cứu và học tập, rất hữu ích trong việc mô phỏng các ca phẫu thuật nhằm giảm tối đa rủi ro trong thực tế.



*Hình 1.3 Một ca phẫu thuật trong thực tại ảo*

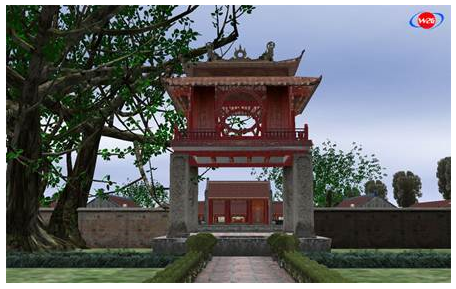
* **Trong khoa học kỹ thuật**

Với sự trợ giúp của thực tại ảo, ngày nay, con người không những có thể xem được hình ảnh trực quan của các thiết bị cần sản xuất mà thậm chí người ta còn có khả năng sử dụng hay thay đổi các chi tiết của các thiết bị đó. Việc này nhằm giúp cho các nhà khoa học và các kỹ sư thuận lợi hơn trong việc tạo ra một sản phẩm hợp ý muốn mà không cần tốn nhiều chi phí.

*Hình 1.4 Các kỹ sư đang thay đổi các chi tiết cho chiếc xe hơi ảo*

* **Trong kiến trúc**

Trong nhiều năm trở lại đây, thực tại ảo đã được sử dụng để xây dựng mô hình của các dự án kiến trúc trước khi các dự án này đưa vào thực tế, nhằm giúp cho người sử dụng có cái nhìn tổng quan và chi tiết về các dự án đó. Bên cạnh đó, thực tại ảo cũng được sử dụng để tái hiện lại các công trình kiến trúc cổ, nhằm lưu giữ lại các di sản văn hóa...



*Hình 1.5 Một góc của Văn Miếu Quốc Tử Giám trên mô hình 3D*

**-  Trong quân sự**

Với việc phát triển của VR, các binh sĩ sẽ được huấn luyện một cách trực quan nhất các kĩ năng cần thiết như: lái máy bay, lái xe tăng,... trước khi tham gia công việc thực tế. Điều này vừa bảo đảm an toàn cho binh sĩ, vừa tiết kiệm được chi phí cho các khóa huấn luyện thực tế. Quân đội Mỹ đã phát triển **một game đặc biệt nhằm huấn luyện binh sĩ chống lại khủng bố dưới dạng chiến thuật thực tại ảo (Hình 1.6).** Đây sẽ là một game rất sống động, có tính hành động cao với môi trường và bối cảnh bám sát với thực tế. Những người lính sẽ phải vận dụng tất cả những kỹ năng đã được rèn giũa trong quân đội.

*Hình 1.6 Ứng dụng của VR trong quân sự*

* **Trong giáo dục**

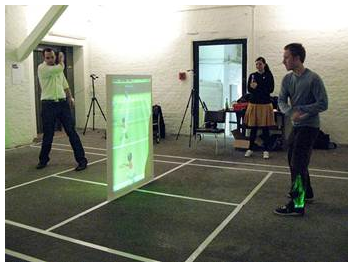
Ở các nước phương Tây việc ở nhà học qua Internet không còn là điều mới mẻ. Công nghệ VR sẽ làm cho việc này trở nên thú vị hơn rất nhiều. Người học có thể điều khiển một nhân vật đại diện cho mình đi lại trong một trường học ảo được xây dựng trên máy tính. Người học cũng có thể tham gia vào bất cứ lớp học ảo nào mà họ thích và nói chuyện với những thành viên khác trong lớp.



*Hình 1.7 Cảnh sinh hoạt trong lớp học ảo*

* **Trong giải trí**

Game thực tại ảo hiện nay đã trở thành một ngành công nghiệp thu được nhiều lợi nhuận. Ở nước ta hiện nay thì game thực tại ảo chưa được biết tới nhiều song ở một số nước phát triển thì đây là một ngành giải trí thu lợi nhuận khổng lồ, ví dụ các nước Mỹ, Nhật, Anh, ...



*Hình 1.8 Game Nintendo Wii*

Như vậy thực tại ảo có ứng dụng trong hầu hết các lĩnh vực của cuộc sống. Qua đó cũng nhận thấy được ý nghĩa to lớn của việc ứng dụng thực tại ảo. Với những vấn đề khó khăn, nếu không có thực tại ảo thì rất khó giải quyết hoặc hiệu quả không cao mà chi phí tốn kém.

## 2.3. Tập tin của VRML

Tập tin của VRML có phần mở rộng là “.wrl” với các phần như sau:

* **Header**: dùng để nhận dạng tập tin VRML và cách mã hóa. Header của file VRML bắt đầu bằng dấu #. Ngoài lần xuất hiện đầu tiên ra thì dấu # đánh dấu những gì theo sau nó là phần chú thích. File tiêu đề của VRML có dạng: *#VRML V1.0 ascii* dành riêng cho phiên bản VRML 1.0 và *#VRML V2.0 utf-8* dành cho phiên bản 2.0.
* **Scene Graph**: chứa những node mô tả các đối tượng và các thuộc tính đi kèm. Nó gần như một cây phả hệ gồm các nhóm đối tượng.
* **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).
* **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. “Event routing” cho phép một sự kiện phát sinh được truyền đến các “đích”- những nút trong hệ thống, từ đó gây ra những thay đổi cho riêng nút đó và hệ thống.

## 2.4. Các nút trong VRML

Tập tin VRML được xây dựng dựa trên tập các đối tượng nhằm đến các mục đích khác nhau. Thông thường các đối tượng có các thuộc tính vật lý của mình như hình dạng, màu sắc, tọa độ điểm, ... Để mô tả cho các đối tượng của thế giới thật, VRML sử dụng thuật ngữ “Nút - Node” để biểu diễn chúng.

Nút là khối cơ sở của tập tin VRML dùng để mô tả những đối tượng mà thuộc tính của chúng được định nghĩa trong nút đó. Nút có thể là các đối tượng hình học như hình hộp, hình nón, hình trụ … hay các đối tượng khác như màu sắc, ánh sáng, âm thanh. Sự tồn tại của nút trong tập tin VRML có thể là một cấu trúc cơ bản đứng đơn lẻ hoặc có thể chứa nhiều các nút có liên hệ với nhau.

Dữ liệu của nút được lưu giữ bởi các trường (Field) trong nút, tuy nhiên ta có thể khai báo chỉ một nút trong file nhưng không thể chỉ đưa ra một trường đơn lẻ mà bắt buộc phải để trong một nút nào đó. Về một khía cạnh nào đó nút tương đương với một lớp (class) trong các ngôn ngữ lập trình hướng đối tượng (Java). 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.

* **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.

* **Special Groups Nodes**: Nhóm các nút nhóm đặc biệt.

Inline

LOD

Switch.

* **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.

Tên các nút trong VRML thường bắt đầu bằng chữ in hoa và chỉ có thể là một trong các tên chuẩn do VRML cung cấp, các trường của nút thường bắt đầu là chữ thường, mỗi loại nút có các trường khác nhau. Giá trị của trường có thể là các giá trị thực hoặc các bộ giá trị thực hoặc có thể là một nút cơ bản. Có thể hình dung các nút như các lớp trong lập trình hướng đối tượng. VRML không cho phép định nghĩa thêm các nút mới mà chỉ được dùng các nút cơ bản trong chín nhóm nút đã nêu.

* *Nút thông tin (WorldInfo)*

Nút **WorldInfo** là nút chứa thông tin chung về thế giới ảo như tiêu đề của thế giới hay một chuỗi thông tin về tác giả hoặc về nội dung của tập tin, nút này không ảnh hưởng đến hình ảnh cũng như các sự kiện trong thế giới (nút này có thể không được sử dụng trong tập tin VRML). Các trường của nút này được thiết kế để lưu trữ tên hoặc tiêu đề của thế giới để trình duyệt có thể hiển thị cho người dùng hoặc phục vụ cho các công cụ tìm kiếm.

Sau đây là một ví dụ của nút **WorldInfo**:

|  |
| --- |
| **WorldInfo** **{**  **title** **“**Hello VRML**”**  **info** **[“**Virtual Reality Modeling Language**”**  **“**Vu Thi Mai”**]**  } |

* *Nút hình dáng (Sharp node)*

Nút **Sharp** là nút cơ sở dùng để chứa các đối tượng hình học thông qua “Geometry nodes” và các thuộc tính cần thể hiện của đối tượng hình học đó qua “Appearance nodes”. Điều đó có nghĩa nếu ta muốn tạo ra bất kỳ một đối tượng nào thì nút hình dáng của đối tượng đó phải được tạo ra trước.

Ví dụ:

|  |
| --- |
| Shape {              appearance Appearance {                          material Material { diffuseColor 0.8941 0.7216 0.6}               }             geometry Box {  size 2 2 2  }  } |

### 2.4.1. Các đối tượng hình học cơ bản

#### 2.4.1.1. Hình hộp (Box)

*Ví dụ:*

|  |
| --- |
| Shape {  geometry Box{ size 2.0 2.0 2.0 }  } |

*Tham số:*

* Ø size X Y Z: kích thước hình hộp
* X: chiều rộng. Y: chiều cao. Z: chiều sâu.

#### 2.4.1.2. Hình nón (Cone)

*Ví dụ:*

|  |
| --- |
| Shape {  geometry Cone {  height 2.0  bottomRadius 1.0  bottom TRUE  side TRUE  }  } |

*Các tham số:*

* **height** X: chiều cao của hình nón.
* **bottomRadius** Y: bán kính của đáy.
* **bottom** TRUE / FALSE: hiện / ẩn đáy.
* **side** TRUE / FALSE: hiện / ẩn mặt bên.

#### 2.4.1.3. Hình cầu (Shpere)

*Ví dụ:*

|  |
| --- |
| Shape {  geometry Sphere{ radius 1.0 }  } |

*Các tham số:*

* Ø **Radius** X: bán kính mặt cầu.

#### 2.4.1.4. Hình trụ (Cylinder)

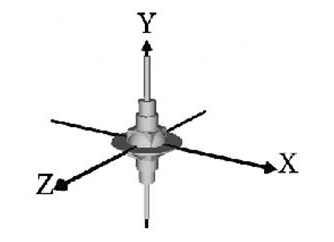
*Ví dụ:*

|  |
| --- |
| Shape {  geometry Cylinder{  height 2.0  radius 1.0  bottom TRUE  top TRUE  side TRUE  }  } |

*Các tham số:*

* **height** X: chiều cao.
* **radius** Y: bán kính mặt đáy.
* **bottom** TRUE / FALSE: hiện / ẩn mặt đáy dưới.
* **top** TRUE / FALSE: hiện / ẩn mặt đáy trên.
* **side** TRUE / FALSE: hiện / ẩn các mặt bên.

### 2.4.2. Các phép biến đổi

Mặc định, mọi vật thể và hình ảnh được xây dựng trong VRML đều được đặt ở tọa độ gốc và các phép biến đổi được dùng để tạo ra một hệ thống hệ trục tọa độ mới mà hệ trục này có vị trí tương đối so với hệ tọa độ mặc định. Có các phép biến đổi như: dịch chuyển, co giãn, quay. Ngoài ra, nếu các hình này được xây dựng trùng lên nhau như hình 2.2 có thể giúp chúng ta tạo ra một hình phức tạp mà không cần phải dùng đến nút **ElevationGrip**.

Nút **Transform** quản lý các phép biến đổi trong VRML như: dịch chuyển, co dãn, quay.

Nút **Transform** gồm có các trường con:

* **translation** (dịch chuyển)
* **rotation** (quay),
* **scale** (co giãn)
* **children :** trong đó trường **children** được dùng để chứa các đối tượng (vật thể, hình ảnh,...).

#### 2.4.2.1. Phép dịch chuyển (Translation)



*Cú pháp:*

|  |
| --- |
| Transform {                          translation X Y Z                          children []  } |

Các tham số:

* **translation** X Y Z: dịch chuyển gốc tọa độ đến điểm X Y Z.

*Ví dụ:*

|  |
| --- |
| Transform {                          translation 1 0 1                          children [  Shape {  geometry Sphere{ radius 1.0 }  }  ]  } |

#### 2.4.2.2. Phép quay (Rotation)



*Cú pháp:*

|  |
| --- |
| Transform {                          rotation X Y Z G                          children []  } |

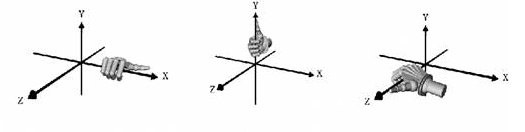
Các tham số:

* **rotation** X Y Z G: quay trục tọa độ theo trục X, Y, Z một góc G.
* G được tính theo radians.

*Ví dụ:*

|  |
| --- |
| Transform {                         rotation 1 0 0 1.57                          children [  Shape {  geometry Box{ size 2 1 0.5}  }  ]  } |

*Lưu ý:* quy định, chiều quay đối tượng được xác định bởi quy tắc bàn tay phải.



(Nắm bàn tay phải lại, choãi ngón cái ra theo như hình vẽ. Ngón tay cái chỉ phương của trục quay thì chiều từ cổ tay đến các ngón tay chi chiều dương khi quay đối tượng).

#### 2.4.2.3. Phép co giãn (Scale)



*Cú pháp:*

|  |
| --- |
| Transform {                          Scale x y z                          children []  } |

Các tham số:

* **Scale** X Y Z: co giãn đối tượng theo trục X ,Y, Z với tỷ lệ x y z.

*Ví dụ:*

|  |
| --- |
| Transform {                          Scale 3 2 0.02                          children [  Shape {  geometry Sphere{ radius 1.0 }  }  ]  } |

### 2.4.3. Màu sắc và hình ảnh

Các đối tượng trong thế giới ảo khi được tạo ra sẽ có màu mặc định là màu trắng nhưng chúng ta có thể thay đổi chúng nhờ vào nút **Appearance** được cung cấp sẵn trong VRML, nút này được sử dụng để quy định các thành phần xuất hiện trên đối tượng như màu sắc, hình ảnh, video,… Nút **Appearance** bao gồm các nút con như: **Material**, **ImageTexture**, **MovieTexture**…

#### 2.4.3.1. Màu sắc trong VRML (Material)

Nút **Material** có chức năng tạo màu cho các đối tượng trong VRML. Màu sắc trong VRML được thể hiện qua 3 tham số là R G B (Red-Green-Blue) với giá trị nằm trong khoảng từ 0.0 đến 1.0. Nút **Material** có các thuộc tính sau:

* **diffuseColor**: xác định màu sắc của đối tượng.
* **emissiveColor**: xác định màu của ánh sáng phát ra từ đối tượng.
* **transparency**: xác định độ trong suốt của đối tượng (0: hiển thị rõ, 1: trong suốt).
* **ambientIntensity**: xác định lượng phản chiếu ánh sáng của đối tượng.
* **specularColor**: xác định màu sắc của các điểm sáng bóng trên bề mặt của đối tượng.
* **shininess**: điều chỉnh cường độ sáng cho những điểm sáng bóng.

*Ví dụ:*

|  |
| --- |
| **Shape** **{**  **appearance** **Appearance** **{**  **material** **Material** **{**  **diffuseColor** 1.0 0.5 0.0  **emissiveColor** 0.0 0.0 0.0  **transparency** 0.5  **specularColor** 0.5 0.5 0.5  **shininess** 0.3  **ambientIntensity** 0.4  **}**  **}**  **geometry** **Box** **{** **size** 1.0 1.0 1.0**}**  } |

#### 2.4.3.2. Vẽ trên hình đối tượng (PixelTexture)

Ngoài việc tô màu cho một đối tượng nào đó, trong VRML còn cho phép ta vẽ hình trên bề mặt đối tượng đó, hình vẽ được tạo ra từ tập hợp các điểm ảnh. Nút PixelTexture sẽ giúp ta thực hiện công việc này.

*Ví dụ:*

|  |
| --- |
| Shape {  appearance Appearance {                          material Material {}                          texture PixelTexture {  image 2 4 3 0xFF0000 0xFF00 0 0 0 0 0xFFFFFF 0xFFFF00  repeatS TRUE  repeatT TRUE                          }  }              geometry Cylinder {height 5}  } |

*Các thuộc tính*:

* **image**: xây dựng một hình vẽ từ tập hợp các điểm ảnh.
* **repeatS và repeatT**: hình được lập đi lập lại theo chiều dọc và chiều ngang.

Ở trong ví dụ trên ta vẽ một bức hình gồm 8 điểm ảnh (mỗi điểm có các màu sắc được chỉ ra bởi các số thập lục phân trong ví dụ ở trên), sau đó bức hình được kéo giãn ra bằng với kích thước của hình trụ và dán lên hình trụ đó.

#### 2.4.3.3. Dán hình đối tượng (ImageTexture)

Thay vì phải mất thời gian để vẽ hình lên đối tượng như trên ta có thể lấy một bức ảnh có sẵn dán lên bề mặt đối tượng như là dán decal. Kĩ thuật này được gọi là ánh xạ kết cấu (texture mapping).

Trong VRML có thể sử dụng một trong các định dạng ảnh sau đây:

* **GIF**: 8bit, có hỗ trợ trong suốt, nhưng không phải là sự lựa chọn tốt cho “texture mapping” vì hình ảnh sẽ bị “bể” khi dán lên đối tượng.
* **JPEG**: từ 8-16 bit, không hỗ trợ trong suốt, là sự lựa chọn tốt cho “texture mapping”.
* **PNG**: từ 8-16 bit, hỗ trợ trong suốt trên từng điểm ảnh, là sự lựa chon tốt nhất cho “texture mapping”.

*Ví dụ:*

|  |
| --- |
| Shape {    appearance Appearance {              material Material {}              texture ImageTexture {  url “TV.jpg”  repeatS TRUE  repeatT TRUE  }    }    geometry Box { size 3 4 5}  } |

*Các thuộc tính*:

Ở ví dụ này chúng ta lấy hình ảnh “hinha.jpg” từ bên ngoài thông qua thuộc tính **url** dán lên đối tượng, các thuộc tính **repeatS** và **repeatT** tương tự như nút **PixelTexture**.

#### 2.4.3.4. Chiếu phim lên bề mặt đối tượng (MovieTexture)

Ngoài việc cho phép vẽ hay dán hình lên đối tượng, VRML còn hỗ trợ việc chiếu một tập tin video trên đối tượng.

VRML chỉ hỗ trợ định dạng video MPEG (bao gồm MPEG1-Systems (âm thanh và hình ảnh) và MPEG1-Video (chỉ có hình ảnh)).

*Cú pháp:*

|  |
| --- |
| MovieTexture {              loop      FALSE              speed      1.0  startTime    0              stopTime    0              url       []              repeatS     TRUE              repeatT     TRUE  } |

*Các thuộc tính*:

* **loop**: xác định tính lập lại của bộ phim.
* **speed**: xác định tốc độ chiếu phim, ví dụ speed có giá trị là 2 thì tốc độ chiếu phim sẽ nhanh gấp hai lần.
* **startTime và stopTime**: xác định khoảng thời gian chiếu và dừng phim dựa trên thời gian thực.
* **url**: tập các liên kết đến các tập tin video định trình chiếu.

*Ví dụ:*

|  |
| --- |
| Shape {              appearance Appearance {                          material Material {}                          texture MovieTexture {                          url “phim.mp4”                          loop TRUE                          }              }              geometry Cylinder {height 5}  } |

### 2.4.4. Âm thanh, ánh sáng, camera và phong cảnh

#### 2.4.4.1. Âm thanh

Để làm cho thế giới ảo trở nên sinh động và hấp dẫn hơn, chúng ta nên thêm âm thanh vào đó. Ta có thể tạo ra âm thanh nền, tiếng chuông cửa, âm thanh khi mở cửa hoặc bất cứ âm thanh nào mà mình muốn có trong thế giới ảo. Tất cả những điều đó được thực hiện bởi hai nút do VRML cung cấp là nút **Sound** và nút **AudioClip**. Có thể hiểu đơn giản như sau: nút **Sound** được dùng để xác định vị trí phát ra âm thanh trong thế giới ảo và âm thanh đó được xác định bởi nút **AudioClip**.

#### 2.4.4.2. Ánh sáng

Khi quan sát một vật thể bất kỳ, các vật thể đó chỉ được chiếu sáng bởi một ánh sáng xung quanh. Đây là một ánh sáng đặc biệt được tạo ra bởi trình duyệt. Tuy nhiên, chúng ta cũng có thể điều khiển ánh sáng này thông qua nút **NavigationInfo**, đây là một nút được sử dụng để xác định các thuộc tính của người sử dụng. Tuy nhiên để có được một thế giới ảo trông thật hơn thì chúng ta nên tạo ra các nguồn sáng như trong thế giới thật, ví dụ như: ánh sáng mặt trời, ánh sáng của đèn điện,… Và VRML hỗ trợ cho chúng ta các loại ánh sáng đó thông các nút **DirectionalLight**, **PointLight**, **SpotLight**.

#### 2.4.4.3. Camera

Khi ta mở một tập tin VRML thì trình duyệt sẽ hỗ trợ mặc định cho chúng ta một góc nhìn, để tùy biến góc nhìn đó ta có thể sử dụng nút **Viewpoint**. Nút **Viewpoint** có chức năng xác định vị trí của người dùng trong thế giới và các thông số của góc nhìn. Với nút này chúng ta có thể tùy biến vị trí và các đặc tính quan sát của người dùng khi thế giới được khởi động lên, việc đó sẽ giúp cho người xem như đang ở trong thế giới thực. Nút này có các thuộc tính như sau:

* **fieldOfView**: xác định một góc trong “radians”. Giá trị của trường này là một số thực nằm trong khoảng từ 0 đến 3,142 và có giá trị mặc định là 0,785398. Góc này sẽ tương đương với ống kính máy quay (camera) nếu có giá trị nhỏ và sẽ tương đương với ống kính góc rộng nếu có giá trị lớn (chẳng hạn có giá trị 3,14), tuy nhiên lúc đó góc nhìn sẽ bị méo.
* **position**: xác định vị trí của người dùng trong thế giới ảo. Có giá trị là một điểm trong không gian.
* **orientation**: Xác định biên độ quay của góc quan sát, trường này có giá trị là kiểu SFRotation (xem trong phần kiểu dữ liệu đã trình bày ở trên).
* **description**: cung cấp một đoạn mô tả cho góc quan sát.
* **jump**: cho phép sự chuyển giữa các góc nhìn trong thế giới, nếu có giá trị TRUE thì cho phép người di chuyển theo góc nhìn, còn nếu có giá trị là FALSE thì chỉ đơn thuần là sự chuyển đổi góc nhìn mà không hề ảnh hưởng đến vị trí của người dùng.

*Ví dụ:*

|  |
| --- |
| Viewpoint {  fieldOfView 0.785398  position 0 0 10  orientation 0 0 1 0  description “Góc nhìn số 1”  jump TRUE  } |

#### 2.4.4.4. Phong cảnh nền và nôi trường

Việc xây dựng một thế giới ảo gồm tất cả các đối tượng giống như với thế giới thực là chuyện không thể vì ta không có đủ thời gian để thực hiện việc đó. Vì thế chúng ta chỉ có thể xây dựng một thế giới ảo giống một phần (khu vực) nào đó của thế giới thực. Với các khu vực xung quanh, nếu chúng ta để trống các khu vực này thì thế giới của chúng ta sẽ không giống như thực được. Đây thật sự là một vấn đề khó khăn, và VRML cung cấp cho chúng ta một công cụ giúp che phủ các phần còn lại đó để thế giới của chúng ta trông thực hơn. Nút **Background** được cung cấp nhằm mô tả các đường chân trời của bạn, nó cho phép chúng ta xác định bầu trời, mặt đất và các hình ảnh xung quanh thế giới của chúng ta.

### 2.4.5. Nhóm đối tượng

#### 2.4.5.1. Anchor

**Anchor**: nút này có chức năng xác định một tập hợp các đối tượng và gắn một siêu liên kết đến một url, chẳng hạn như là một siêu liên kết đến thế giới VRML khác, đến một trang HTML hoặc đến một dữ liệu nào đó mà trình duyệt có thể đọc được. Khi kích chuột vào một trong các đối tượng bên trong nút Anchor thì toàn bộ trình duyệt sẽ đưa đến địa chỉ url. Khi đưa chuột lên một đối tượng nào đó trong nút này thì có thể nhìn thấy được dòng ghi chú của nó. Tất cả các nút trong nút **Anchor** đều hiển thị.

*Ví dụ:*

|  |
| --- |
| Anchor {  children [  Shape {  geometry Sphere {}  }  ]  url “http://ninkuhack.blogspot.com/”  description “Blog của tôi!”  parameter [“target=my\_frame” ]  bboxCenter 0 0 0  bboxSize -1 -1 -1  } |

*Các thuộc tính:*

* **children**: chứa các nút con trong nhóm.
* **url**: xác định địa chỉ của siêu liên kết. Có thể có nhiều liên kết đặt trong url, khi đó bộ trình duyệt sẽ tìm kiếm dữ liệu trong các liên kết này theo thứ tự ưu tiên giảm dần.
* **parameter**: cung cấp thêm thông tin cho trình duyệt. Ví dụ ta có thể xác định cửa sổ mà url hiển thị. Như ví dụ trên là hiện url trên một cửa sổ khác.
* **description**: ghi chú, dòng ghi chú này sẽ hiển thị khi người dùng di chuột qua đối tượng chứa trong nút **Anchor**.
* **bboxCenter**: xác định tâm của một hình hộp bao quanh các nút trong nhóm. Giá trị của trường này là một điểm trong không gian.
* **bboxSize**: xác định kích thước của hình hộp trên bao quanh các nút trong nhóm. Mặc định trường này có giá trị -1 -1 -1 tức hình hộp không được xác định. Các giá trị thành phần của trường này phải lớn hơn hoặc bằng 0. Nếu các nút con không nằm trong hình hộp thì hình hộp là không xác định.

#### 2.4.5.2. Group

**Group**: tạo ra một tập hợp các đối tượng như là một thực thể duy nhất. Nút này có các trường: **children**, **bboxCenter**, **bboxSize**. Các trường này tương tự như của nút **Anchor**. Tất cả các nút con nằm trong nút **Group** đều hiển thị.

*Ví dụ:*

|  |
| --- |
| Group {  children […]  bboxCenter 0 0 0  bboxSize -1 -1 -1  } |

#### 2.4.5.3. Switch

**Switch**: tạo ra một nhóm chuyển đổi gồm tập hợp các nút con, chỉ có một nút con trong nhóm được hiển thị và nút này là do người dùng lựa chọn. Các nút con được đánh số thứ tự từ 0, nếu trường **whichChoice** có giá trị -1 tức là không chọn nút con nào.

*Ví dụ:*

|  |
| --- |
| Switch {  whichChoice -1  choice […]  } |

*Các thuộc tính:*

* **whichChoice**: lựa chọn một nút con trong nhóm.
* **Choice**: chứa tập các nút con trong nhóm.

#### 2.4.5.4. Transform

**Transform**: tạo ra một nhóm các đối tượng và đặt chúng tại hệ trục tọa độ mới.

#### 2.4.5.5. Inline

**Inline**: tạo ra một nhóm các đối tượng đặc biệt được lấy ra từ một tập tin VRML khác (được chỉ ra sau **url**). Tất cả các đối tượng trong tập tin này sẽ được hiển thị. Nút này thường được sử dụng để gọi trực tiếp một đối tượng bên ngoài vào thế giới hiện tại.

*Ví dụ:*

|  |
| --- |
| Inline {  url [“cuaso.wrl”]  bboxCenter 0 0 0  bboxSize -1 -1 -1  } |

*Các thuộc tính:*

* **url**: chứa tập đường dẫn của các tập tin VRML ngoài thế giới hiện tại.

### 2.4.6. Tái sử dụng đối tượng

#### 2.4.6.1. Inline

**Inline**: sử dụng nút **Inline** để gọi trực tiếp một đối tượng bên ngoài tập tin vào tập tin hiện tại. Nút này đã được trình bày ở trên.

#### 2.4.6.2. DEF

**DEF** (DEFine): cho phép ta định nghĩa một đối tượng hay một kiểu thuộc tính.

*Ví dụ:*

Tạo đối tượng

|  |
| --- |
| Shape {  appearance Appearance {  material DEF RedColor Material {  diffuseColor 1.0 0.0 0.0                          }              }              geometry Box { size 5 5 5}  } |

Sử dụng:

|  |
| --- |
| Shape {              appearance Appearance {                          material USE RedColor              }              geometry Sphere { }  } |

Như ví dụ trên ta định nghĩa đối tượng RedColor kiểu **Material**. Sau đó muốn sử dụng lại ta chỉ gọi hàm bằng từ khóa **USE**.

### 2.4.7. Nội suy và cảm biến

#### 2.4.7.1. Các nút nội suy (Interpolators nodes)

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:

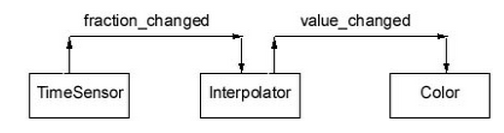
* ColorInterpolator
* CoordinateInterpolator
* NormalInterpolator
* OrientationInterpolator
* PositionInterpolator
* ScalarInterpolator

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**.

Các nút **Interpolators** thường được kết hợp với **TimeSensor** để tạo ra các hoạt họa. Xem hình vẽ dưới đây để hiểu rõ hơn về các nút **Interpolators.**



Như trong hình vẽ là một mô tả về nút **ColorInterpolator** (các nút khác cũng tương tự như vậy). Vậy khi nút TimeSensor được kích hoạt thì nó sẽ gửi các giá trị liên tục cho các nút **Interpolators** và các nút này cũng sẽ gửi liên tục các giá trị cho một nút khác, theo hình vẽ thì nút **ColorInterpolators** đã gửi liên tục các giá trị màu sắc đến đối tượng nào đó trong thế giới.

*Ví dụ:*

|  |
| --- |
| OrientationInterpolator {                 key [ 0.0, 0.50, 1.0 ]                 keyValue [                             0.0 1.0 0.0  0.0,                             0.0 1.0 0.0  3.14,                             0.0 1.0 0.0  6.28                 ]  } |

Trong ví dụ này nếu nút này nhận một giá trị là 0.0 thì nó sẽ gửi trả ra giá trị (0.0 1.0 0.0 0.0) và nếu nhận giá trị từ khoảng 0.0 đến 0.5 thì nó sẽ tính toán và gửi ra giá trị nằm trong khoảng 0.0 đến 3.14 (chính xác là ở đây nếu nó nhận được giá trị là 0,3 thì nó sẽ trả ra giá trị là 1,88), và tương tự với các giá trị khác.

#### 2.4.7.2. Nhúng mã vào VRML (Scipt)

Nút **Script** mở ra khả năng linh hoạt, mềm dẻo để mở rộng gần như vô hạn các yêu cầu hành động của chúng ta cũng như hành động mong đợi phản hồi từ thế giới ảo. Nút **Script** cho phép chúng ta xác định việc tương tác với thế giới ảo bằng cách sử dụng các ngôn ngữ lập trình như Java, Javascript hoặc VRMLScript (được đưa ra mới đây bởi Silicon Graphics, Inc).

Trong nút **Script** có thể có các trường hay các thuộc tính, nhưng không giống với các nút khác, trong nút **Script**, chúng ta có thể xác định các sự kiện mà thẻ này có thể nhận và gửi. Một nút **Script** sẽ thực hiện một hành động nào đó mỗi khi nó nhận một sự kiện, trong mỗi hành động đó nút **Script** có thể tạo ra nhiều sự kiện. Hơn nữa nút **Script** có thể được sử dụng để xây dựng các thủ tục khởi tạo (initiazation procedures), dừng và tắt (shutdown procedures) các đối tượng trong thế giới ảo.

Một nút **Script** có thể nhận nhiều sự kiện, các giá trị mà các sự kiện này tạo ra là các biến thuộc loại **eventIn**. Tên của các biến này là tên của các sự kiện mà nút **Script** tiếp nhận. Các biến này là chỉ có thể đọc, nghĩa là chúng ta không thể thiết lập giá trị trực tiếp cho bất kỳ biến nào thuộc loại này. Một nút **Script** cũng có thể tạo ra nhiều sự kiện, các giá trị mà các sự kiện này tạo ra là các biến thuộc loại **eventOut**.

*Cú pháp:*

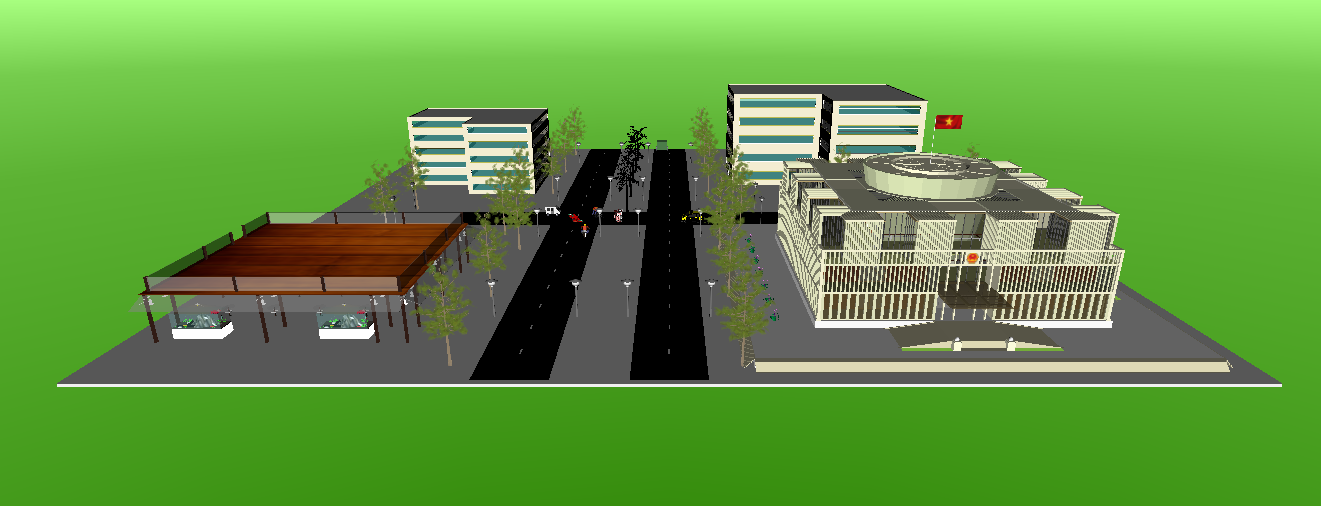
|  |
| --- |
| Script  {  url []  directOutput FALSE  mustEvaluate FALSE  eventIn Datatype EventName  eventOut Datatype EventName  field Datatype FieldName InitialValue  } |

*Các tham số:*

* **url**: chỉ ra chương trình script được xử lí, giá trị của trường này có thể là một đường dẫn hoặc là một đoạn lệnh. Các ngôn ngữ thông dụng hỗ trợ cho việc xây dựng các chương trình script này là: JAVA, JavaScript, VRMLScript.
* **directOutput**: nếu có giá trị là FALSE thì nút này có thể đọc các giá trị gửi đi của bất kỳ nút nào có thành phần giao tiếp và cũng gửi trả các giá trị cho bất cứ nút nào chờ nhận sự kiện. Ngược lại nếu có giá trị TRUE thì nút **Script** có thể tạo ra các giá trị khác nhau từ các giá trị gửi đi của các nút khác nhau và gửi trả các giá trị đó cho các nút đang chờ nhận sự kiện.
* **mustEvaluate**: nếu giá trị bằng FALSE thì trình duyệt sẽ chọn một thời điểm thuận lợi để xử lý các sự kiện, còn nếu bằng TRUE thì các sự kiện sẽ được xử lý ngay khi chúng được tiếp nhận.

# CHƯƠNG 3: HÌNH ẢNH SẢN PHẨM SAU KHI MÔ PHỎNG MÔ HÌNH VA CHẠM GIAO THÔNG

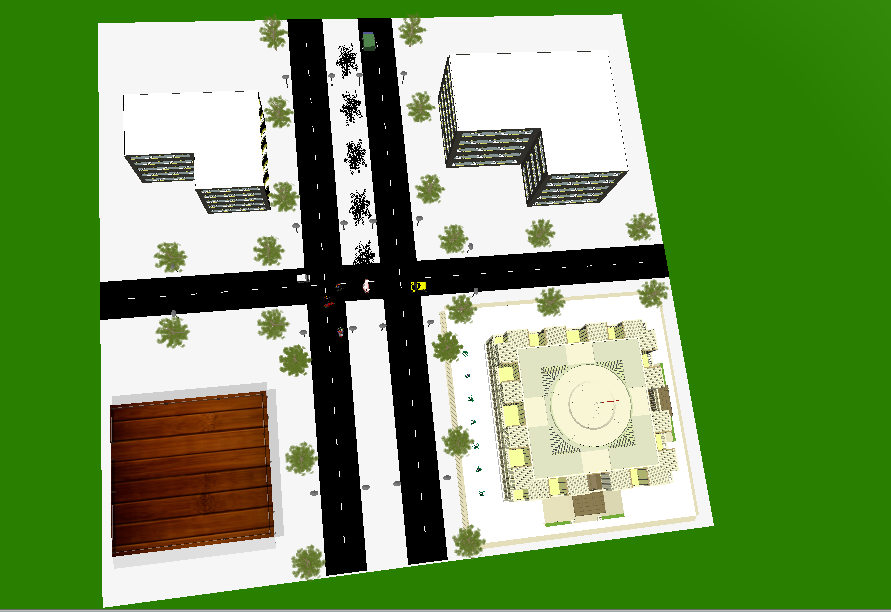
## 3.1. Hình ảnh tổng quan mô hình



*Hình 3.1.1 Tổng quan mô hình nhìn từ trước*



*Hình 3.1.2 Tổng quan mô hình nhìn từ phía sau*

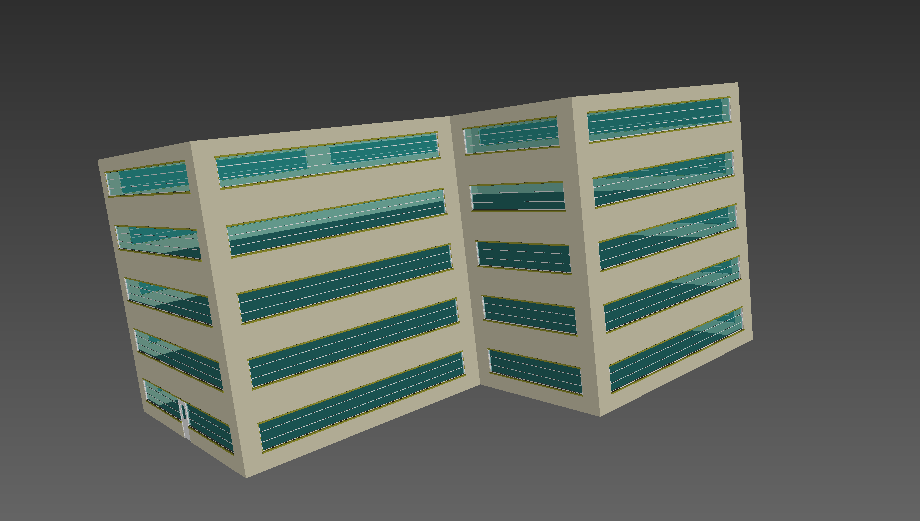


*Hình 3.1.2 Tổng quan mô hình nhìn từ trên xuống*

## 3.2. Các thành phần trong mô hình

### 3.2.1. Tổng quan mô hình va chạm

* Nhà cao tầng



* Quán café



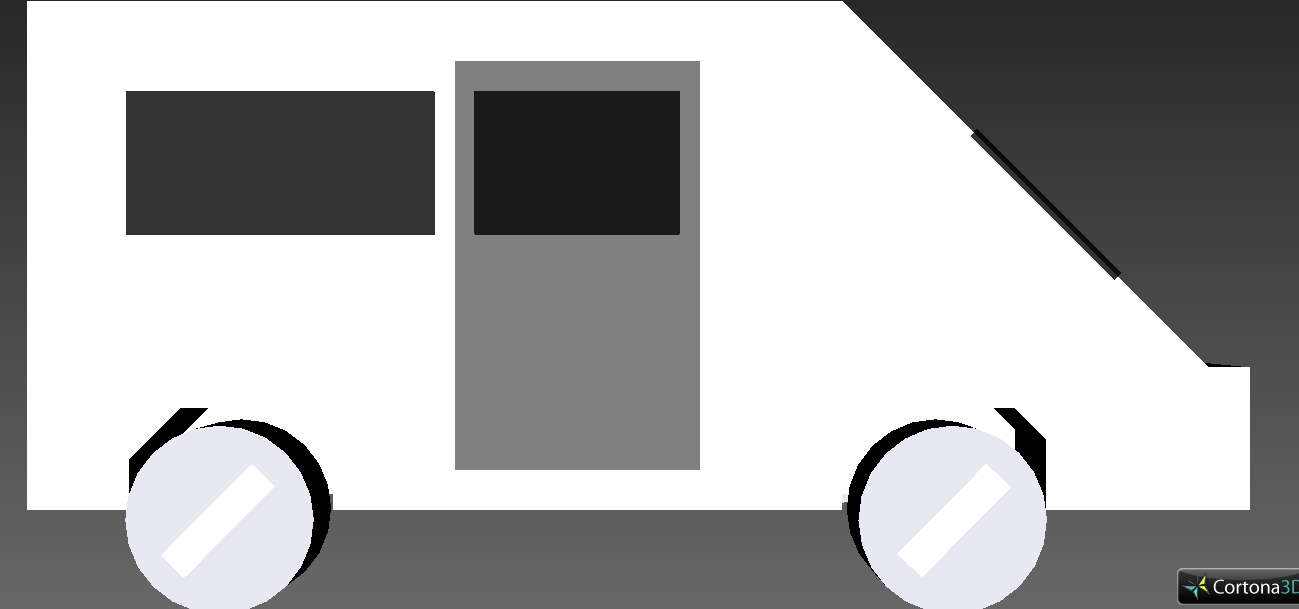
* Nhà quốc hội

****

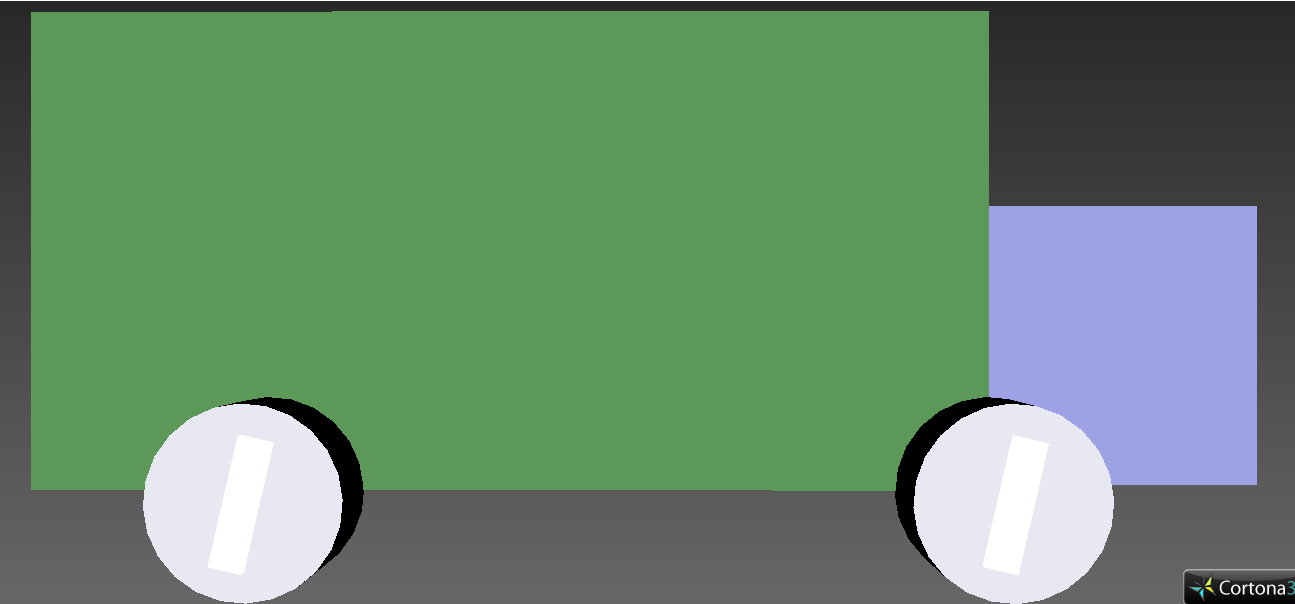
* Taxi



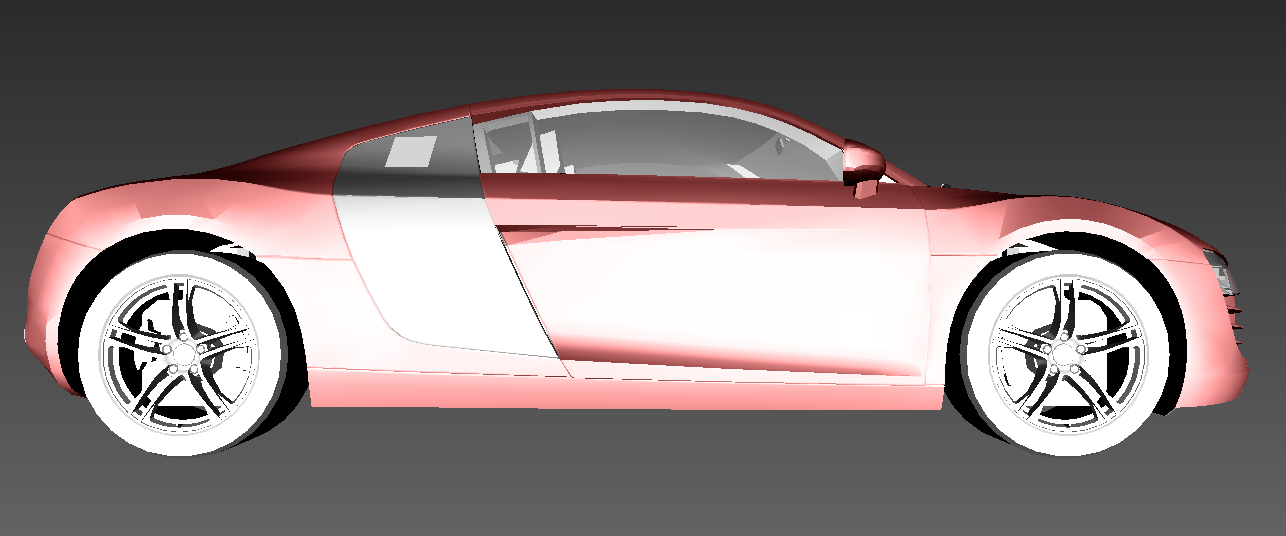
* Xe thùng



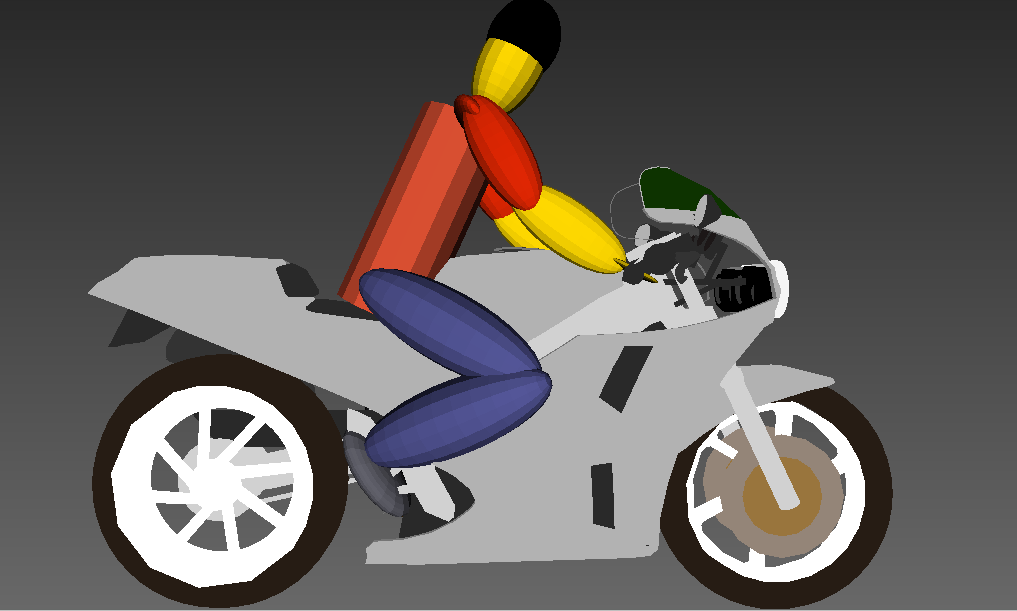
* Xe tải



* Xe ô tô đỏ



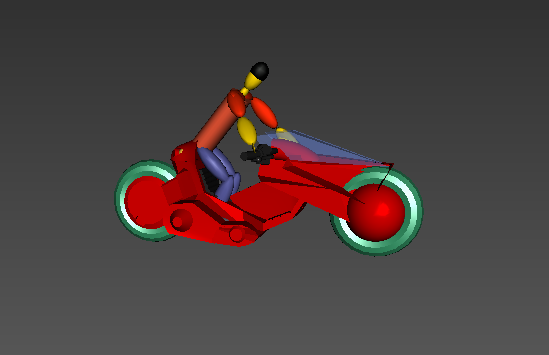
* Xe máy Kawasaki



* Xe máy Lead



* Xe máy

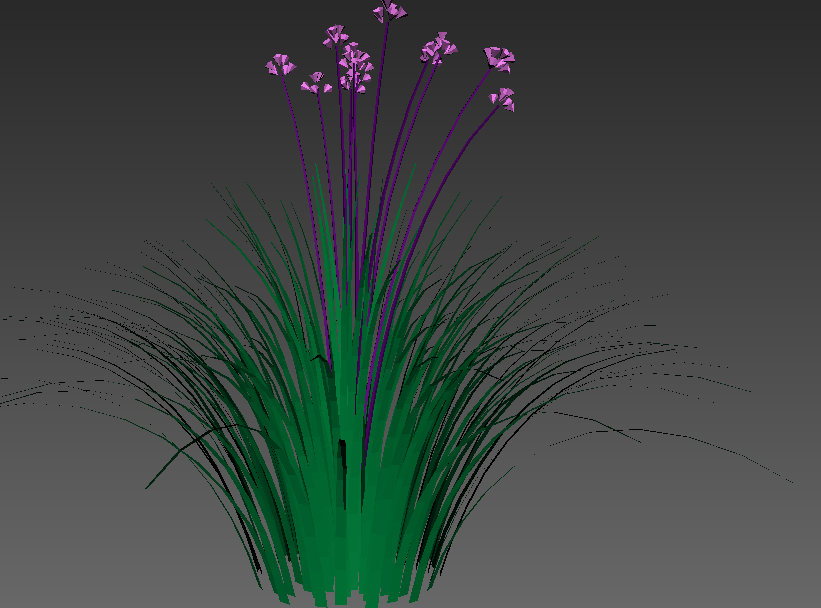


### 3.2.2. Một số vật thể trong mô hình

* Cây dừa



* Cây bụi



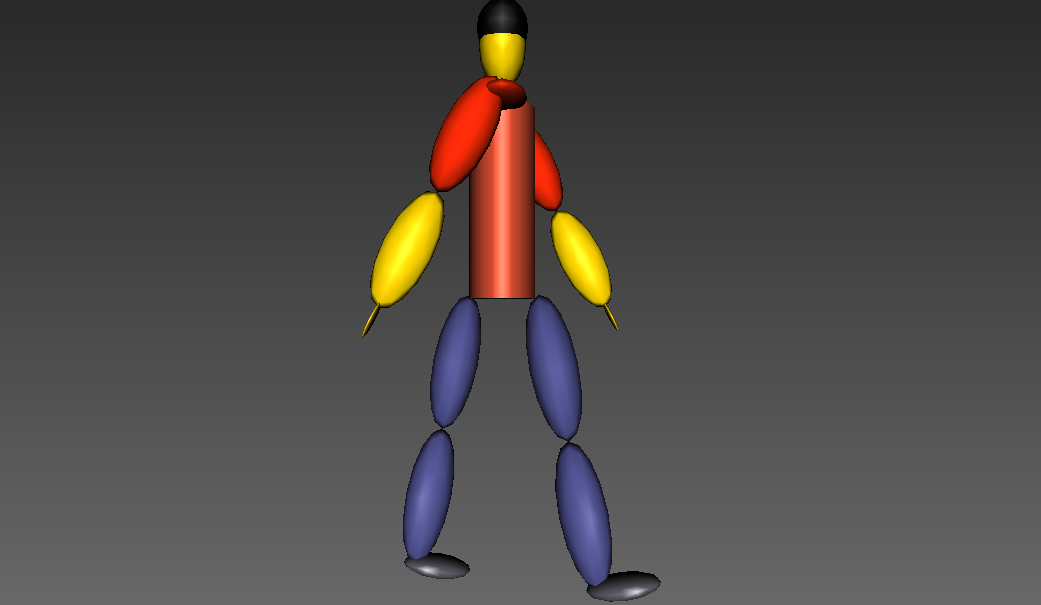
* Cây 2 bên đường



* Đèn đường



* Người



# 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>

[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>

[4] <http://text.123doc.org/document/7500-ngon-ngu-mo-hinh-hoa-thuc-tai-vrml.htm>

[5] <http://hoidapit.com.vn/Questions/ViewQuestions/1048/chia-se-bo-tai-lieu-hoc-thuc-tai-ao-bang-ngon-ngu-vrml.html>

[6] [*http://www.3dcadbrowser.com/*](http://www.3dcadbrowser.com/)

[7][*http://www.lighthouse3d.com/vrml/tutorial/*](http://www.lighthouse3d.com/vrml/tutorial/)

[8][*http://vrmlworks.crispen.org/models.html*](http://vrmlworks.crispen.org/models.html)