Lời cảm ơn

Trước hết em xin chân thành thầy Ngô Trường Giang là giáo viên hướng dẫn em trong quá tình thực tập. Thầy đã giúp em rất nhiều và đã cung cấp cho em nhiều tài liệu quan trọng phục vụ cho quá trình tìm hiểu về đề tài "Tìm hiểu một số kỹ thuật trong đồ họa 3D và ứng dụng".

Thứ hai, Em xin chân thành cảm ơn các thầy cô trong bộ môn công nghệ thông tin đã chỉ bảo bảo em trong quá trình học và rèn luyện trong 4 năm học vừa qua. Đồng thời em cảm ơn các bạn sinh viên lớp CT901 đã gắn bó với em trong quá trình rèn luyện tại trường.

Cuối cùng em xin chân thành cảm ơn ban giám hiệu trường Đại Học Dân Lập Hải Phòng đã tạo điều kiện cho em có kiến thức, thư viện của trường là nơi mà sinh viên trong trường có thể thu thập tài liệu trợ giúp cho bài giảng trên lớp. Đồng thời các thầy cô trong trường giảng dạy cho sinh viên kinh nghiệm cuộc sống. Với kiến thức và kinh nghiệm đó sẽ giúp em cho công việc và cuộc sống sau này.

Em xin chân thành cảm ơn!

Hải Phòng, tháng 07 năm 2009

Sinh viên

Nguyễn Phi Hùng

MỤC LỤC

Mở đầu

Ngày nay công nghệ thông tin và đặc biệt là các ứng dụng đồ họa 3 chiều ngày càng phát triển mạnh mẽ. Ứng dụng phổ biến nhất của đồ họa 3 chiều chính là Game – lĩnh vực công nghệ thông tin mang lại nhiều lợi nhuận nhất, ngoài ra là một số các lĩnh vực khác như là y học, xây dựng… Với mong muốn tiếp cận và phát triển lĩnh vực đồ họa 3 chiều để giải quyết một số bài toán trong thực tế, em đã tìm hiểu về thư viện đồ họa mã nguồn mở Ogre.

Bài toán thực tế đặt ra cho một công ty xây dựng là: công ty muốn bán một hoặc nhiều căn nhà nhưng chúng còn đang được xây dựng, thế nên người mua không thể xem trước căn nhà của họ được. Để giải quyết bài toán này ta sẽ dùng Ogre để đồ họa 1 căn nhà 3D và cho phép người dùng có thể đi lại và xem toàn bộ nội thất của căn nhà trong tương lai.

Đồ án được chia làm 4 chương:

- Chương 1: Đưa ra cái nhìn tổng quan về kỹ thuật đồ họa.
- Chương 2: Tìm hiểu kỹ hơn về một số kỹ thuật ứng dụng trong đồ họa 3D.
- Chương 3: Tìm hiểu tổng quan về về thư viện đồ họa mã nguồn mở Ogre.
- Chương 4: Xây dựng chương trình để giải quyết bài toán đặt ra.

CHƯƠNG 1: Tổng quan về kỹ thuật đồ họa

1.1 Các khái niệm tổng quan của kỹ thuật đồ họa máy tính

Definition (ISO): Phương pháp và công nghệ chuyển đổi dữ liệu từ thiết bị đồ hoạ sang máy tính.

Computer Graphics là phương tiện đa năng và mạnh nhất của giao tiếp giữa con người và máy tính.

Computer Graphics (Kỹ thuật đồ hoạ máy tính) là một lĩnh vực của Công nghệ thông tin mà ở đó nghiên cứu, xây dựng và tập hợp các công cụ (mô hình lý thuyết và phần mềm) khác nhau: kiến tạo, xây dựng, luu trữ, xử lý các mô hình (model) và hình ảnh (image) của đối tượng. Các mô hình (model) và hình ảnh này có thể là kết quả thu được từ những lĩnh vực khác nhau của rất nhiều ngành khoa học (vật lý, toán học, thiên văn học…)

Computer graphics xử lý tất cả các vấn đề tạo ảnh nhờ máy tính.

1.2 Các kỹ thuật đồ họa

1.2.1 Kỹ thuật đồ họa điểm

- Các mô hình , hình ảnh của các đối tượng được hiển thị thông qua từng pixel (từng mẫu rời rạc).
- Đặc điểm: có thể thay đổi thuộc tính
  - Xoá đi từng pixel của mô hình và hình ảnh các đối tượng.
  - Các mô hình hình ảnh được hiển thị như một lưới điểm (grid) các pixel rời rạc.
  - Từng pixel đều có vị trí xác định, được hiển thị với một giá trị rời rạc (số nguyên) các thông số hiển thị (màu sắc hoặc độ sáng)

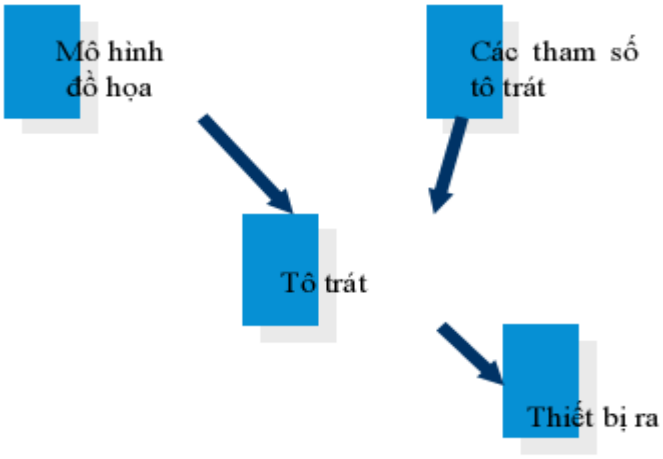
+ Tập hợp tất cả các pixel của grid cho chúng ta mô hinh, hình ảnh đối tượng mà chúng ta muốn hiển thị.

Hình 1.1 Ảnh đồ họa điểm

Phương pháp để tạo ra các pixel

- Phương pháp dùng phần mềm để vẽ trực tiếp từng pixel một.
- Dựa trên các lý thuyết mô phỏng (lý thuyết Fractal, v.v) để xây dựng nên hình ảnh mô phỏng sự vật.
- Phương pháp rời rạc hóa (số hóa) hình ảnh thực của đối tượng.
- Có thể sửa đổi (image editing) hoặc xử lý (image processing) mảng các pixel thu được theo những phương pháp khác nhau để thu được hình ảnh đặc trưng của đối tượng.

1.2.2 Kỹ thuật đồ họa vector



Hình 1.2 Mô hình đồ họa vector

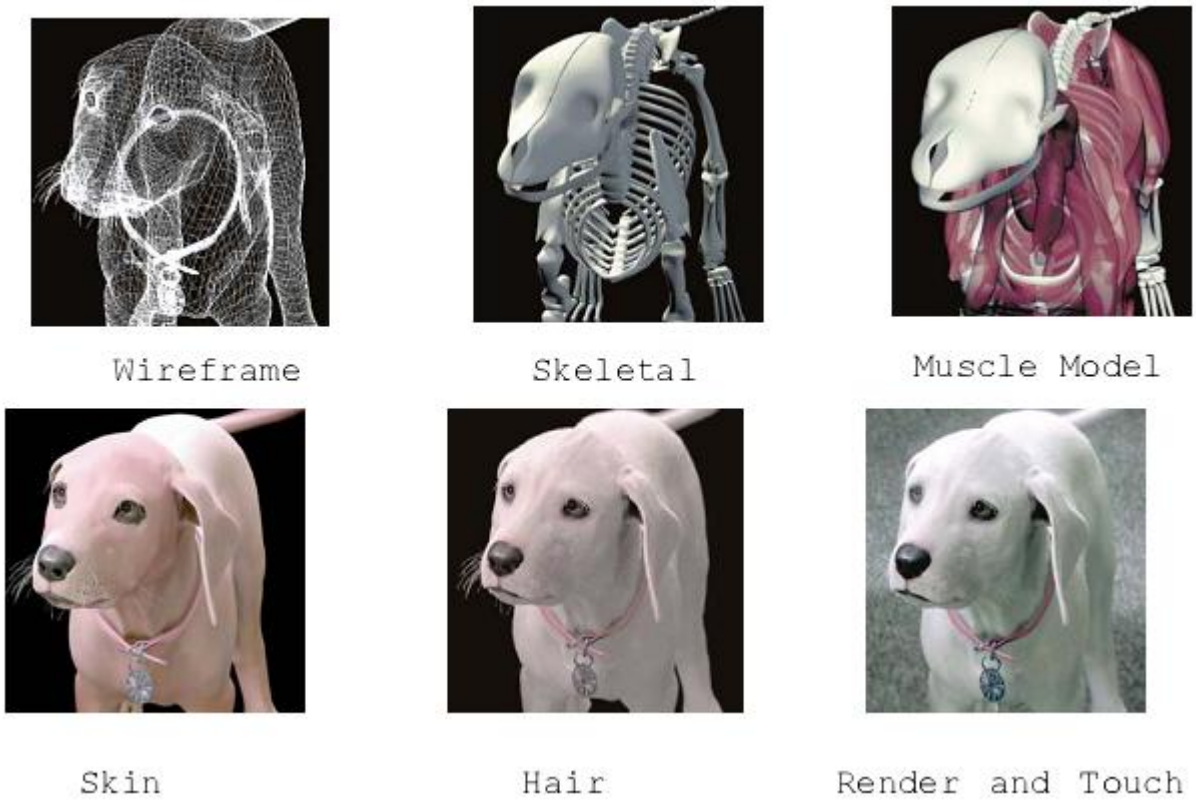
- Mô hình hình học (geometrical model) cho mô hình hoặc hình ảnh của đối tượng.
- Xác định các thuộc tính của mô hình hình học này.
- Quá trình tô trát (rendering) để hiển thị từng điểm của mô hình, hình ảnh thực của đối tượng.

Có thể định nghĩa đồ họa vector: Đồ họa vector = geometrical model + rendering.

So sánh đồ họa điểm và đồ họa vector

Đồ họa điểm(Raster Graphics)	Đồ hoạ vector(Vector Graphics)
- Hình ảnh và mô hình của các vật thể được biểu diễn bởi tập hợp các điểm của lưới (grid) - Thay đổi thuộc tính của các pixel $\Rightarrow$ thay đổi từng phần và từng cùng của hình ảnh. - Copy được các pixel từ một hình ảnh này sang hình ảnh khác.	- Không thay đổi thuộc tính của từng điểm trực tiếp - Xử lý với từng thành phần hình học cơ sở của nó và thực hiện quá trình tô trát và hiển thị lại. - Quan sát hình ảnh và mô hình của hình ảnh và sự vật ở nhiều góc độ khác nhau bằng các thay đổi điểm nhìn và góc nhìn.

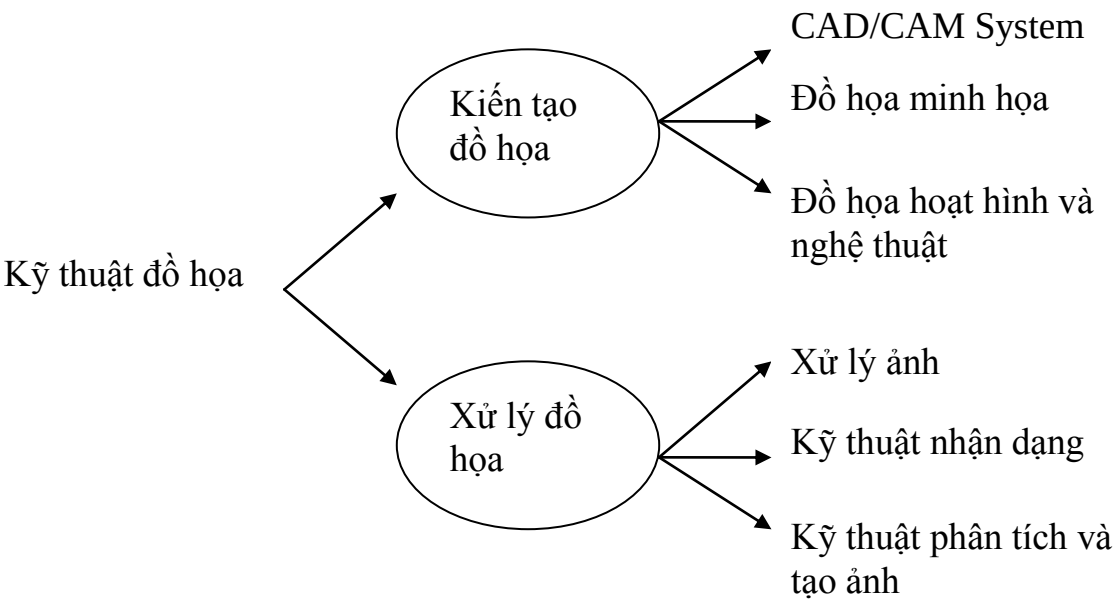
Ví dụ về hình ảnh đồ họa vector



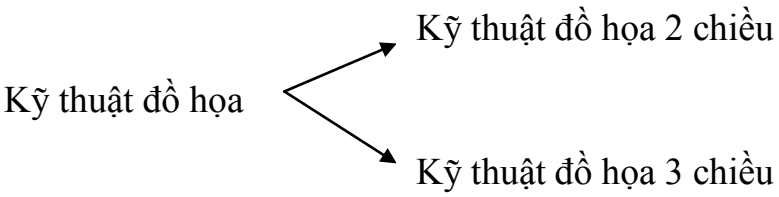
Hình 1.3 Ví dụ về đồ họa vector

1.2.3 Phân loại của đồ họa máy tính

Phân loại theo các lĩnh vực hoạt động của đồ họa máy tính



Phân loại theo hệ tọa độ



- *Kỹ thuật đồ họa 2 chiều*: là kỹ thuật đồ họa máy tính sử dụng hệ tọa độ hai chiều (hệ tọa độ thẳng), sử dụng rất nhiều trong kỹ thuật xử lý bản đồ, đồ thị.
- *Kỹ thuật đồ họa 3 chiều*: là kỹ thuật đồ họa máy tính sử dụng hệ tọa độ ba chiều, đòi hỏi rát nhiều tính toán và phức tạp hơn nhiều so với kỹ thuật đồ họa hai chiều.

Các lĩnh vực của đồ họa máy tính:

- Kỹ thuật xử lý ảnh (*Computer Imaging*): sau quá trình xử lý ảnh cho ta ảnh số của đối tượng, Trong quá trình xử lý ảnh sử dụng rất nhiều các kỹ thuạt phức tạp: kỹ thuật khôi phục ảnh, kỹ thuật làm nối ảnh, kỹ thuật xác định biên ảnh.
- Kỹ thuật nhận dạng (*Pattern Recognition*): từ những ảnh mẫu có sẵn ta phân loại theo các trúc, hoặc theo các tiêu trí được xác định từ trước và bằng các thuật toán chọn lọc để cso thể phân tích hay tổng hợp cá ảnh gốc, các ảnh gốc này được lưu trong một thư viện và căn cứ vào thư viện này ta xây dựng được các thuật giải phân tích và tổ hợp ảnh.
- Kỹ thuật tổng hợp ảnh (*Image Synthesis*): là lĩnh vực xây dựng mô hình và hình ảnh của các vật thể dựa trên các đối tượng và mối quan hệ giữa chúng.

- Các hệ CAD/CAM (Computer Aided Design/Computer Aided Manufacture System): kỹ thuạt đồ họa tập hợp các công cụ, các kỹ thuật trợ giúp cho thiết kế các chi tiết và các hệ thống khác nhau: hệ thống cơ, hệ thống điện, hệ thống điện tử…
- Đồ họa minh họa (Presentation Graphics): gồm các công cụ giúp hiển thị các số liệu thí nghiệm một cách trực quan, dựa trên các mẫ đồ thị hoặc các thuật toán có sẵn.
- Đồ họa hoạt hình và nghệ thuật: bao gồm các công cụ giúp cho các họa sĩ, các nhà thiết kế phim hoạt hình chuyên nghiệp làm các kỹ xảo hoạt hình, vẽ tranh… ví dụ: phần mềm Studio, 3D Animation, 3D Studio Max.

1.2.4 Các ứng dụng tiêu biểu của kỹ thuật đồ họa

Đồ họa máy tính là một trong những linh vực lý thú nhất và phát triển nhanh nhất của tin học. Ngay từ khi xuất hiện nó đã có sức lôi cuốn mãnh liệt, cuốn hút rất nhiều người ở nhiều lĩnh vực khác nhau như khoa học nghệ thuật, kinh doanh, quản lý...Tính hấp dẫn của nó có thể được minh họa rất trực quan thông qua các ứng dụng của nó.

- Xây dựng giao diện người dùng (User Interface):

Giao diện đồ hoạ thực sự là cuộc cách mạng mang lại sự thuận tiện và thoải mái cho người dùng ứng dụng. Giao diện WYSIWYG và WIMP đang được đa số người dùng ưa thích nhừ tính thân thiện, dễ sử dụng của nó.

- Tạo các biểu đồ trong thương mại, khoa học, kỹ thuật

Các ứng dụng này thường được dùng để tóm lược các dữ liệu về tài chính, thống kê, kinh tế, khoa học, toán học… giúp cho nghiên cứu, quản lý… một các có hiệu quả.

- Tự động hóa văn phòng và chế bán điện tử

- Thiết kế với sự trợ giúp của máy tính (CAD_CAM)
- Lĩnh vực giải trí, nghệ thuật và mô phỏng
- Điều khiển các quá trình sản xuất (Process Control)
- Lĩnh vực bản đồ (Cartography)
- Giáo dục và đào tạo

CHƯƠNG 2: Một số kỹ thuật ứng dụng trong đồ họa 3D

2.1 Các phép biến đổi hình học ba chiều

2.1.1 Hệ tọa độ thuần nhất

- Hệ tọa độ thuần nhất: (Homogeneous Coordinates) : Mỗi điểm (x,y,z) trong không gian Descartes được biểu diễn bởi một bộ bốn tọa độ trong không gian 4 chiều thu gọn (hx,hy,hz,h). Người ta thường chọn h=1.
- Các phép biên đổi tuyến tính là tổ hợp của các phép biến đổi sau : tỉ lệ, quay, biến dạng và đối xứng. Các phép biến đổi tuyến tính có các tính chất sau :
  - Gốc tọa độ là điểm bất động
  - Ảnh của đường thẳng là đường thẳng
  - Ảnh của các đường thẳng song song là đường thẳng song song
  - Bảo toàn tỷ lệ khoảng cách
  - Tổ hợp các phép biến đổi có tính phân phối

Ma trận biến đổi tổng quát trong hệ tọa độ thuần nhất (4x4)

$$[T] = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ g & i & j & r \\ l & m & n & s \end{bmatrix} \text{ hay } [T] = \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & i & j & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

2.1.2 Phép tịnh tiến

$$[T(dx, dy, dz)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

$$[X'] = [X].[T(dx, dy, dz)]$$

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1].[T(dx, dy, dz)] = [x + dx \quad y + dy \quad z + dz \quad 1]$$

2.1.3 Phép tỷ lệ

$$Ts = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[X'] = [X].Ts$$

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1].Ts$$

$$= [x.Sx \quad y.Sy \quad z.Sz \quad 1]$$

Với Sx, Sy, Sz là các hệ số tỷ lệ trên các trục tọa độ

2.1.4 Phép biến dạng

- Ta có tất cả các phần tử nằm trên đường chéo chính bằng 1
- Các phần tử chiếu và tịnh tiến bằng 0

$$Tsh = \begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ g & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[X'] = [X].Tsh$$

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1].Tsh$$

$$= [x + yd + gz \quad bx + y + iz \quad cx + fy + z \quad 1]$$

2.1.5 Phép quay 3 chiều

- Quay quanh trục Oz

$$[Tz] = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Quay quanh trục Ox

$$[x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Quay quanh trục Oy

$$[y] = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.1.6 Phép đối xứng

- Qua mặt phẳng tọa độ

$$(yOx) : Mr(x) = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(zOx) : Mr(y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(xOy) : Mr(z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Qua các trục

$$Mx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$My = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Mz = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Qua gốc tọa độ

$$Mo = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

2.2 Quan sát 3 chiều (Phép chiếu - Projection)

2.2.1 Các phép chiếu

Định nghĩa về phép chiếu

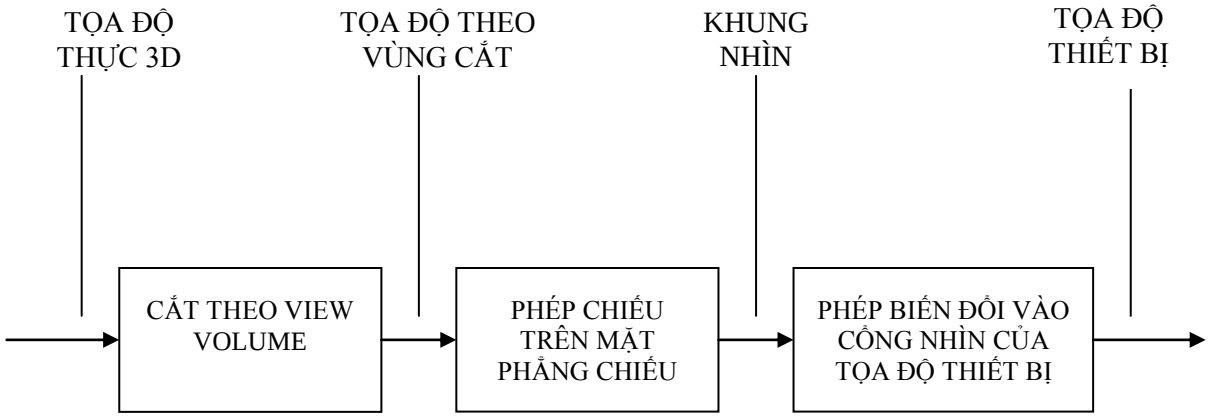
Một cách tổng quát, phép chiếu là phép chuyển đổi những điểm của đối tượng trong hệ thống tọa độ n chiều thành những điểm trong hệ thống tọa độ có số chiều nhỏ hơn n.

Định nghĩa về hình chiếu

Ảnh của đối tượng trên mặt phẳng chiếu được hình thành từ phép chiếu bởi các đường thẳng gọi là tia chiếu (projection) xuất pháp từ một điểm gọi là tâm chiếu (center of projection) đi qua các điểm của đối tượng giao với mặt chiếu (projection plan)

Các bước xây dựng hình chiếu

- Đối tượng trong không gian 3D với tọa độ thực được cắt theo một không gian xác định gọi là view volume.
- View volume dược chiếu lên mặt phẳng chiếu. Diện tích choản bởi view volume trên mặt phẳng chiếu đó sữ cho chúng ta khung nhìn.
- Là việc ánh xạ khung nhìn vào trong một cổng nhìn bất kỳ cho trước trên màn hình để hiển thị hình ảnh.



Hình 2.1 Mô hình nguyên lý của tiến trình biểu diễn đối tượng 3D

2.2.1.1 Phép chiếu song song (Parallel Projections)

Phép chiếu song song (Parallel Projections) là phép chiếu mà ở đó các tia chiếu song song với nhau hay xuất phát từ điểm vô cùng.

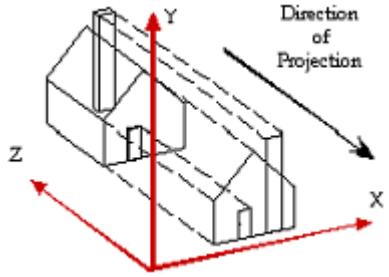
Phân loại phép chiếu song song dựa trên hướng của tia chiếu (Direction Of Projection) và mặt phẳng chiếu (projection plane).

2.2.1.1.1 Phép chiếu trực giao (Orthographic projection)

Là phép chiếu song song và tia chiếu vuông góc với mặt phẳng chiếu. Về mặt toán học, phép chiếu trực giao là phép chiếu với một trong các mặt phẳng tọa độ có giá trị bằng 0. Thường dùng mặt phẳng z=0, ngoài ra x=0 và y=0.

Ứng với mỗi mặt phẳng chiếu ta có một ma trận chiếu tương ứng.

$$\bar{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \bar{x} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \bar{z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Hình 2.2 Phép chiếu trực giao

Thông thường thì người ta không sử dụng cả 6 mặt phẳng để suy diễn ngược hình của một đối tượng mà chỉ sử dụng một trong số chúng như: hình chiếu bằng, đứng, cạnh.

Cả sáu góc nhìn đều có thể thu được từ một mặt phẳng chiếu thông qua các phép biến đổi hình học như quay, dịch chuyển hay lấy đối xứng.

Ví dụ: giả sử chúng ta có hình chiếu bóng trên mặt phẳng z=0, với phép quay đối tượng quanh trục một góc 90 sẽ cho ta hình chiếu cạnh.

Đối với các đối tượng mà các mặt của chúng không song song với một trong các mặt phẳng hệ tọa độ thì phép chiếu này không cho hình dạng thật của vật thể. Muốn nhìn vật thể chính xác hơn người ta phải hình thành phép chiếu thông qua viếc quay và dịch chuyển đối tượng sao cho mặt phẳng đó song song với các trục toạ độ.

Hình của đối tượng quá phức tạp cần thiết phải biết các phần bên trong của đối tượng đôi lúc chúng ta phải tạo mặt cắt đối tượng.

2.2.1.1.2 Phép chiếu trục lượng (Axonometric)

Phép chiếu trục lượng là phép chiếu mà hình chiếu thu được sau khi quay đối tượng sao cho ba mặt của đối tượng được trông thấy rõ nhất (thường mặt phảng chiếu là z=0).

Có 3 phép chiếu

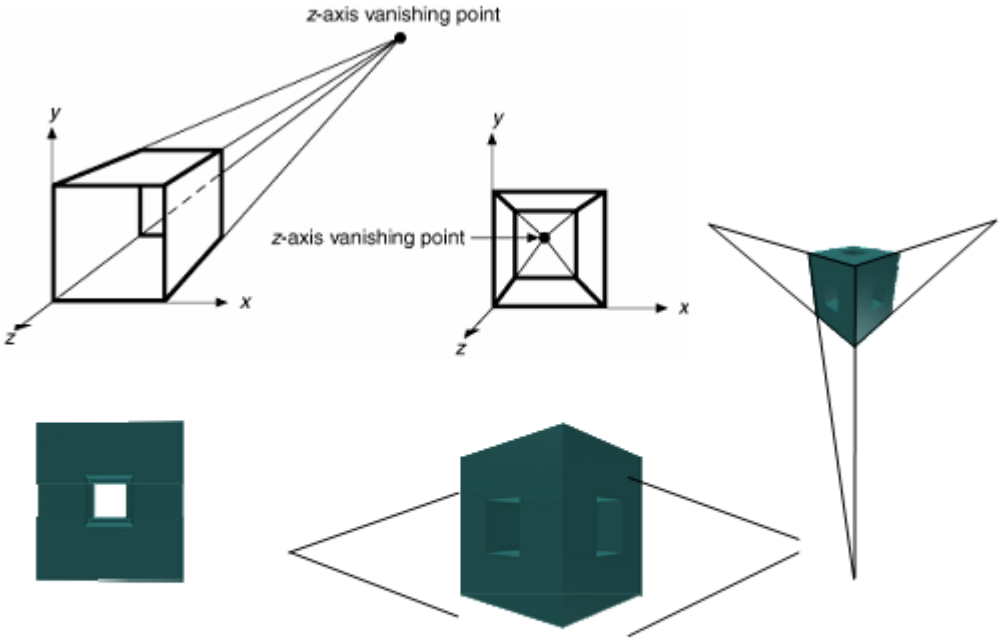
- Phép chiếu Trimetric
- Phép chiếu Dimetic
- Phép chiếu Isometric

2.2.1.2 Phép chiếu phối cảnh (Perspective Projection)

Phép chiếu phối cảnh là phép chiếu mà các tia chiếu không song song với nhau mà xuất phát từ một điểm gọi là tâm chiếu. Phép chiếu phối cảnh tạo ra hiệu ứng về luật xa gần tạo cảm giác về độ sâu của đối tượng trong thế giới thật mà phép chiếu song song không lột tả được.

Các đoạn thẳng song song của mô hình 3D sau phép chiếu hội tụ tại một điểm gọi là điểm triệt tiêu (vanishing point).

Phân loại phép chiếu phối cảnh dựa vào tâm chiếu - *Centre Of Projection* (COP) và mặt phẳng chiếu - projection plane



Một tâm chiếu	Hai tâm chiếu	Ba tâm chiếu
1 point Perspective	2 point Perspective	3 point Perspective

Hình 2.3 Phép biến đổi phối cảnh

2.2.1.2.1 Phép chiếu phối cảnh một tâm chiếu

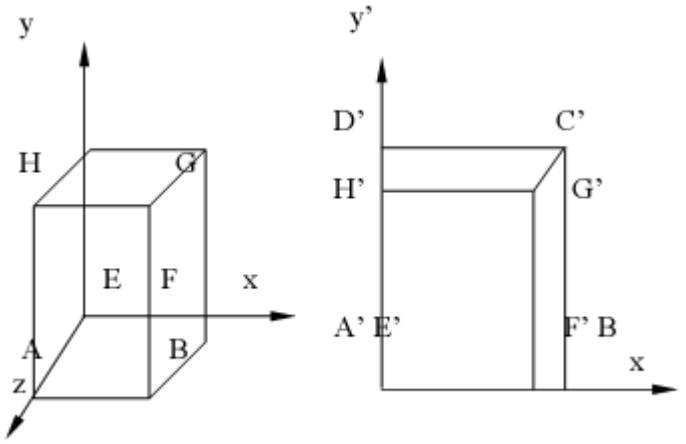
Giả sử khi mặt phẳng được đặt tại z = 0 và tâm phép chiếu nằm trên trục z, cách trục z một khoảng zc = -1/r.

Nếu đối tượng cũng nằm trên mặt phẳng z = 0 thì đối tượng sẽ cho hình ảnh thật.

Phương trình biến đổi:

[x y z 1][Tr] = [x y z rz+1]

Ma trận biến dổi một điểm phối cảnh [Tr] có dạng:



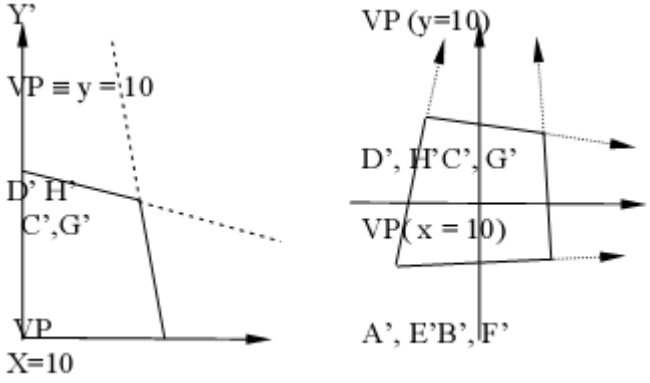
Hình 2.4 Phép chiếu phối cảnh một tâm chiếu

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & rz+1 \end{bmatrix}$$

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{rz+1} & \frac{y}{rz+1} & 0 & 1 \end{bmatrix}$$

2.2.1.2.2 Phép chiếu phối cảnh hai tâm chiếu



Hình 2.5 phép chiếu phối cảnh hai tâm chiếu

$$[C] = [pq].[z]$$

$$= \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[T_{pq}] = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

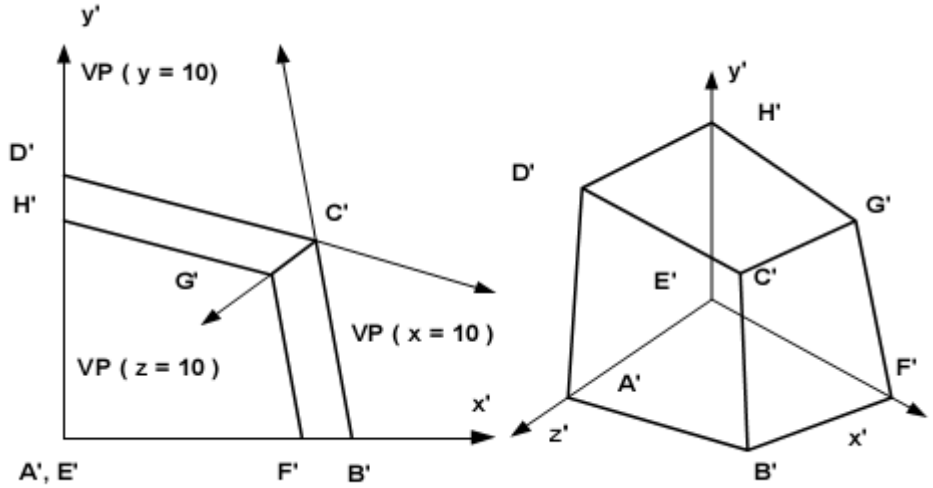
$$[x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x \quad y \quad z \quad (px+qy+1)]$$

$$[x' \quad y' \quad z' \quad 1] = \left[\frac{x}{(px+qy+1)} \quad \frac{y}{(px+qy+1)} \quad \frac{z}{(px+qy+1)} \quad 1 \right]$$

Hai tâm chiếu: [-1/p 0 0 1] và [0 -1/q 0 1]

Điểm triêu tiêu (VP – Vanishing point) tương ứng trên 2 trục x và y là điểm: [1/p 0 0 1] và [0 1/q 0 1].

2.2.1.2.3 Phép chiếu phối cảnh ba tâm chiếu



Hình 2.6 Phép chiếu phối cảnh ba tâm chiếu

$$[pqr] = [p] \cdot [q] \cdot [r]$$

$$= \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[T_c] = [T_{pqr}] \cdot [T_z] = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x \quad y \quad z \quad (px+qy+rz+1)]$$

$$[x' \quad y' \quad z' \quad 1] = \left[\frac{x}{(px+qy+rz+1)} \quad \frac{y}{(px+qy+rz+1)} \quad \frac{z}{(px+qy+rz+1)} \quad 1 \right]$$

Ba tâm chiếu: trên trục x tại điểm [-1/p 0 0 1], y tại điểm [0 -1/q 0 1] và z tại điểm [0 0 -1/r 1]

Điểm triệt tiêu – VP sẽ tương ứng với các giá trị:

[1/p 0 0 1], [0 1/q 0 1] [0 0 1/r 1]

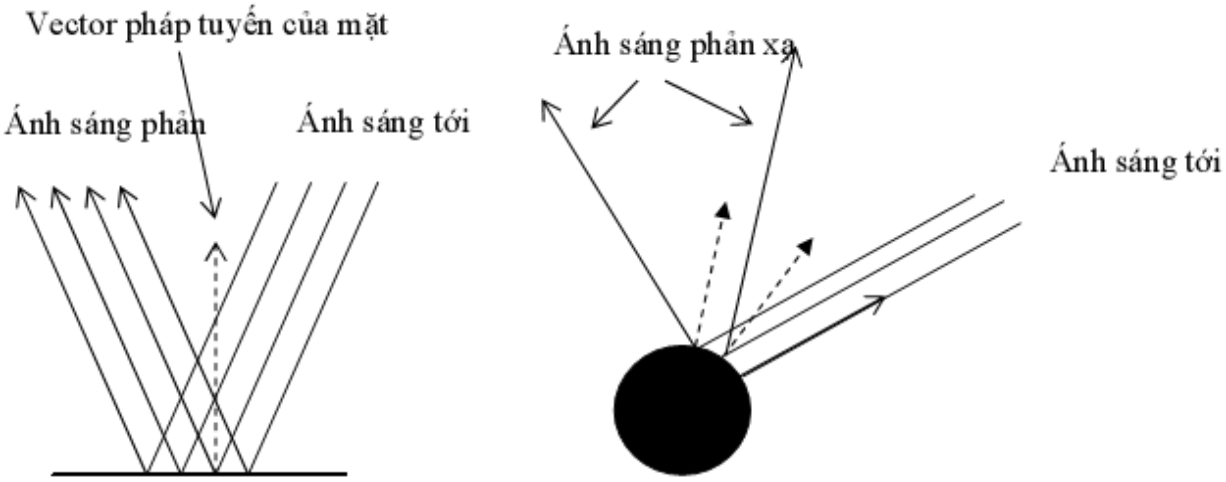
2.2.2 Chiếu sáng và tô bóng

Khi biểu diễn các đối tượng 3 chiều, một yếu tố không thể bỏ qua để tăng tính thực của đối tượng đó là tạo bóng sáng cho vật thể. Để thừjc hiện được điều này, chúng ta cần phải lần lượt tìm hiểu các dạng nguồn sáng có trong tự nhiên, cũng như các tính chất đặc trưng khác nhau của mỗi loại nguồn sáng.

2.2.2.1 Nguồn sáng xung quanh

Ánh sáng xung quanh là mức ánh sáng trung bình, tồn tại trong một vùng khôn gian. Một không gian lý tưởng là khôn gian mà tại đó mọi vật đều được cung cấp một lượng ánh sáng lên bề mặt là như nhau, từ mọi phía ở mọi nơi. Thông thường ánh sáng xung quanh được xác định với một mức cụ thể gọi là

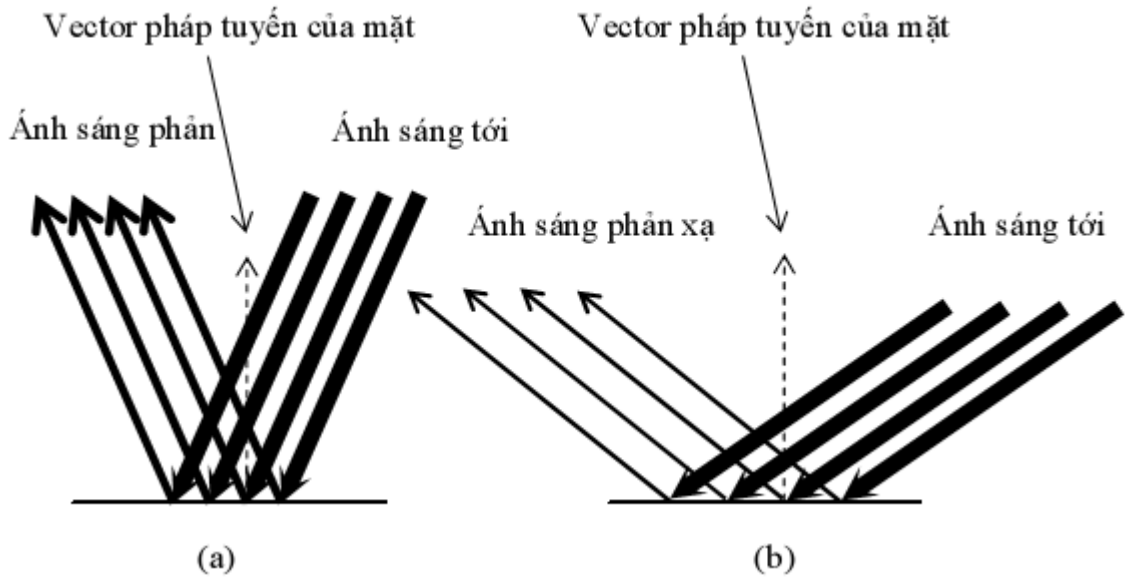
mức sáng xung quanh của vùng không gian mà vật thể đó cư ngụ, sau đó ta cộng với cường độ sáng có được từ các nguồn sáng khác để có được cường độ sáng cuối cùng lên một điểm hay một mặt của vật thể.



Hình 2.7. Sự phản xạ của ánh sáng

2.2.2.2 Nguồn sáng định hướng

Nguồn sáng định hướng giống như những gì mà mặt trời cung cấp cho chúng ta. Nó bao gồm một tập các tia sáng song song, bất kể cường độ của chúng có giống nhau hay không. Có hai loại kết quả của ánh sáng định hướng khi chúng ta chiêu đến bề mặt là: khuếch tán và phản chiếu. Nếu bề mặt phản xạ toàn bộ (giống như trong gương) thì các tia phản xạ sẽ có hướng ngược với hướng của góc tới. Trong trường hợp ngược lại, nếu bề mặt là không phản xạ toàn phần (có độ xám, xù xì) thì một phần các tia sáng sẽ bị tỏa đi các hướng khác hay bị hấp thụ, phần còn lại thì phản xạ lại, và lượng ánh sáng phản xạ lại này tỷ lệ với góc tới. Ở đây chúng ta sẽ quan tâm đến hiện tượng phản xạ không toàn phần vì đây là hiện tượng phổ biến.



Hình 2.8. Sự phản xạ không toàn phần của ánh sáng

Trong hình 2.8 thể hiện sự phản xạ ánh sáng không toàn phần. Độ đậm nét của các tia ánh sáng tới thể thiện cường độ sáng cao, độ mảnh của các tia phản xạ thể hiện cường độ sáng thấp. Nói chung, khi bề mặt là không phản xạ toàn phần thì cường độ của ánh sáng phản xạ luôn bé hơn so với cường độ của ánh sáng tới, và cường độ của tia phản xạ còn tỷ lệ với góc giữa tia tới với vector pháp tuyến của bề mặt, nếu góc này càng nhỏ thì cường độ phản xạ càng cao. Ở đây ta chỉ quan tâm đến thành phần ánh sáng khuếch tán và tạm bỏ qua hiện tượng phản xạ toàn phần.

Nếu gọi θ là góc giữa tia tới với vector pháp tuyến của bề mặt thì Cos(θ) phụ thuộc vào tia tới a và vector pháp tuyến của mặt n theo công thức:

$$Cos(\theta) = \frac{\vec{a}.\vec{n}}{|\vec{a}|.|\vec{n}|} \qquad (*)$$

Trong công thức trên Cos(θ) bằng tích vô hướng của a và n chia cho tích độ lớn của chúng. Nếu ta đã chuẩn hóa độ lớn của các vector a và n về 1 từ trước thì ta có thể tính giá trị trên một cách nhanh chóng như sau:

Cos(θ) = tích vô hướng của $\vec{a}$ và $\vec{n} = a.x * n.x + a.y * n.y + a.z * n.z$

Vì Cos(θ) có giá trị từ +1 đến -1 nên ta có thể suy ra công thức tính cường độ của ánh sáng phản xạ là:

Cường độ AS phản xạ = Cường độ AS định hướng * $[Cos(\theta)+1/2]$ (**)

Trong đó $[Cos(\theta)+1/2]$ có giá trị trong khoảng từ 0 đến 1. Vậy qua công thức (*) và (**) chúng ta có thể tính được cường độ của ánh sáng phản xạ trên bề mặt khi biết được cường độ của ánh sáng định hướng cũng như các vector pháp tuyến của mặt và tia tới.

2.2.2.3 Nguồn sáng điểm

Nguồn sáng định hướng là tương đương với nguồn sáng điểm đặt ở vô tận. Nhưng khi nguồn sáng điểm được mang đến gần đối tượng thì các tia sáng từ nó phát ra không còn song song nữa mà được tỏa ra theo mọi hướng theo dạnh hình cầu. Vì thế, các tia sáng sẽ rơi xuống các điểm trên bề mặt dưới các góc khác nhau. Giả sử vector pháp tuyến của mặt là n = (xn, yn, zn), điểm đang xét có tọa độ là (x_0, y_0, z_0) và nguồn sáng điểm có tọa độ là (plx, ply, plz) thì ánh sáng sẽ rọi đến điểm đang xét theo vector $(x_0 - plx, y_0 - ply, z_0 - plz)$ hay tia tới:

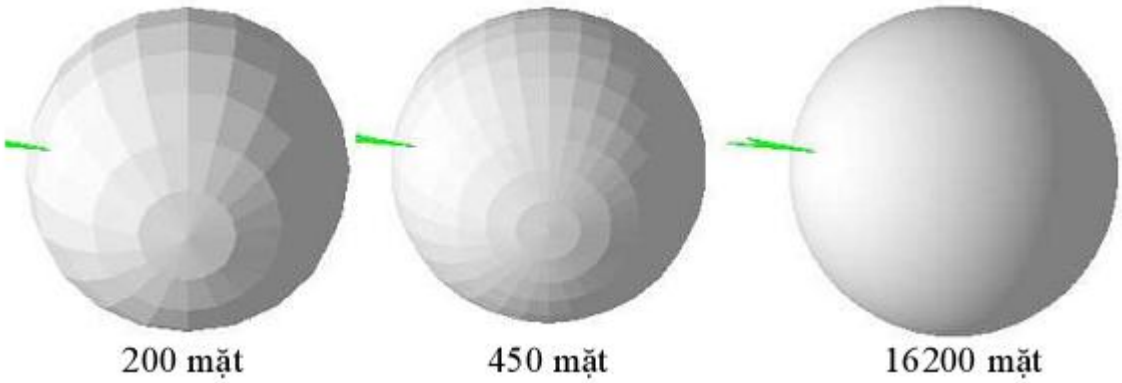
$$a = (plx - x_0, ply - y_0, plz - z_0)$$

Từ đó cường độ sáng tại điểm đang xét sẽ phụ thuộc vào Cos(θ) giữa n và a như đã trình bày trong nguồn sáng định hướng.

2.2.2.4 Mô hình bóng Gouraud

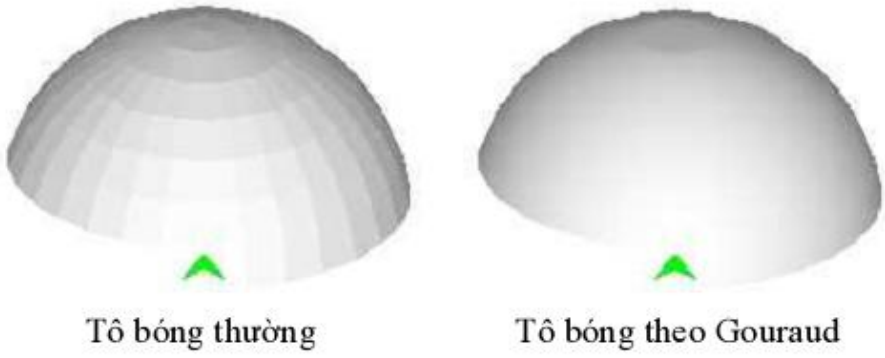
Mô hình bóng Gouraud là một phương pháp vẽ bóng, tạo cho đối tượng 3D có hình dáng cong có một cái nhìn có tình thực hơn. Phương pháp này đặt cơ sở trên thực tế sau: đối với các đối tượng 3D có bề mặt cong thì người ta thường xấp sỉ bề mặt cong của đối tượng bằng nhiều mặt đa giác phẳng, ví dụ như một mặt cầu cso thể xấp sỉ bởi một tập các mặt đa giác phẳng có kích thước nhở sắp xếp lại, khi số đa giác xấp xỉ tăng lên thì tính thực của mặt cầu sẽ tăng, sẽ cho ta

cảm giác mặt cầu trông tròn trịa hơn, mịn và cong hơn. Tuy nhiên, khi số đa giác xấp xỉ một mặt cong tăng thì khối lượng tính toán và lưu trữ cũng tăng theo tỷ lệ thuận theo số mặt, điều đó dẫn đến tốc độ thực hiện sẽ trở nên chậm chạp hơn. Vấn đề thứ 2 nảy sinh là khi ta phóng lớn hay thu nhỏ vật thể. Nếu ta phóng lớn thì rõ ràng các đa giác cũng được phóng lớn theo cùng tỷ lệ, dẫn đến hình ảnh về các mặt đa giác lại hiện rõ và gây ra cảm giác không được trơn mịn. Ngược lại, khi ta thu nhỏ thì nếu số đa giác xấp xỉ lớn thì sẽ dẫn đến tình trạng các đa giác nhỏ, chồng chất lên nhau không cần thiết.



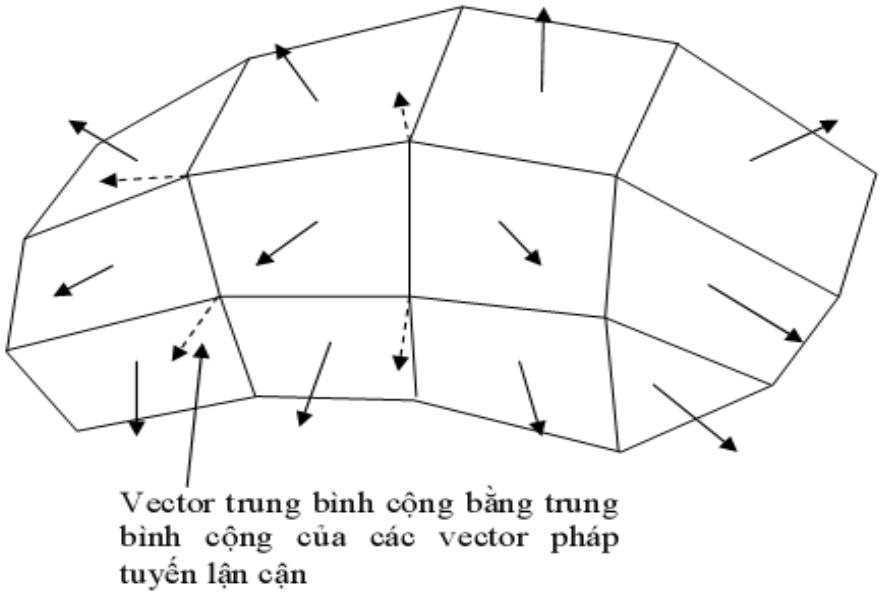
Hình 2.9. So sánh vật thể với số mặt đa giác tăng dần

Để giải quyết vấn đề trên, chúng ta có thể tiến hành theo phương pháp tô bóng Gouraud. Mô hình bóng Gouraud tạo cho đối tượng một cái nhìn giống như là nó có nhiều mặt đa giác bằng cách vẽ mỗi mặt không chỉ với một cường độ sáng mà vẽ với nhiều cường độ sáng khác nhau trên các vùng khác nhau, làm cho mặt phẳng nơm như bị cong. Bởi thực chất ta cảm nhận được độ cong của các mặt cong do hiệu ứng ánh sáng khi chiếu lên mặt, tại các điểm trên mặt cong sẽ có vector pháp tuyến khác nhau nên sẽ đón nhận và phản xạ ánh sáng khác nhau, từ đó ta sẽ cảm nhận được các độ sáng khác nhau trên cùng một mặt cong.



Hình 2.10. So sáng tô bóng thường và tô bóng Gouraud

Thường thì mỗi mặt đa giác có một vector pháp tuyến, và như phần trên đã trình bày, vector pháp tuyến đó được dùng để tính cường độ của ánh sáng phản xạ trên bề mặt của đa giác từ đó suy ra cường độ sáng của mặt. Tuy nhiên mô hình Gouraud lại xem một đa giác không chỉ có một vector pháp tuyến, mà mỗi đỉnh của mặt đa giác lại có một vector pháp tuyến khác nhau, và từ vactor pháp tuyến của các đỉnh chúng ta sẽ nội suy ra được vector pháp tuyến của từng điểm trên mặt đa giác, từ đó tính được cường độ sáng của điểm. Như thế, các điểm trên cùng một mặt của đa giác sẽ có cường độ sáng khác nhau và cho ta cảm giác mặt đa giác không phải là mặt phẳng mà là mặt cong.



Hình 2.11. Mô tả vector trung bình cộng của các mặt

CHƯƠNG 3: Giới thiệu về Engine OGRE

3.1 Giới thiệu tổng quan về OGRE

3.1.1 Lịch sử phát triển

- Gần 1999

Sinbad thực hiện dự án 'DIMClass' của anh ấy, dự án tạo ra một thư viện Direct3D định hướng đối tượng tương đối dễ sử dụng, nó đã trở nên tách biệt đến mức không cần dựa vào Direct3D nào. Và bắt đầu lập kế hoạch một thư viện đầy tham vọng có thể là Hệ giao tiếp lập trình ứng dụng (API) với nền tảng độc lập.

- 25/2/2000

Đăng ký dự án mã nguồn mở, lấy tên là OGRE. Nhưng không bắt đầu phát triển vì một vài cam kết khác.

- 2/2005

OGRE v1.0.0 "Azathoth" được đưa ra – xem xét lại toàn bộ hệ thống tài nguyên, bộ đệm pixel phần cứng, HRD, CEGui, hệ thống xuất XSI.

- 5/2005

OGRE là 'dự án của tháng' trên *SourceForge*.net

- 4/11/2005

Ankh được phát hành như là sản phẩm thương mại đầu tiên sử dụng OGRE

- 7/5/2006

OGRE 1.2 [Dagon] được chính thức phát hành

- 25/3/2007

OGRE 1.4 [Eihort] được chính thức phát hành

- 28/8/2008

  OGRE 1.6 [Shoggoth] được chính thức phát hành

3.1.2 Một số khái niệm và đặc điểm về OGRE

- Engine là một dụng cụ, phương tiện, động cơ.
- Game engine là tập hợp của các bộ thư viện các hàm độc lập cộng với hệ thống công cụ hỗ trợ cho nhiều công đoạn trong việc phát triển game. Thí dụ như ngay trong các sản phẩm này đã có những phần mềm giúp thiết kế mô hình (model editor), dựng màn chơi (level editor), trình viết mã (script editor) và thậm chí là cả hệ thống mô phỏng vật lí.
- Graphics engine là tập hợp của các bộ thư viện các hàm độc lập cộng với hệ thống công cụ hỗ trợ cho nhiều công đoạn trong việc phát triển đồ họa.
- Ogre là một engine kết xuất đồ họa mã nguồn mở (OGRE - *Object-oriented Graphics Rendering Engine*) mà được viết và bảo trì bởi một nhóm cốt lõi nhỏ, và sự đóng góp từ cộng đồng của nó. Người ta đánh giá OGRE là một engine mở miễn phí tốt nhất hiện nay.
- Bạn có thể làm 1 game bằng OGRE. Không như một số engine 3D khác, chỉ định hướng vào một loại game, với OGRE bạn có thể làm cả game "2.5D" hoặc 3D nếu thành thạo. Thậm chí là 1 MMORPG (*Massively Multiplayer Online Role-Playing Game* - một dạng trò chơi mà nhiều người chơi trực tiếp nhập vai và tương tác với nhau trong thế giới ảo), mặc dù đó là trên lý thuyết vì OGRE chỉ chuyên về đồ họa, nếu làm 1 MMORPG bạn cần 1 đội giỏi, thêm các thư viện âm thanh, mạng, các công cụ để phát triển đối tượng …vv. Một game online trên thị trường Việt Nam hiện nay có sử dụng đồ họa từ OGRE là Thiên Long Bát Bộ.
- OGRE không phải là một game engine, mà nó chỉ là một graphics engine, một game engine đầy đủ cần có:

- Âm thanh.
- Mạng.
- Nguồn vào (mô hình đồ họa, các thực thể, thông tin giữa nhiều người chơi…v.v.).
- Va chạm (chuyên xử lý về vật lý trong game).

- Trang chủ của engine OGRE: http://ogre3d.org

3.1.3 Cấu trúc quản lý cảnh trong OGRE

Để quản lý và sử dụng các đối tượng, OGRE dùng SceneManager (quản lý cảnh). SceneManager sẽ quản lý các đối tượng bằng một đồ thị cảnh có dạng hệ đẳng cấp (hình cây) gồm một nút gốc và các nút con. 1 SceneManager trong Ogre chịu trách nhiệm về những điều sau đây:

- Tạo ra và đặt vào các đối tượng có thể di chuyển, các đèn chiếu sáng, và các camera trong cảnh, trong đó có thể truy cập chúng một cách hiệu quả bằng một đồ thị hình cây.
- Tải và tập hợp các hình học.
- Triển khai thực hiện các truy vấn cảnh để cung cấp câu trả lời cho các câu hỏi như là, "Đối tượng nào được chứa trong một hình cầu được đặt đúng tâm tại một điểm đặc biệt ở thế giới không gian?"
- Chọn lọc những đối tượng không nhìn thấy được và đưa những đối tượng nhìn thấy vào trong những hàng đợi hoàn trả để vẽ.
- Tổ chức và lựa chọn (bằng việc tăng khoảng cách) những ánh sáng vô hướng từ hình phối cảnh cảnh có thể trả lại hiện thời.
- Thiết lập và vẽ bất kỳ đổ bóng nào trong cảnh.

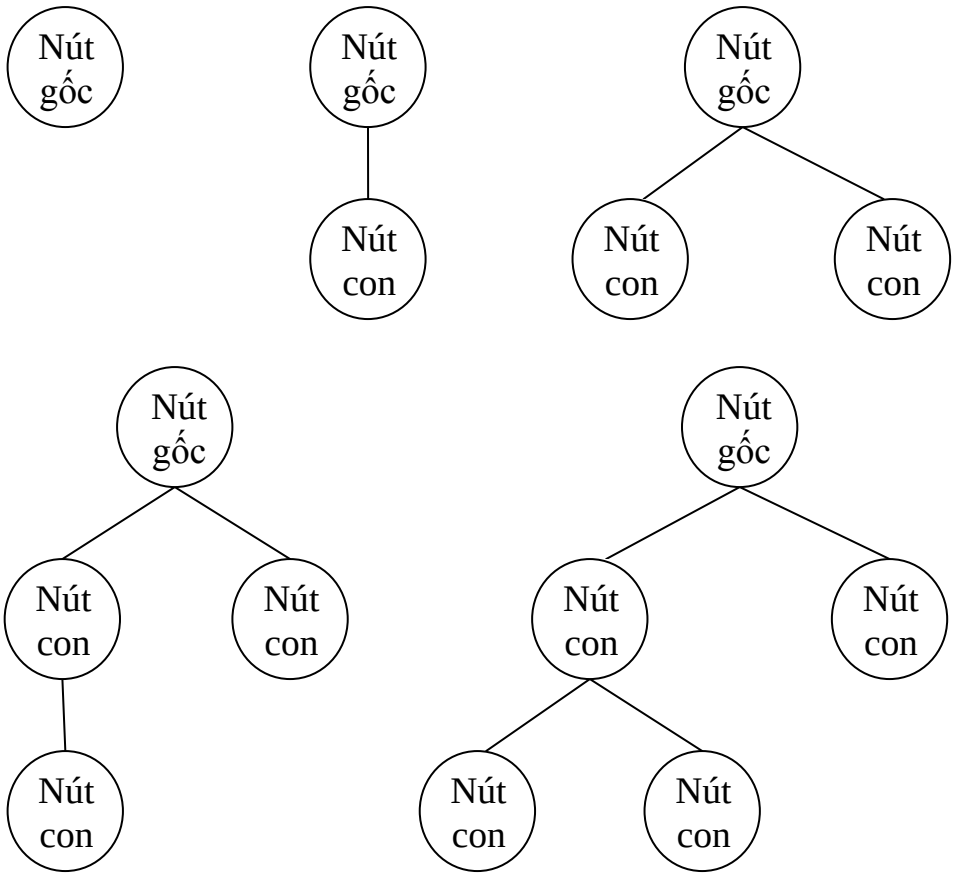
- Thiết lập và vẽ mọi đối tượng khác trong cảnh, như những hình nền và những hộp bầu trời.

Bất cứ thứ gì có thể tồn tại trong một cảnh được quản lý bởi scene manager. Điều này cho phép những chuyên viên thiết kế của SceneManager có thể tùy chỉnh những trạng thái của quá trình tạo đối tượng nếu cần thiết. Lưu ý rằng "quản lý" liên quan đến toàn bộ vòng đời của các đối tượng được tạo ra bởi scene manager, các kiểu quản lý gồm: tạo, nhận, hủy, và "hủy tất cả" các loại. Bất kỳ đối tượng nào thu được từ scene manager phải được huỷ do chính scene manager: nói cách khác, bạn không "xoá" bất kỳ một trong số con trỏ của scene manager trả lại cho ứng dụng của bạn. Nếu bạn muốn giải phóng các đối tượng cảnh (scene) hoặc giũ sạch cảnh của bạn bằng tay bạn cần phải để cho scene manager làm cho bạn.

Những scene manager (quản lý cảnh) cũng chính là nguồn gốc của các nút được sử dụng để định nghĩa cấu trúc của biểu đồ cảnh. Các scene node (nút cảnh) được tổ chức bên trong scene manager trong một hệ đẳng cấp: một nút cảnh chỉ có 1 nút bố mẹ và có thể có không hoặc nhiều nút con.

Scene manager bảo đảm đã tạo ra ít nhất một nút (có thể sử dụng ngay): nút cảnh gốc. Đây là nút duy nhất trong biểu đồ cảnh không có nút bố mẹ: nút gốc, theo định nghĩa, không có cha mẹ. Bạn không thể hủy cảnh nút gốc. Một điểm đặc biệt nữa trong hệ đẳng cấp các nút đó là khi bạn tác động đến 1 nút con sẽ không làm ảnh hưởng đến nút bố mẹ, nhưng nếu bạn tác động đến 1 nút bố mẹ thì các nút con sẽ bị tác động theo.

Sự phân cấp nút cảnh được tạo ra bằng việc thêm các nút con vào những nút hiện đã có trong biểu đồ cảnh. Điều này có nghĩa rằng nút đầu tiên mà bạn thêm vào biểu đồ cảnh sẽ là một nút con của nút cảnh gốc. Hình 3.1 cho thấy sự phân cấp của biểu đồ cảnh sau khi khi thêm vài nút.



Hình 3.1 Trạng thái biểu đồ qua quá trình thêm 6 nút, bắt đầu với 1 nút gốc trong biểu đồ cảnh trống không và tiến triển đến trạng thái cuối cùng

Người quản lý cảnh không đưa nội dung hoặc dữ liệu cho các đối tượng trong cảnh khi bạn tạo chúng, bạn cần phải đính các đối tượng được tạo mới vào một nút cảnh. Nút cảnh này không cần phải đã được gắn vào biểu đồ cảnh thì bạn mới có thể đính kèm vào nội dung các nút; bạn có thể đính kèm nội dung vào bất kỳ nút cảnh trong bất kỳ trạng thái vào bất cứ lúc nào bạn muốn. Bạn cũng có thể đính kèm nhiều hơn một nội dung đối tượng đến một nút cảnh.

Một điều quan trọng cần được hiểu là 3 thao tác không gian chủ yếu (dịch chuyển, quay, co giãn) được thực hiện trên các nút cảnh, không phải trên nội dung các đối tượng. Nói cách khác, cái di chuyển là nút cảnh, không phải nội dung, nội dung chỉ là được cho đi chơi cùng, như bạn sẽ nhìn thấy trong thời gian ngắn.

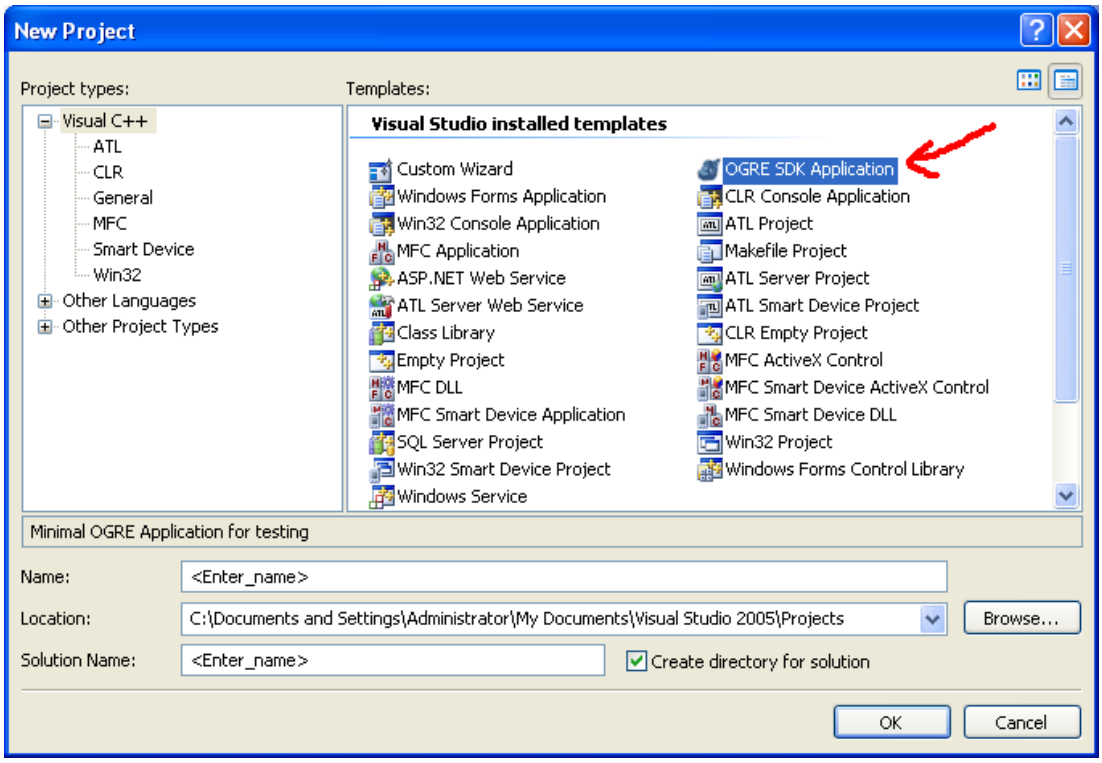
3.2 Cấu hình Engine OGRE

3.2.1 Yêu cầu phần mềm

- Công cụ biên dịch C++: MS's Visual C++ 6/7/7.1/8/9 hoặc mới nhất là C#. Tuy nhiên trong hướng dẫn này tôi sẽ hướng dẫn đầy đủ cách cài đặt và sử dụng OGRE trên MS's Visual C++ 8.0 ở trong bộ Microsoft Visual Studio 2005 Service Pack 1.
- OGRE 1.6.1 SDK for Visual C++ .Net 2005 (8.0) SP1
- Ogre SDK VC8.0 Appwizard để tạo 1 dự án có kiểu OGRE

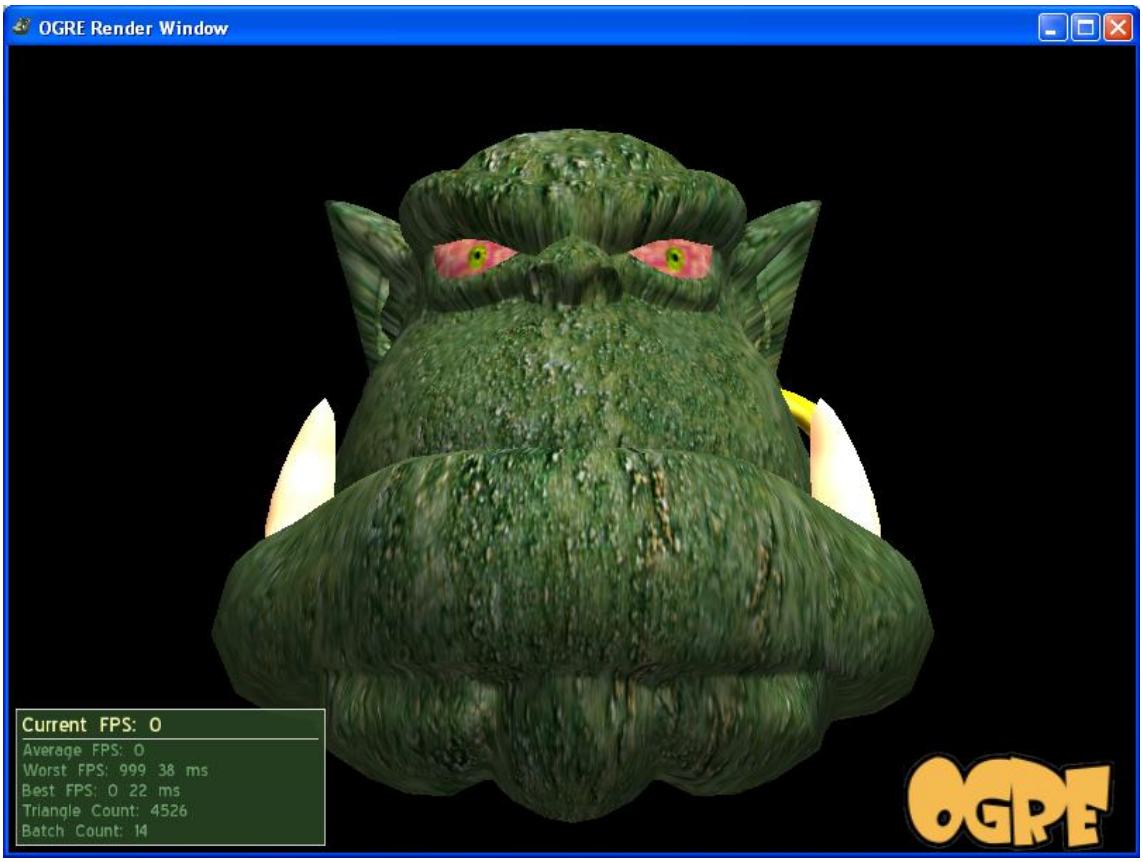
3.2.2 Các bước cài đặt và chạy thử nghiệm

- B1: Cài đặt *Microsoft Visual Studio 2005 Service Pack 1*. Lưu ý, phải đúng là "*Service Pack 1*", nếu bạn chưa update thì sau khi cài xong Visual Studio 2005 bạn cần phải update để lên Pack 1. Tải pack 1 tại đây.
- B2: Cài đặt OGRE 1.6.1 SDK for Visual C++ 8.0 SP1.
- B3: Giải nén file *ogresdkwizard80_Eihort_v1_4_2.zip*, sau đó vào thư mục vừa giải nén, chạy 2 file: *VC8_Express_Setup.js* và *VC8_Setup.js*
- B4: Chạy *Microsoft Visual C++ 8.0*, vào File -> New -> Project. Bạn sẽ thấy một kiểu dự án "OGRE SDK Application". Chọn kiểu này, điền tên dự án, ấn "OK" rồi "Finish".



Hình 3.2 Tạo và đặt tên dự án OGRE

- B5: Tất nhiên là ấn "F5" để bắt đầu build dự án. Một dự án OGRE được khởi tạo có 2 file chính là *.h* và chấm *.cpp*, các câu lệnh để bạn phát triển dự án của mình sẽ được thêm vào file *.h*



Hình 3.3 Chương trình demo ban đầu của 1 dự án OGRE

3.3 Một số bài học và câu lệnh đồ họa 3D

Nhóm phát triển OGRE có đưa ra một số bài học để người dùng có thể học cách sử dụng OGRE một cách nhanh và hiệu quả nhất. Gồm có 8 bài học cơ bản, 7 bài học trung bình, 1 bài nâng cao và một số bài học mở rộng tại http://www.ogre3d.org/wiki/index.php/Ogre_Tutorials. Qua các bài học này, các bạn sẽ được học cách điều khiển camera, ánh sáng, vật thể, chuột, bàn phím …v.v. Ngoài ra khi cài OGRE bạn có thể tham khảo các câu lệnh của OGRE trong "OGRE API Reference".

Một số câu lệnh đồ họa 3D

- **Quay quanh trục:**

virtual void Ogre::Node::rotate(const Vector3& axis, const Radian& angle, TransformSpace relativeTo=TS_LOCAL)

axis : trục cần quay quanh

angle: góc cần quay theo chiều dương

relativeTo: không gian của phép quay, có 3 loại không gian cơ bản là TS_LOCAL, TS_PARENT, TS_WORLD.

  + TS_LOCAL: không gian cục bộ, chính là không gian của nút đó. Quay ở đây là quay quanh chính trục của nút đó.
  + TS_PARENT: không gian gốc, là không gian nút mẹ của nút đó. Quay ở đây là quay quanh trục của nút mẹ.
  + TS_WORLD: không gian toàn cục, không gian chính.

- **Tịnh tiến:**

virtual void Ogre::Node::translate(const Vector3& d, TransformSpace relativeTo=TS_PARENT)

- **Co giãn:**

virtual void Ogre::Node::scale(const Vector3& scale)

Một đoạn mã ví dụ:

```
Entity* ogreHead = mSceneMgr->createEntity("Head", "ogrehead.mesh");
// Tải 1 file .mesh (file vật thể 3D) từ bên ngoài vào và đặt tên là “Head”
SceneNode* headNode = mSceneMgr->getRootSceneNode()->createChildSceneNode();
// Tạo 1 nút cảnh và gắn nó vào nút gốc
headNode->attachObject(ogreHead);
// Gắn vật thể vào nút
headNode->scale( 1, .5 , 1);
// Co vật thể ½ theo trục y
headNode->rotate(Vector3( 1, 0, 0 ), Degree( 90 ));
// Quay vật thể 90^0 theo trục Ox
headNode->translate( Vector3( 80, 0, 0 ));
// Dịch chuyển vật thể 80 đơn vị theo trục Ox
```

CHƯƠNG 4: Thực nghiệm

4.1 Phát biểu bài toán ứng dụng

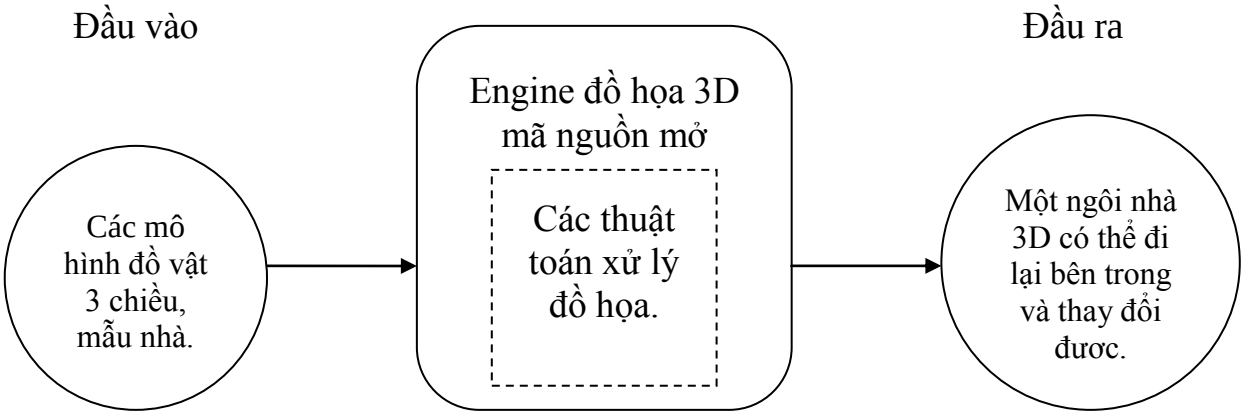
Qua tìm hiểu về lý thuyết một số kỹ thuật đồ họa 3D, vận dụng kiến thức đó để có thể sử dụng thành thạo các kỹ thuật đồ họa 3D của một bộ thư viện đồ họa 3D mã nguồn mở, qua đó viết một ứng dụng 3D về quản lý bất động sản, nhà cửa.

Bài toán đặt ra là khi khách hàng đến một công ty bất động sản để mua nhà, họ sẽ được công ty đưa ra các catalo, ảnh chụp về những mẫu nhà mà công ty đang bán. Và sau đó khách hàng có thể đến tận nơi để xem nhà. Nhưng nếu công ty đang xây dựng một tòa cao ốc, một mẫu nhà mới mà chưa hoàn thành thì khách hàng không thể xem trước cách bố trí nội thất, không gian của ngôi nhà. Và như thế họ sẽ khó mà tìm được một ngôi nhà ưng ý trong tương lai của mình.

Để khắc phục vấn đề này, ta sẽ xây dựng mô hình 3D các mẫu nhà của công ty, kết hợp với các thuật toán xử lý đồ họa đề khách hàng có thể "đi lại" trong ngôi nhà ảo, xem xét kiến trúc và nội thất của ngôi nhà một cách chân thực và sống động nhất. Không chỉ xem xét đơn thuần, các khách hàng còn có thể chọn màu sắc tường nhà, kê lại đồ đạc, thay mẫu sopha, bàn ghế mà họ thích…v.v. Sau khi khách hàng đã vừa ý và ký kết hợp đồng với công ty, mẫu nhà và nội thất do họ bố trí sẽ được lưu vào 1 file riêng hoặc lưu vào cơ sở dữ liệu của công ty. Đội ngũ thiết kế và thi công sẽ nhận được file này hoặc mở dữ liệu ra từ cơ sở dữ liệu bằng phần mềm trên, sau đó họ có trách nhiệm bố trí nhà theo như dữ liệu.

Như vậy khách hàng đã có thể tìm được ngôi nhà yêu thích của mình trong tương lai, và tất nhiên lợi nhuận của công ty sẽ tăng cao. Hơn nữa, công ty có thể liên kết để quảng bá sản phẩm cho các hãng bán sản phẩm nội thất . Vì khi khách hàng kích vào một đồ dùng trong ngôi nhà ảo, sẽ có 1 bảng thông tin hiện ra đầy

đủ tên sản phẩm, giá cả, nhà sản xuất…v.v. để khách hàng sẽ có sự lựa chọn tốt nhất cho ngôi nhà tương lai của mình.



Hình 4.1. Sơ đồ bài toán

4.2 Một số vấn đề chính và hướng giải quyết

4.2.1 Tạo các file đối tượng đồ họa

- Vấn đề

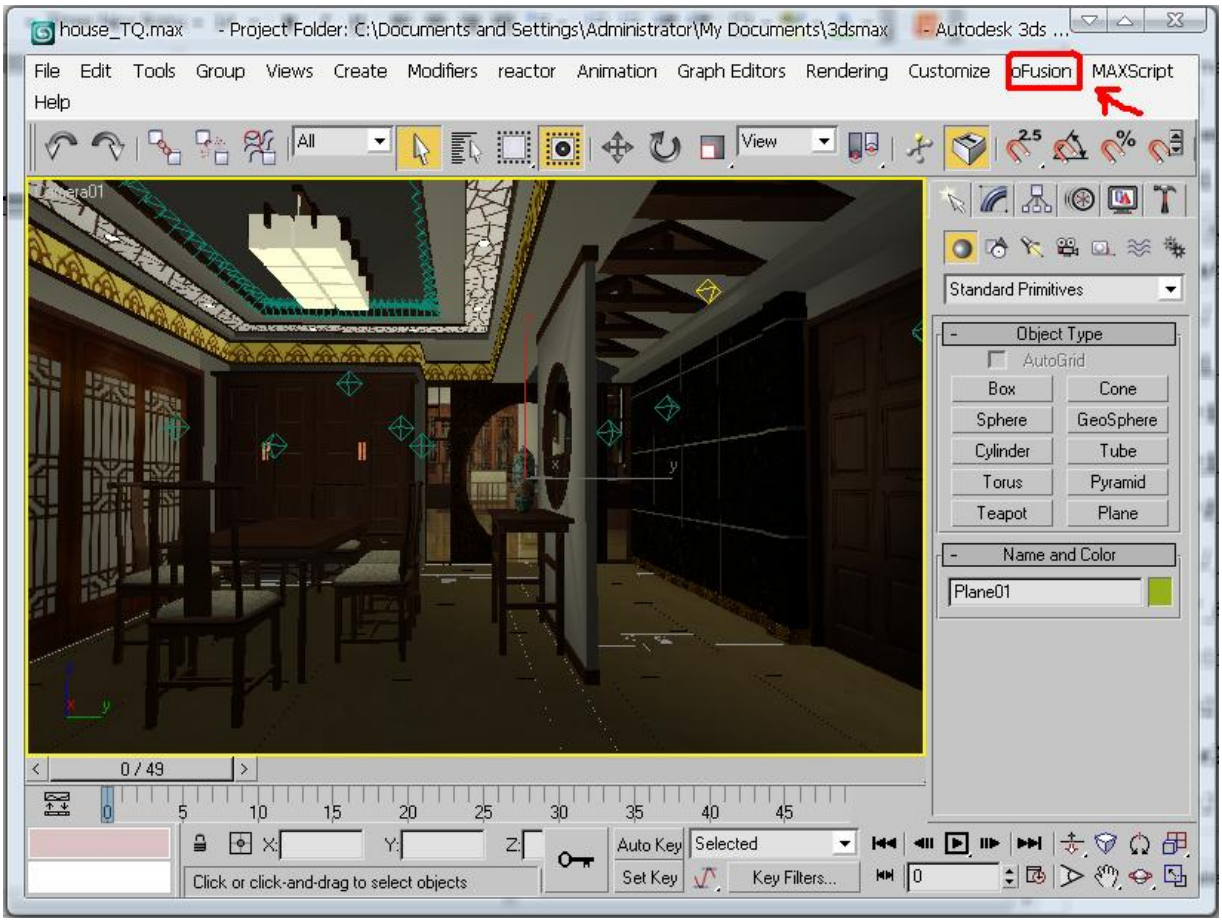
Trong mỗi một ngôi nhà luôn có rất nhiều đồ đạc khác nhau, ví dụ như là giường, tủ, bàn ghế, ti vi… Mỗi vật có màu sắc, thiết kế khác nhau và được tạo nên bởi rất nhiều các mặt phẳng. Nên nếu vẽ các đồ đạc này là rất khó và tốn nhiều thời gian trong OGRE.

- Hướng giải quyết

Ta sẽ dùng 1 phần mềm trung gian mà có thể vẽ các đối tượng đồ họa một cách nhanh và hiệu quả. Sau đó ta sẽ xuất ra file định dạng của OGRE để tải đối tượng đã được vẽ vào trong chương trình. Như thế sẽ làm chương trình ngắn và đơn giản hơn rất nhiều.

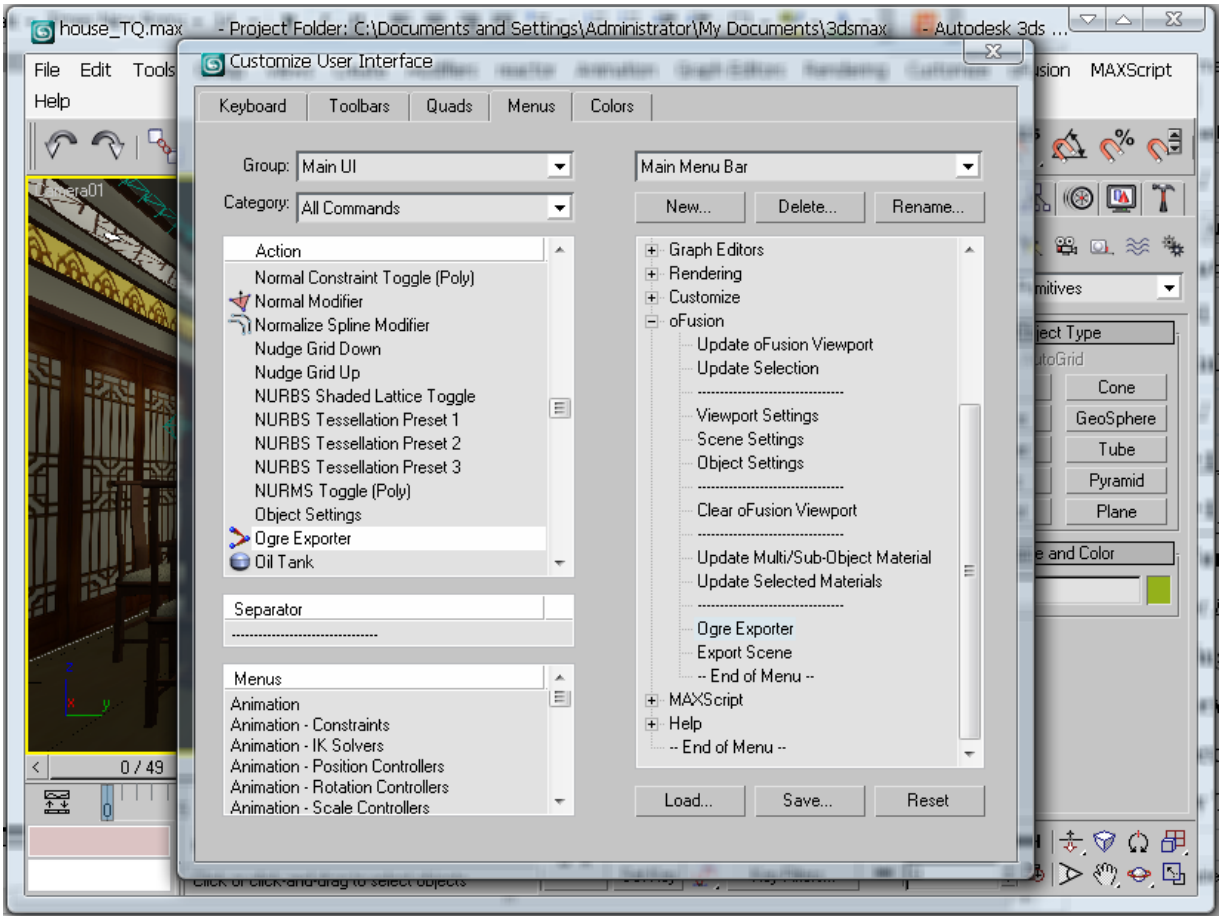
Phần mềm trung gian chuyên về đồ họa sẽ được sử dụng ở đây là 3DS MAX 2008. Đây là một phần mềm chuyên về thiết kế đồ họa 3D được sử dụng khá phổ biến hiện nay. Nhưng trong 3DS MAX 2008 lại chỉ cung cấp 1 số định

dạng file nhất định trong đó không có định dạng "*.mesh*", mà đây lại là định dạng được dùng để tải đối tượng đồ họa vào trong OGRE. Vì thế nên ta cần phải cài thêm 1 số plugin cho 3DS MAX 2008. Cụ thể ở đây là 2 plugin: *Ogre3DSExporter-1.2.2* và *oFusion_ce_1.86*.



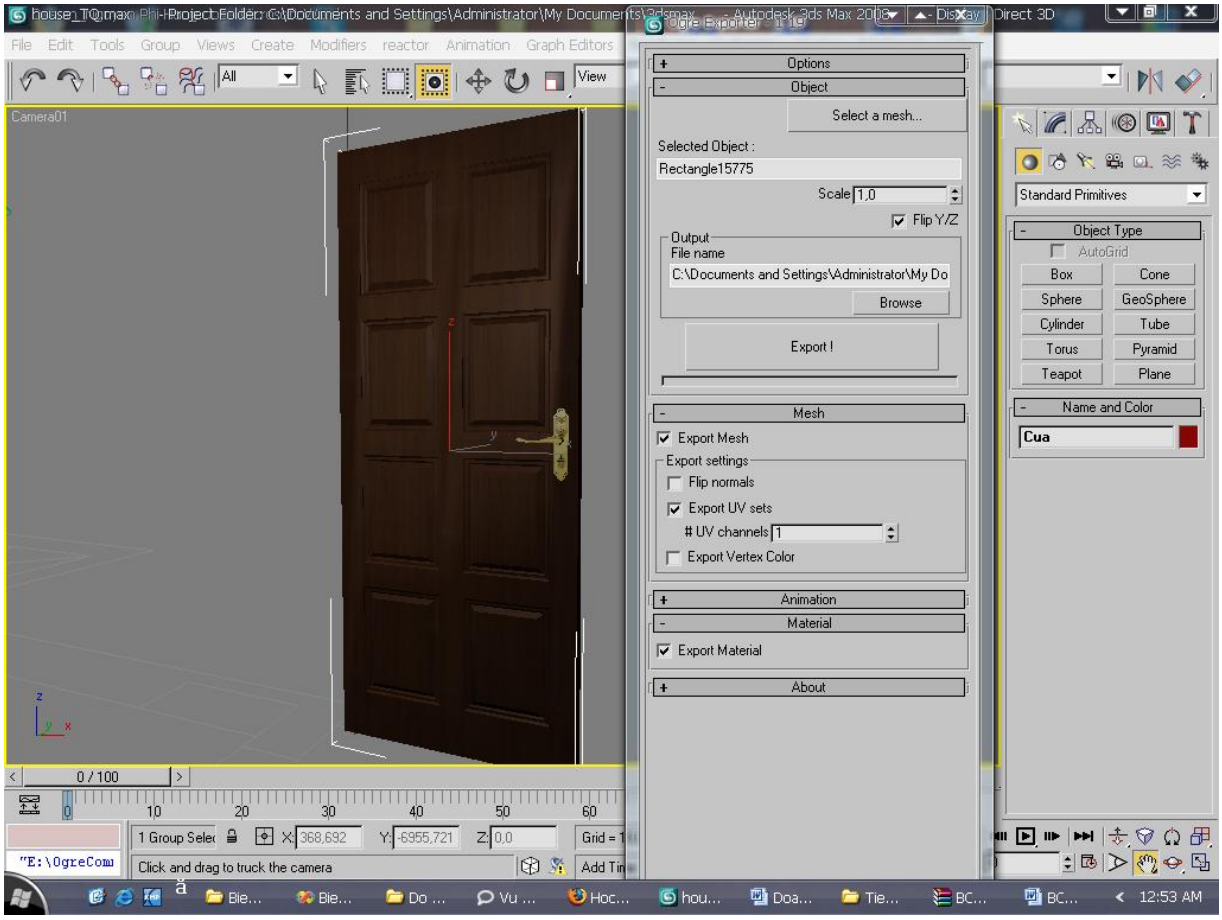
Hình 4.2. Plugin *oFusion_ce_1.86* sau khi được cài thêm.

Để hiện thêm plugin *Ogre3DSExporter-1.2.2* bạn cần phải vào menu *Customize -> Customize User Interface*. Sau đó sẽ hiện ra bảng *Customize User Interface*, chuyển sang tab *Menus*. Ở ô *Action*, chọn *Ogre Exporter* sau đó "kéo" nó qua menu *oFusion* như hình bên dưới.



Hình 4.3. Kéo Plugin *Ogre3DSExporter-1.2.2* ra ngoài.

Với phần *Export Scene* của plugin oFusion, bạn có thể Export ra 1 file *.osm* chứa tên các đối tượng đang được tải vào 3DS MAX, 1 file *.material* chứa các loại chất liệu được phủ lên đối tượng và các file *.mesh* chính là các đối tượng đồ họa. Sau đó ta có thể gọi file .osm để tải toàn bộ các đối tượng đồ họa vào trong OGRE mà không phải tải từng đối tượng 1. Nhưng có 1 nhược điểm với plugin này là các đối tượng đồ họa sau khi được tải vào OGRE ko giữ nguyên được vị trí ban đầu như trong 3DS MAX và việc xuất các hoạt ảnh từ trong 3DS MAX thông qua lugin này cũng không được tốt. Để khắc phục vấn đề này, ta sẽ sử dụng đến plugin Ogre Exporter. Với plugin này ta sẽ phải chọn từng đối tượng cần xuất một, sau đó chọn tên lưu ra file, chọn xuất kèm theo file chất liệu (.material) , nếu có kèm theo hoạt ảnh thì phải chọn thời gian hoạt ảnh diễn ra, tên hoạt ảnh. Lưu ý là chỉ có thể xuất các hoạt ảnh được tạo từ các Bone.



Hình 4.4. Giao diện plugin Ogre Exporter.

Mã nguồn minh họa:

```
// Khởi tạo nút chứa các đối tượng sẽ được tải vào từ file .osm
SceneNode* headNode = mgr->getRootSceneNode()->createChildSceneNode("Ngoai
canh");
OSMScene oScene( mgr, win);
// Tải file .osm
oScene.initialise( "view.osm");
// Thực thi tải các đối tượng trong file vào nút đã tạo
oScene.createScene(headNode);
// Trả lại các đối tượng cho quản lý cảnh
mgr = oScene.getSceneManager();
mgr->getSceneNode("Ngoai canh")->scale( 0.18, 0.18, 0.18);
mgr->getSceneNode("Ngoai canh")->translate( 3500, 450, 1750);
mgr->getSceneNode("Ngoai canh")->rotate(Vector3( 0, 1, 0), Degree(-90));
```

4.2.2 Chọn đối tượng bằng chuột (Select)

- Vấn đề:

Khi người dùng đi lại trong nhà, họ sẽ được nhìn thấy nội thất bên trong của ngôi nhà. Các đồ nội thất này được cung cấp bởi các nhà sản xuất khác nhau đi kèm với rất nhiều thông tin như là: giá cả, thời hạn bảo hành, nơi sản xuất... Để biết được tất cả những thông tin đi kèm đó, tất nhiên người dùng sẽ phải chỉ chuột vào đồ cần biết thông tin và kích chuột. Sau đó phải có 1 bảng thông tin hiện ra để người dùng có thể xem.

- Hướng giải quyết

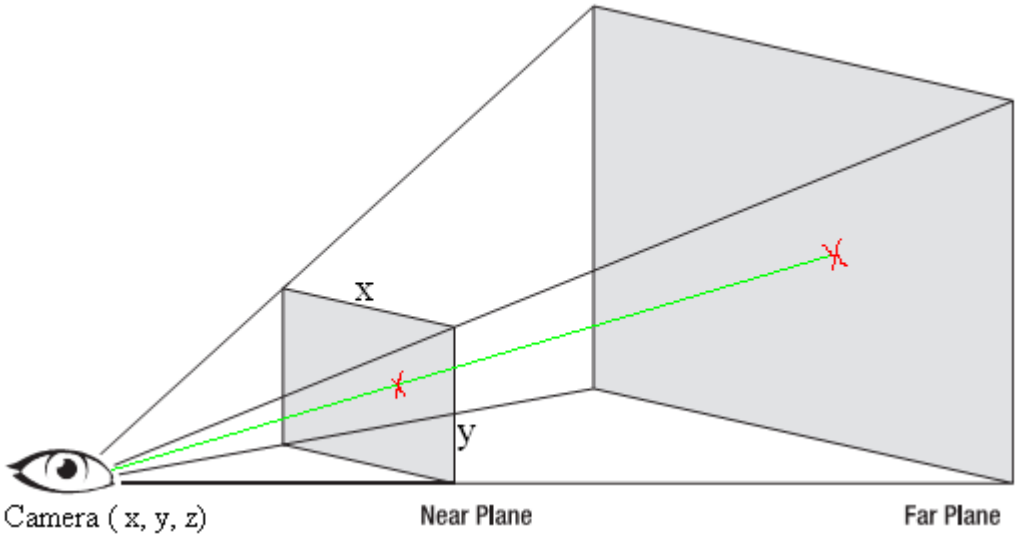
Ta sẽ dựng 1 tia dựa vào vị trí camera và con trỏ chuột. Tia này chạy dọc theo hướng mà ta đã kích chuột. Nó sẽ đi xuyên qua các vật nằm trên hướng đi của nó, và lấy tên của tất cả các vật này cho vào 1 danh sách. Sau đó dựa vào tên của vật được chọn, ta đưa ra các thông tin kèm theo

Để dựng được 1 tia trong không gian 3 chiều ta cần biết 2 điểm. Điểm thứ nhất ta có thể dễ dàng có được, đó chính là tọa độ của camera:

```
mCamera->getPosition();
```

mCamera là biến camera hiện thời, hàm getPosition() sẽ trả ra 1 giá trị có kiểu Vector3(Real x, Real y, Real z).

Điểm thứ 2 ta sẽ dựa vào vị trí đã kích chuột. Màn hình mà chúng ta quan sát là màn hình 2 chiều, vì vậy vị trí mà ta kích chuột cũng có tọa độ 2 chiều. Nhưng tọa độ 2 chiều này nằm trên một mặt phẳng quan sát trong không gian 3 chiều. Nên dựa vào vị trí camera và vị trí kích chuột ta có thể dựng được 1 tia.



Hình 4.5. Minh họa camera và mặt phẳng quan sát

Mã nguồn minh họa:

```
// Lấy vị trí của con trỏ chuột
CEGUI::Point mousePos = CEGUI::MouseCursor::getSingleton().getPosition();
// Tạo 1 biến có kiểu Ray (tia) dựa vào hàm getCameraToViewportRay với 2
tham số đầu vào là tọa độ của con trỏ chuột tính theo chiều dài và rộng
của khung nhìn
Ray mouseRay = mCamera->getCameraToViewportRay(
      mousePos.d_x/(mWindow->getWidth()),
      mousePos.d_y/(mWindow->getHeight()));
// Thiết lập tia
mRaySceneQuery->setRay(mouseRay);
// Thực hiện việc bắn tia xuyên qua các vật thể và trả ra kết quả
RaySceneQueryResult &result = mRaySceneQuery->execute();
// Tạo 1 danh sách
RaySceneQueryResult::iterator itr;
// Sử dụng 1 vòng for để duyệt qua các đối tượng bị bắn
for (itr = result.begin(); itr != result.end(); itr++)
```

4.2.3 Di chuyển đối tượng

- Vấn đề

Khi người dùng đi lại trong nhà, họ quan sát được các loại đồ nội thất và cách bố trí của chúng. Nếu không thích cách bố trí đồ đạc như vậy, họ sẽ lựa chọn sau đó di chuyển đồ cần bố trí lại.

- Hướng giải quyết

Vấn đề lựa chọn đối tượng đã được nêu và giải quyết ở trên. Sau khi lựa chọn đối tượng, ta cần lấy được tên của đối tượng (tên của nút mà đối tượng được gắn vào). Một lưu ý nhỏ là kết quả trả ra sau khi lựa chọn đối tượng được chia làm 2 loại chính, thứ nhất là các đối tượng có thể di chuyển được, đó chính là các vật thể mà ta load vào trong chương trình từ các file .mesh bên ngoài; thứ 2 là các đối tượng không thể di chuyển được, đó chính là địa hình. Nên trước khi lấy tên đối tượng ta cần kiểm tra nó có phải là loại di chuyển được hay không.

```
// Kiểm tra xem có phải là đối tượng di chuyển được hay không
if (itr->movable)
// Lấy nút bị bắn trả về cho đối tượng hiện tại để sử lý
    mCurrentObject = itr->movable->getParentSceneNode();
```

Sau khi lấy được tên nút gắn đối tượng, ta chỉ việc thay đổi tọa độ của nút theo ý muốn.Tọa độ mới của nút ta sẽ tính mỗi khi di chuyển con trỏ chuột, ta lại bắn 1 tia dựa theo tọa độ mới của con trỏ chuột. Ta sẽ lọc kết quả trả về là loại không thể di chuyển được (địa hình), và nơi tia bắn cắt vào địa hình sẽ là tọa độ mới của vật cần di chuyển.

Mã nguồn minh họa

```
// Trong hàm bắt sự kiện di chuyển con trỏ chuột
// Nếu chuột trái đang được ấn và đối tượng đã được chọn
if (mLMouseDown  && mCurrentObject)
{
CEGUI::Point mousePos = CEGUI::MouseCursor::getSingleton().getPosition();
Ray mouseRay = mCamera->getCameraToViewportRay(
      mousePos.d_x/(mWindow->getWidth()),
      mousePos.d_y/(mWindow->getHeight()));
mRaySceneQuery->setRay(mouseRay);
mRaySceneQuery->setSortByDistance(false);

RaySceneQueryResult &result = mRaySceneQuery->execute();
RaySceneQueryResult::iterator itr;

for (itr = result.begin(); itr != result.end(); itr++)
// Kiểm tra xem vật bị bắn có phải là địa hình không
  if (itr->worldFragment)
  {
// Thiết lập vị trí mới cho vật đã được lựa chọn
      mCurrentObject->setPosition(itr->worldFragment->singleIntersection);
      break;
  } // if
} // if
```

4.2.4 Lực chọn mở cửa và tắt bật ánh sáng

- Vấn đề

Khi giới thiệu cho khách hàng xem nhà, tất nhiên là ta phải đi từ cửa chính vào, và tất nhiên ta cần mở cửa ra để đi vào trong nhà. Trong mỗi gian phòng của ngôi nhà có bố trí rất nhiều đèn và ánh sáng, hầu hết trong số chúng đều bị tắt và lúc này người dùng cần phải bật đèn lên để có thể nhìn được hết vẻ đẹp của ngôi nhà hoặc tắt một vài cái đèn nào đó.

Một vấn đề nữa là việc bố trí nhiều đồ đạc và ánh sáng trong một ngôi nhà sẽ làm chậm chương trình.

- Hướng giải quyết

Để có thể lựa chọn và mở cửa, đầu tiên là ta phải tạo chuyển động mở cửa cho đối tượng đồ họa cửa trong 3DS MAX sau đó xuất ra file *.skeleton* sẽ được gọi tự động khi ta tải file *.mesh* tương ứng. Mỗi cái cửa có thể chia ra làm nhiều đối tượng khác nhau, ta phải gắn tất cả các đối tượng này vào cùng 1 nút và đặt

tên nút là cua1, cua2, cua3… Ta sẽ lưu tên các đối tượng thuộc cửa vào 1 file *.cfg* và sau đó tải file vào để lấy tên các đối tượng rồi tiến hành tải tất cả các đối tượng đồ họa thuộc cửa thông qua 1 vòng for. Như vậy sẽ tránh được việc gõ tên cố định của từng đối tượng trong chương trình và có thể thay tên đối tượng mà không phải làm lại chương trình.

Sau đó ta tạo nút mở cửa, nó sẽ hiện ra khi người dùng nhấn vào cửa và sẽ biến mất sau khi được nhấn vì lúc này cửa đã mở. Để biết là người dùng đang nhấn vào 1 cánh cửa. Trong hàm chọn đối tượng ta sẽ tiến hành kiểm tra tên của đối tượng được chọn, nếu 3 ký tự đầu là "cua" thì lúc đó ta sẽ gọi hoạt ảnh mở cửa đã được tạo từ trước.

Mã nguồn minh họa

+ Tạo 1 nút "Cua" từ 1 file cua1.cfg

```
// Create Door
SceneNode* mDoorNode = mgr->getRootSceneNode()->
      createChildSceneNode("Cua");

// Load cfg file
ConfigFile cfg;
cfg.load( "cua1.cfg" );
availableMeshes = cfg.getMultiSetting("Mesh");

Entity* mEntity;

// Load doi tuong
for(int i=0; i < availableMeshes.size(); i++)
{
  mEntity = mgr->createEntity(availableMeshes[i],
      availableMeshes[i] + ".mesh");
  mDoorNode->attachObject(mEntity);
}
```

+ Tạo nút "Mo cua" khi người dùng nhấn vào cửa thì sẽ hiện ra

```
// Open button
CEGUI::WindowManager *winm = CEGUI::WindowManager::getSingletonPtr();
// Tạo 1 trang (sheet) để hiện thị các GUI
CEGUI::Window *sheet = winm->createWindow("DefaultGUISheet",
      "CEGUIDemo/Sheet");
// Tạo nút Open
CEGUI::Window *quit = winm->createWindow("TaharezLook/Button",
      "Open");
open->setText("Mo cua");
open->setSize(CEGUI::UVector2(CEGUI::UDim(0.15, 0),CEGUI::UDim(0.05, 0)));
open->setPosition(CEGUI::UVector2(CEGUI::UDim(0.5,0),CEGUI::UDim(0.5,0)));
open->setVisible(false);

// Thêm nút Open lên trang sheet

sheet->addChildWindow(quit);

mSystem->setGUISheet(sheet);
```

Hình 4.6. Lựa chọn mở cửa

Mỗi một căn nhà thì gồm có nhiều phòng khác nhau. Nếu ta tải đầy đủ tất cả các phòng cùng 1 lúc thì chương trình sẽ chạy rất chậm. Để cải thiện tốc độ chương trình, khi người dùng đang xem 1 phòng nào đó thì tất cả các phòng còn lại sẽ bị ”vô hiệu”. Để làm vô hiệu một căn phòng, ta sẽ gắn tất cả các vật có trong phòng vào 1 nút chung và sau đó chỉ cần giọi hàm removeChild(nodeName) từ nút gốc (nút được khởi tạo mặc định, tất cả các nút được tạo sau đó đều phải gắn vào nút gốc mới có thể “thấy” được). Hàm này sẽ không xóa nút đi mà chỉ tạm thời ngắt nó ra khỏi đồ thị cảnh. Khi người dùng xem xong 1 căn phòng và muốn xang phòng khác, họ sẽ lựa chọn mở cửa. Lúc này ta tiến hành ngắt phòng cũ và gắn phòng mới vào đồ thị cảnh bằng cách gọi hàm addChild(nodeName) từ nút gốc.

Và trong mỗi căn phòng có một hoặc nhiều ánh sáng khác nhau. Với mỗi phòng ta sẽ tạo 1 file riêng để lưu lại các ánh sáng có trong phòng. File này phải lưu được các thông tin cơ bản của ánh sáng gồm: tên, điểm đặt (chỉ áp dụng cho ánh sáng điểm và ánh sáng đèn), cường độ sáng khuếch tán, cường độ sáng phản xạ, hướng chiếu (chỉ áp dụng cho ánh sáng đèn và ánh sáng hướng). Khi người dùng vào một phòng nào đó ta phải gọi file ánh sáng của phòng đó để tạo các ánh sáng có trong phòng. Sau đó ta cần tạo 1 cửa sổ trong chương trình để hiển thị tên của các ánh sáng, với mỗi ánh sáng cần phải tạo thêm lựa chọn tắt – bật ánh sáng bằng nút RadioButton và 1 thanh ScrollBar (thanh kéo) để có thể tăng giảm ánh sáng theo ý muốn. Khi người dùng ấn vào 1 nút trên cửa sổ này, thông qua các hàm bắt sự kiện ta có thể lấy được tên của nút được thay đổi. Với các nút RadioButton ta sẽ gọi hàm setVisible(bool onoff) để tắt hoặc bật ánh sáng. Với các thanh ScrollBar ta sẽ lấy được giá trị mới, sau đó thay đổi màu hiện tại bằng màu cũ cộng thêm với giá trị mới. Nhưng khi thay đổi giá trị màu mới, ta sẽ mất đi giá trị màu cũ nên đến lần kéo sau màu của ánh sáng sẽ bị cộng dồn, và sau vài lần kéo thì màu của ánh sáng đã bị thay đổi không thể trở về giá trị ban đầu.

Thế nên khi tải file ánh sáng của căn phòng, ta phải tiến hành sao lưu lại giá trị màu của các ánh sáng này. Khi đó giá trị màu mới bằng giá trị màu được sao lưu cộng thêm với vị trí của thanh ScrollBar.

Mã nguồn minh họa

+ Tạo 1 nút RadioButton

```
// Tạo nút có kiểu là RadioButton, tên là rOn
CEGUI::Window* radio = winm->createWindow("TaharezLook/RadioButton", "rOn"
                                + StringConverter::toString(i));
// Thay đổi giá trị Text hiển thị cạnh nút
radio->setText((CEGUI::utf8*)"Bá°-t");
((CEGUI::RadioButton*)radio)->setSelected(true);
((CEGUI::RadioButton*)radio)->setMaxSize(
      CEGUI::UVector2(CEGUI::UDim(1, 0), CEGUI::UDim(1, 0)));
// Thiết lập giá trị Left, Top, Width, Height
((CEGUI::RadioButton*)radio)->setArea(
      CEGUI::UDim(0.5, 0), CEGUI::UDim(0.2, 0),
      CEGUI::UDim(0.2, 0), CEGUI::UDim(0.25, 0));
// Chọn nhóm (mỗi nhóm có thể có nhiều nút RadioButton)
((CEGUI::RadioButton*)radio)->setGroupID(i);
// Chọn ID cho nút, các nút trong 1 nhóm phải khác ID
((CEGUI::RadioButton*)radio)->setID(0);
```

+ Tạo 1 thanh ScrollBar

```
// Tạo 1 thanh kéo ngang
CEGUI::Window* horibar = winm->createWindow(
      "TaharezLook/HorizontalScrollbar", "cuongdo" +
      StringConverter::toString(i));
horibar->setMaxSize(
      CEGUI::UVector2(CEGUI::UDim(1, 0), CEGUI::UDim(1, 0)));
horibar->setArea(
      CEGUI::UDim(0.15, 0), CEGUI::UDim(0.75, 0),
      CEGUI::UDim(0.8, 0), CEGUI::UDim(0.2, 0));
horibar->setProperty("PageSize","0");
horibar->setProperty("StepSize","1");
horibar->setProperty("OverlapSize","0");
// Đặt giá trị của toàn bộ thanh kéo
horibar->setProperty("DocumentSize","2");
// Đặt vị trí ban đầu cho thanh kéo
horibar->setProperty("ScrollPosition","1");
horibar->setProperty("InheritsAlpha","False");
```

+ Hàm bắt và xử lý sự kiện thay đổi nút RadioButton

```
bool handleLightOnOff(const CEGUI::EventArgs& e)
{
  // Lấy nhóm của nút RadioButton chọn
  CEGUI::uint gr = ((CEGUI::RadioButton*)((const
      CEGUI::WindowEventArgs&)e).window)->
      getSelectedButtonInGroup()->getGroupID();
  // Lấy ID của nút được chọn
  CEGUI::uint id = ((CEGUI::RadioButton*)((const
      CEGUI::WindowEventArgs&)e).window)->
      getSelectedButtonInGroup()->getID();
  // Bật ánh sáng
  if (id == 0)
      mgr->getLight(stLight[gr])->setVisible(true);
  // Tắt ánh sáng
  if (id == 1)
      mgr->getLight(stLight[gr])->setVisible(false);
  return true;
}
```

+ Hàm bắt và xử lý sự kiện thay đổi ScrollBar

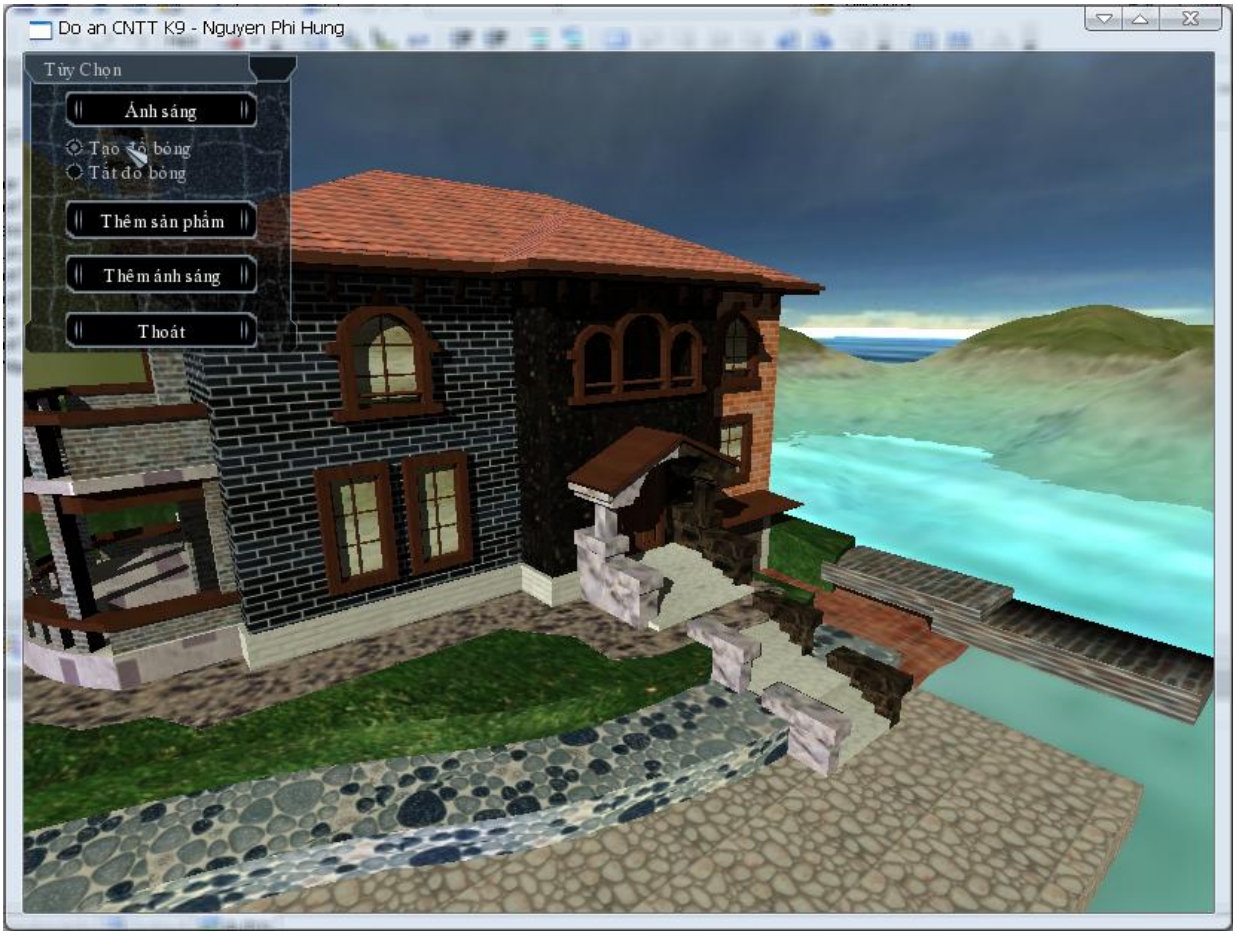
```
bool handleLightChanged(const CEGUI::EventArgs& e)
{
  // Lấy độ dài tên của thanh ScrollBar
  CEGUI::uint l = ((CEGUI::Scrollbar*)((const
      CEGUI::WindowEventArgs&)e).window)->getName().length();
  CEGUI::String numl = ((CEGUI::Scrollbar*)((const
      CEGUI::WindowEventArgs&)e).window)->getName();
  // Đoạn chuyển đổi kiểu dữ liệu từ CEGUI::String -> Ogre::String
  String num;
  char h[20];
  sprintf(h,numl.c_str());
  num = h;
  // Chuyển đổi kiểu dữ liệu từ Ogre::String -> CEGUI::int
  l = StringConverter::parseInt(num.substr(7,l));
  String copyl;
  // Lấy tên của ánh sáng đã được sao chép từ trước
  copyl = "copyLight" + stLight[l];
  // Lấy vị trí mới của thanh Scroll Bar
  float scrollval = ((CEGUI::Scrollbar*)((const
      CEGUI::WindowEventArgs&)e).window)->getScrollPosition() - 1;
  // Thay đổi giá trị màu mới
  mgr->getLight(stLight[l])->setDiffuseColour(
      mgr->getLight(copyl)->getDiffuseColour() +
      ColourValue(scrollval, scrollval, scrollval));
  mgr->getLight(stLight[l])->setSpecularColour(
      mgr->getLight(copyl)->getSpecularColour() +
      ColourValue(scrollval, scrollval, scrollval));
  return true;
}
```

4.3 Giao diện chương trình và một số chức năng chính

4.3.1 Giao diện chương trình

Giao diện chính của chương trình là một cửa sổ 800*600, bên trong là môi trường 3D với góc trên bên trái là cửa sổ con gồm các chức năng chính của chương trình gồm:

- Chỉnh sửa ánh sáng
- Tắt bật đổ bóng
- Thêm sản phẩm
- Thêm ánh sáng
- Xem thông tin sản phẩm, quay, co giãn và bỏ sản phẩm không ưng ý.

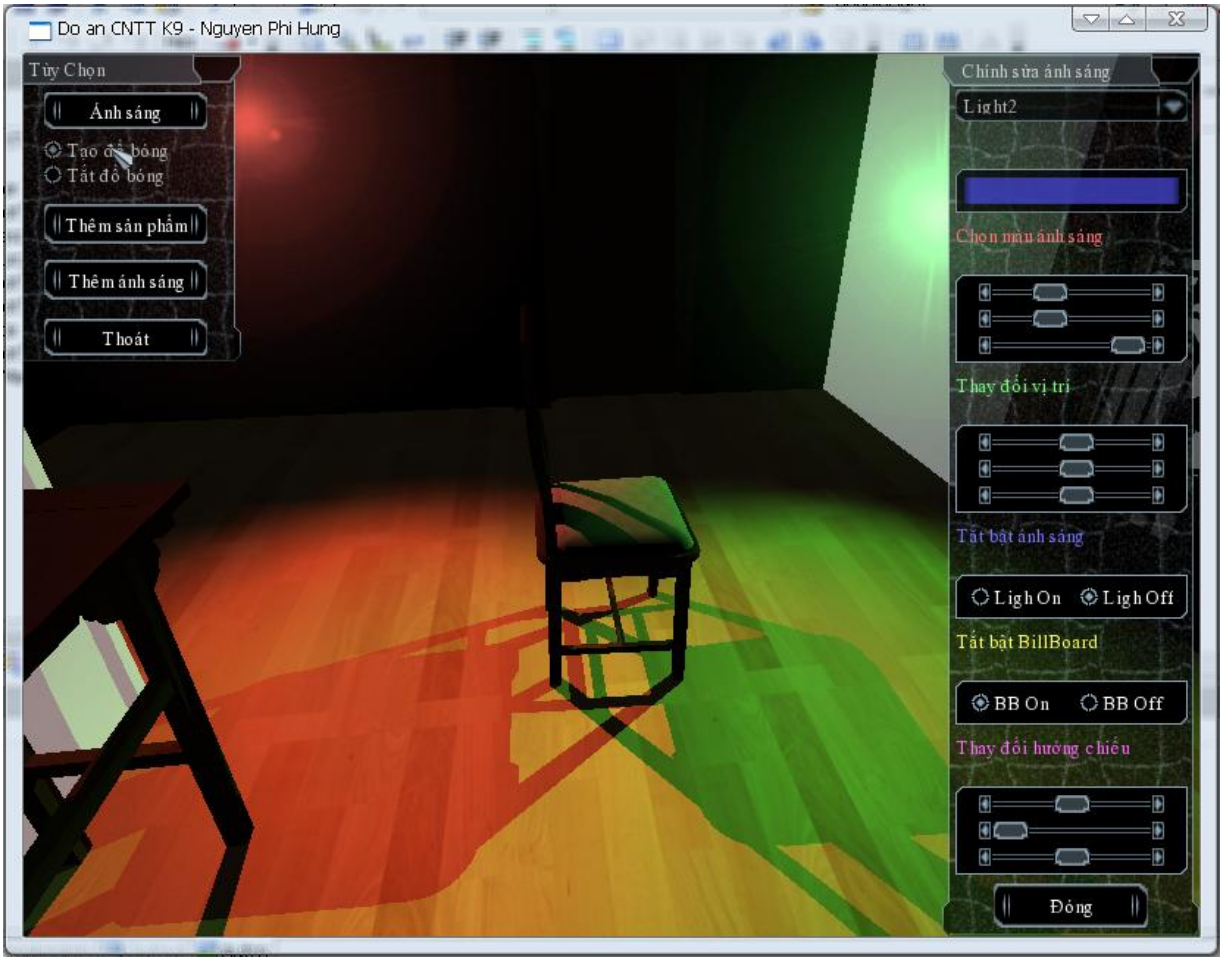


Hình 4.7. Giao diện chính của chương trình

4.3.2 Các chức năng chính

- **Chỉnh sửa ánh sáng**

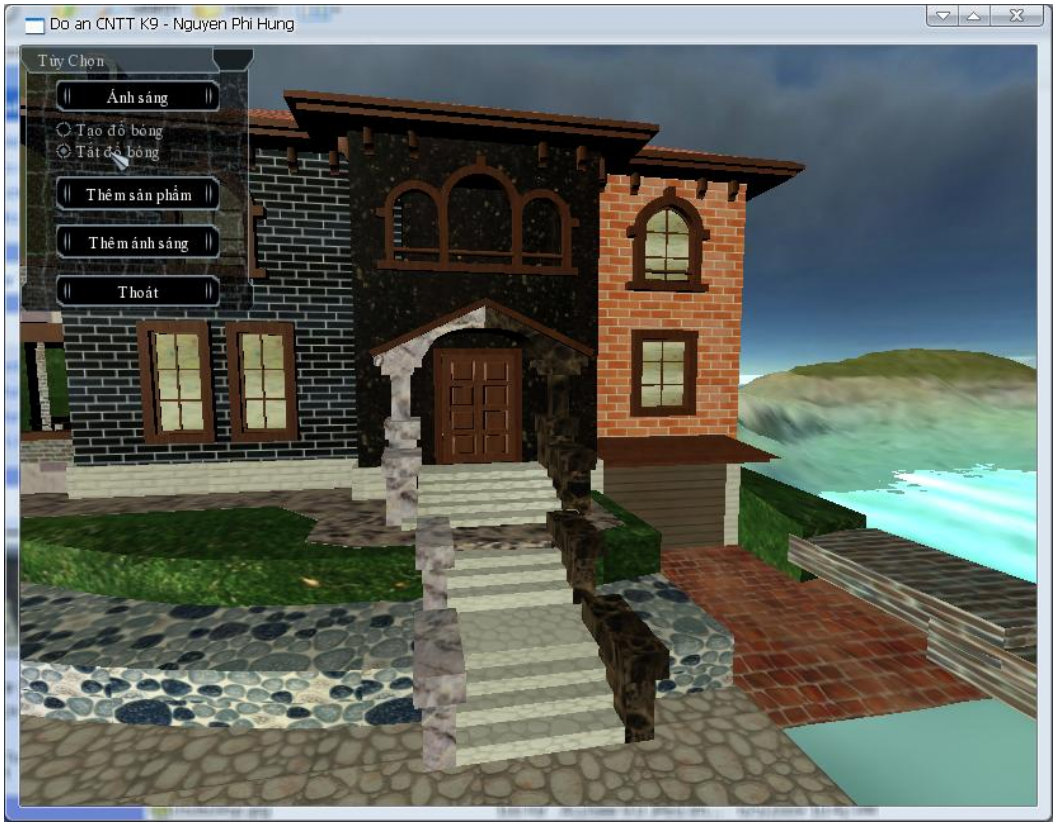
Chức năng này cho phép người dùng xem tất cả các ánh sáng có trong cảnh và có thể tắt bật, thay đổi cường độ sáng của bất kỳ một ánh sáng nào.



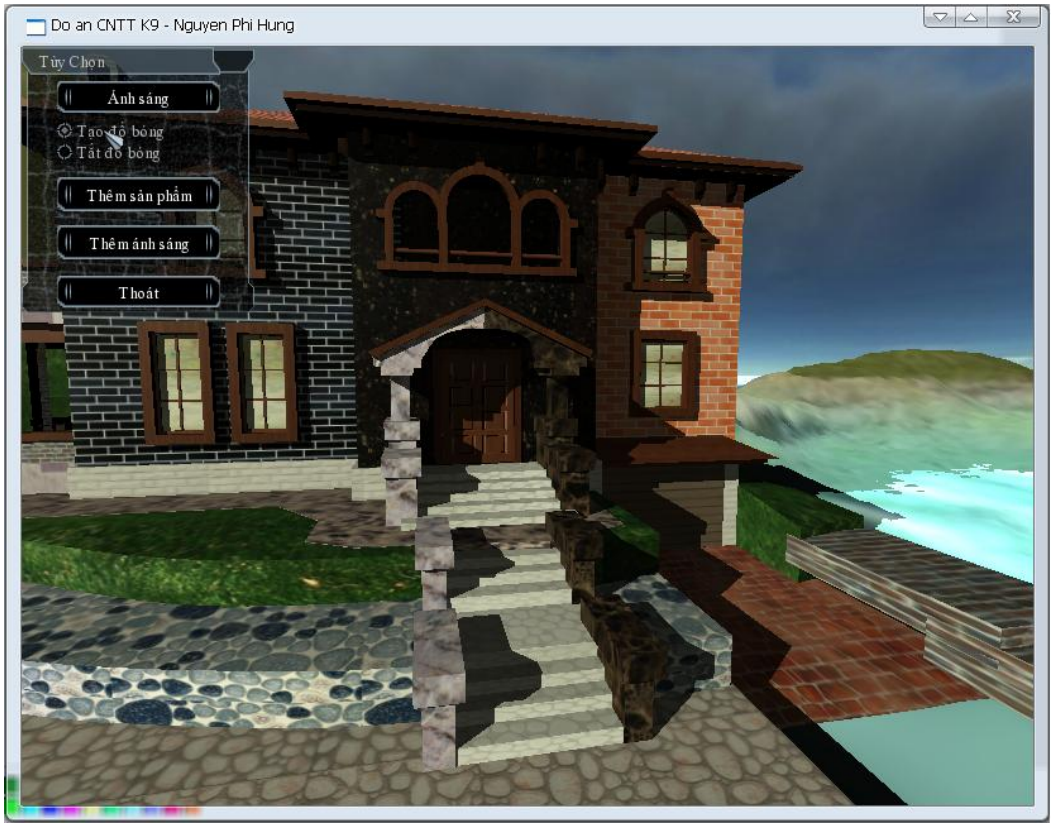
Hình 4.8. Thiết lập và chỉnh ánh sáng

- **Tắt bật đổ bóng**

Giúp tạo đổ bóng để các vật trông thật hơn



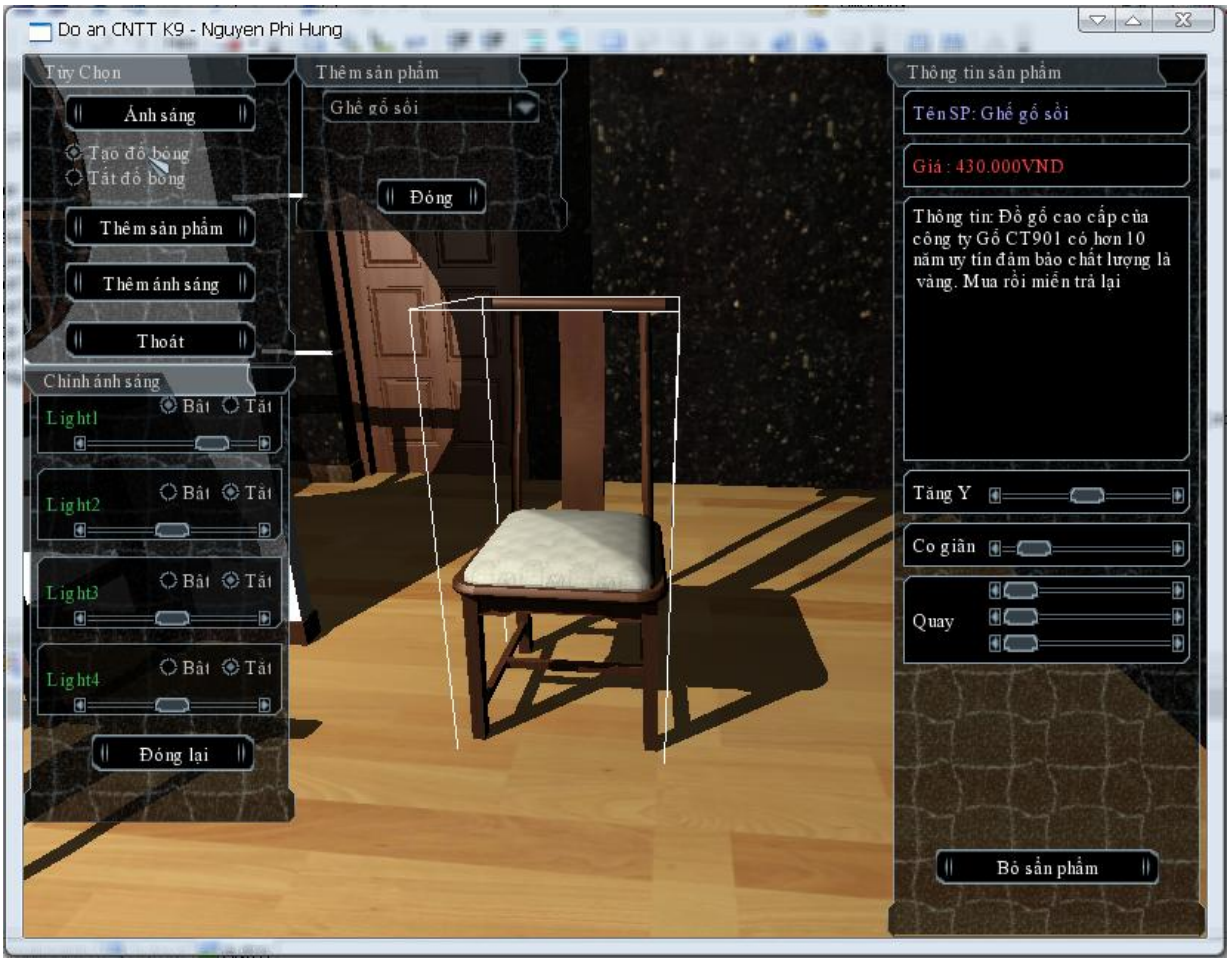
Hình 4.9. Trước khi bật đổ bóng



Hình 4.10. Sau khi bật đổ bóng

- **Thêm sản phẩm**

Người dùng sẽ được chọn sản phẩm trong 1 hộp ListBox, sau đó kich chuột xuống vị trí trên sàn muốn đặt sản phẩm.



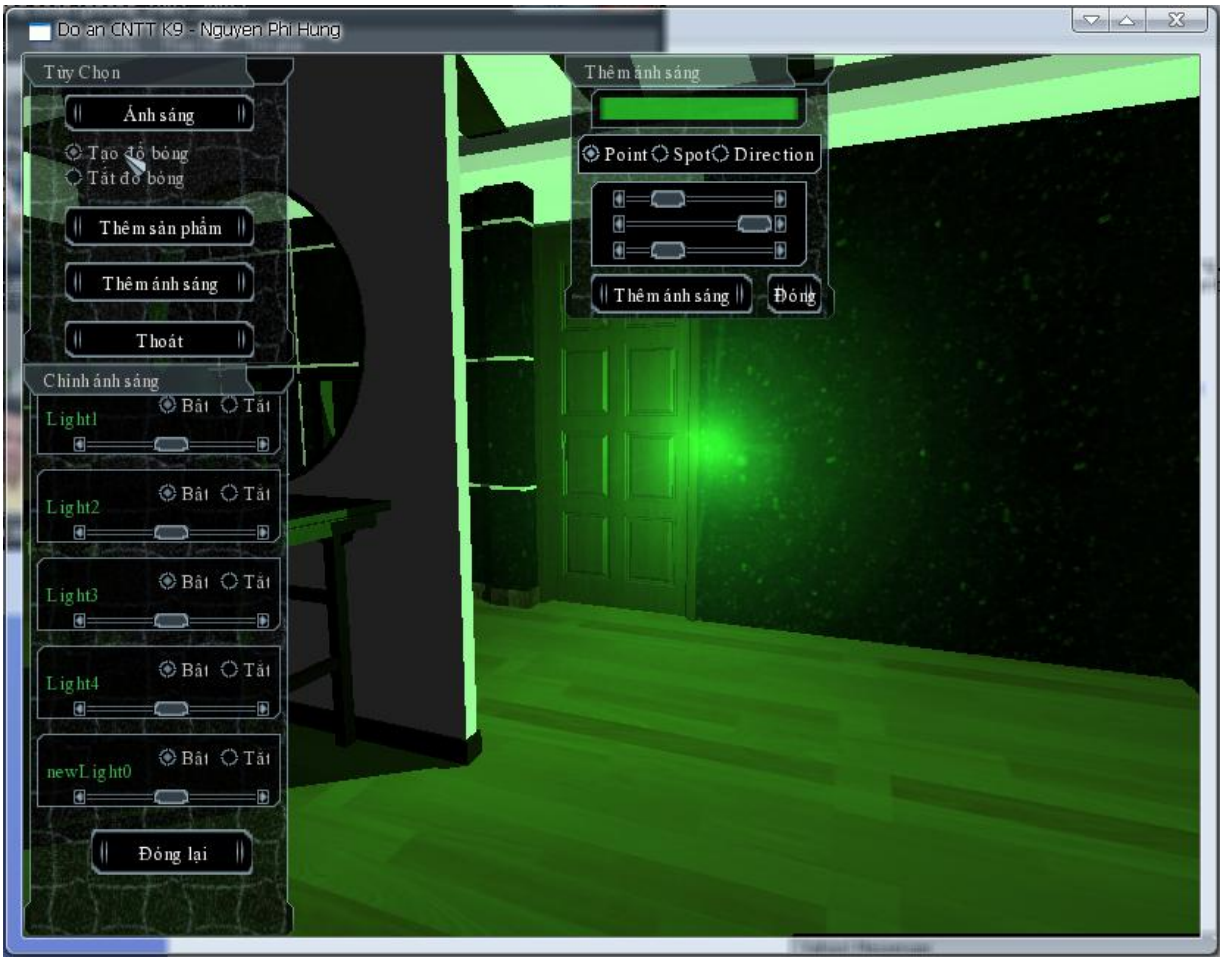
Hình 4.11. Sản phẩm được thêm và bảng thông tin sản phẩm

- **Xem thông tin sản phẩm, quay, co giãn và bỏ sản phẩm không ưng ý**

Mỗi sản phẩm sẽ đi kèm các thông tin: tên sản phẩm, giá, và thông tin về nhà sản xuất. Để xem các thông tin này, đơn giản là người dùng chỉ cần kich vào sản phẩm muốn xem. Ngoài ra họ còn có thể di chuyển sản phẩm, quay chúng theo 3 trục hoặc co giãn nếu muốn (tính năng co giãn chỉ mang tính đồ họa không thể áp dụng trong thực tế).

- **Thêm ánh sáng**

Thêm một ánh sáng mới với một trong 3 kiểu: Point – ánh sáng điểm, Spot – ánh sáng đèn, Direction – ánh sáng hướng (mặt trời). Giúp bạn nhìn rõ hơn hoặc phối màu đèn trong nhà theo ý muốn. Để thêm một ánh sáng mới đầu tiên là chọn màu, sau đó ấn nút "Thêm ánh sáng". Nếu bạn chọn kiểu ánh sáng là Spot hoặc Point thì phải kich vào một vị trí trên sàn nơi bạn muốn đặt ánh sáng.



Hình 4.12. Thêm một ánh sáng mới

4.4 Đánh giá kết quả đạt được và hướng phát triển

Chương trình đã tạo được một môi trường đồ họa 3 chiều của 1 căn nhà bao gồm đầy đủ không gian bên ngoài là một hòn đảo nhỏ trên biển, các phòng bên trong của ngôi nhà với đầy đủ nội thất. Cho phép người dùng đi lại, lựa chọn, thay đổi và xem thông tin về các sản phẩm nội thất, giúp đáp ứng được phần nào nhu cầu muốn xem căn nhà tương lai của người mua.

Tuy nhiên, chương trình còn khá nhiều thiếu sót như là:

- Tốc độ chạy chương trình chậm
- Chưa thể tự động hóa để có thể thêm các căn phòng khác nhau mà không phải xây dựng lại chương trình.
- Chưa thể lưu file chứa các thay đổi của người dùng về căn nhà.
- Chưa thể thao tác với mô hình cơ sở dữ liệu như là SQL Server 2000 mà phải tải vào nhiều file được tự tạo ra từ trước để lấy vào các thông tin cần thiết cho việc chạy chương trình.

Chính vì vậy việc khắc phục các thiếu sót trên là hướng phát triển tương lai của chương trình nhằm tạo ra một sản phẩm thương mại hoàn thiện để phục vụ khách hàng ngày một tốt hơn.

Kết luận

Lĩnh vực đồ họa 3 chiều ngày càng phát triển và có nhiều đóng ghóp quan trọng trong cuộc sống của con người. Nhu cầu tạo ra các thế giới ảo 3 chiều ngày càng nhiều. Với đề tài "Tìm hiểu về một số kỹ thuật đồ họa và ứng dụng", khóa luận này đã trình bày được tổng quan về các kỹ thuật đồ họa 3 chiều cơ bản, trong đó tập trung đi sâu vào nghiên cứu và sử dụng thư viện đồ họa mã nguồn mở Ogre để xây dựng được một ngôi nhà 3 chiều ảo - Bước đi đầu tiên để tạo ra các ứng dụng có quy mô lớn hơn và hoàn thiện hơn.

Trong tương lai em sẽ tiếp tục nghiên cứu thêm để có thể hoàn thiện hơn nữa về giao diện và tính năng của chương trình nhằm trở thành một sản phẩm thương mại hoàn thiện, mang tính chuyên nghiệp cao.

Tuy nhiên do hạn chế về điều kiện và thời gian, khoá luận sẽ không thể tránh khỏi những thiếu xót. Kính mong được sự đóng góp ý kiến của thầy cô và các bạn để em có thể hoàn thiện hơn đề tài nghiên cứu của mình trong đợt làm khoá luận tốt nghiệp này.

Trân trọng cảm ơn!

Tài liệu tham khảo

[1.] James D.Foley, Andrie van Dam, Steven K.Feiner, Jonhn F. Hughes, Computer Graphics Principles and Practice, Addison Wesley, 1994.

[2.] Hoàng Kiếm, Dương Anh Đức, Lê Đình Duy, Vu Hải Quân. Giáo trình cơ sở Đồ hoạ Máy tính, NXB Giáo dục, 2000.

[3.] Lê Tấn Hùng, Huỳnh Quyết Thắng. Kỹ thuật đồ hoạ máy tính, NXB khoa học và kỹ thuật, 2002.

[4.] Steven Harrington, Computer Graphics A Programming Approach, McGraw Hill International Edition, 1987.

[5.] Pro OGRE 3D Programming