Mục lục

Lời cảm ơn

Em xin chân thành gửi lời cảm ơn tới các thầy cô trường DHDL Hải Phòng trong những năm vừa rồi đã dạy dỗ vun đắp kiến thức để em có điều kiện hoàn thành đồ án tốt nghiệp này.

Em xin cảm ơn các thầy cô tại phòng nghiên cứu thực tại ảo – viện khoa học và công nghệ Việt Nam đã tạo điều kiện thuận lợi cho em nghiên cứu và phát triển đề tài trong quá trình làm đồ án.

Đặc biệt em xin cảm ơn thầy giáo PGS. TS Đỗ Năng Toàn khoa công nghệ thông tin viện khoa học và công nghệ Việt Nam đã chỉ bảo tận tình giúp em hoàn thành đồ án tốt nghiệp.

Cuối cùng em xin gửi lời biết ơn đến gia đình, bạn bè đã ủng hộ và giúp đỡ em trong suốt thời gian qua. Do trình độ bản thân có hạn nên không tránh khỏi những thiếu xót, mong thầy cô và các bạn góp ý giúp đỡ để em có thể hoàn thiện đồ án của mình.

Em xin chân thành cảm ơn!

Hải Phòng, ngày tháng 07 năm 2010

Sinh viên thực hiện

Triệu Minh Đức

MỞ ĐẦU

Đồ họa máy tính là một lãnh vực phát triển nhanh nhất trong tin học. Nó được áp dụng rộng rãi trong nhiều lãnh vực khác nhau thuộc về khoa học, kỹ nghệ, y khoa, kiến trúc và giải trí.

Thuật ngữ đồ họa máy tính (Computer Graphics) được đề xuất bởi nhà khoa học người Mỹ tên là William Fetter vào năm 1960 khi ông đang nghiên cứu xây dựng mô hình buồng lái máy bay cho hãng Boeing.

Các chương trình đồ họa ứng dụng cho phép chúng ta làm việc với máy tính một cách thoải mái và thân thiện nhất.

Năm 1966, Sutherland ở Học viện Công nghệ Massachusetts là người đầu tiên đặt nền bóng cho đồ họa 3D bằng việc phát minh ra thiết bị hiển thị trùm đầu (head-amounted display) được điều khiển bởi máy tính đầu tiên. Nó cho phép người nhìn có thể thấy được hình ảnh dưới dạng lập thể 3D. Từ đó đến nay đồ họa 3D trở thành một trong những lĩnh vực phát triển rực rỡ nhất của đồ họa máy tính.

Nó được ứng dụng rộng rãi trong hầu hết tất cả các lĩnh vực như Điện ảnh, Hoạt hình, kiến trúc và các ứng dụng xây dựng các mô hình thực tại ảo…..Và không thể không nhắc đến vai trò tối quan trọng của đồ họa 3D trong việc tạo ra các game sử dụng đồ họa hiện nay như Doom, Halflife…. Việc sử dụng đồ họa 3D trong game làm cho người chơi thích thú và có cảm giác như đang sống trong một thế giới thực. Có thể nói đồ họa 3D đã đang và sẽ tạo nên một nền công nghiệp game phát triển mạnh mẽ.

Mục đích chính của đồ họa 3D là tạo ra và mô tả các đối tượng, các mô hình trong thế giới thật bằng máy tính sao cho càng giống với thật càng tốt. Việc nghiên cứu các phương pháp các kỹ thuật khác nhau của đồ họa 3D cũng chỉ hướng đến một mục tiêu duy nhất đó là làm sao cho các nhân vật, các đối tượng, các mô hình được tạo ra trong máy tính giống thật nhất. Và một trong các phương pháp đó chính là sinh ảnh.

Nhận biết được sự quan trọng của bóng nên khóa luận này em muốn **"Tìm hiểu kỹ thuật sinh ảnh Ray Tracing".** Nội dung khóa luận bao gồm, Phần mở đầu, Phần kết luận và 3 chương nội dung, cụ thể:

Chương 1: Các kiến thức cơ bản của đồ họa 3D.

Chương này nói về các kiến thức cơ bản về ánh sáng, về hiển thị 3D và về các bộ đệm, và khái quát các kỹ thuật sinh ảnh.

Chương 2: Kỹ thuật sinh ảnh dựa vào Raytracing

Chương này đi vào chi tiết kỹ thuật sinh ảnh Ractracing.

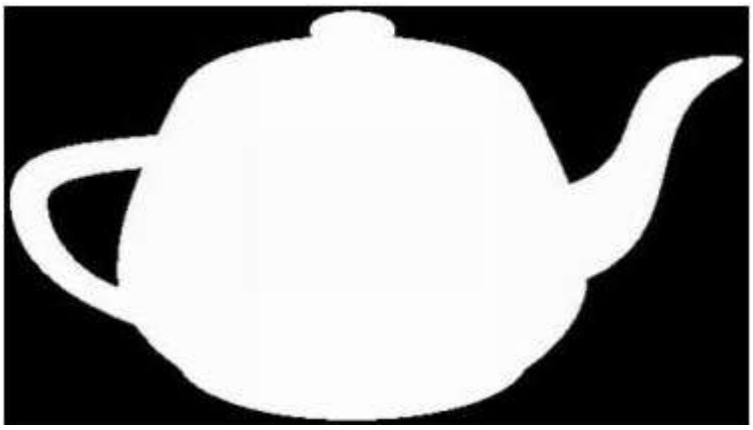
Chương 3: Chương trình thực nghiệm.

Chương 1: CÁC KIẾN THỨC CƠ BẢN CỦA ĐỒ HỌA 3D

1.1. ÁNH SÁNG

Ánh sáng trong đồ họa 3D đóng vai trò khá quan trọng. Và đặc biệt nó là thành phần không thể thiếu để tạo ra bóng. Có nguồn sáng chỉ chiếu theo một hướng nhất đinh (giống ánh sáng mặt trời), có nguồn sáng chiếu ra toàn khung cảnh….Trong một khung cảnh có thể có nhiều nguồn sáng. Các nguồn sáng này có thể được tắt bật từng cái giống như ta tắt đèn bằng công tắc vậy. Theo mô hình ánh sáng của OpenGl thì ánh sáng gồm có 4 thành phần chính: Emissive Light, Ambient Light, Diffuse Light, Specular Light. Các thành phần này có thể được tính toán độc lập với nhau, và cuối cùng được kết hợp lại với nhau.

Ambient Light là ánh sáng bị phân rã bởi môi trường và không thể xác định hướng của chúng. Nếu trong một khung cảnh ta không xác định nguồn sáng thì kết quả đưa ra cũng giống như khi chúng ta sử dụng Ambient Light.



Hình 1.1: Chiếc ấm được chiếu bằng Ambient Light

Diffuse Light (ánh sáng khuếch tán) là ánh sáng chiếu theo một hướng nhất, tuy nhiên khi nó gặp một bề mặt nó sẽ bị phân rã bằng nhau về mọi hướng, Vì thế nó sáng bằng nhau cho dù có đặt mắt nhìn ở đâu chăng nữa. Mọi nguồn sáng đến từ một điểm hay từ một hướng nhất định đều có thành phần Diffuse Light.

Hình 1.2: Ấm chè được chiều bằng Diffuse Light

Specular Light là ánh sáng phản xạ. Khi gặp một bề mặt nó sẽ phản xạ lại đúng theo quy luật phản xạ. Nó có thể được nhìn thấy trên những bề mặt cong.

Hình 1.3. Ấm chè được chiếu bằng Specular Light

1.2. HIỂN THỊ 3D

1.2.2. Giới thiệu

Các đối tượng trong mô hình 3D được xác định với tọa độ thế giới. Cùng với các tọa độ của đối tượng, người dùng cũng phải xác định vị trí và hướng của camera ảo trong không gian 3D và xác định vùng nhìn (là một vùng không gian được hiển thị trên màn hình)

Việc chuyển từ các tọa độ thế giới sang tọa độ màn hình được thực hiện theo 3 bước (hình 1.4):

- Bước đầu tiên thực hiện một phép biến đổi để đưa camera ảo trở về vị trí và hướng tiêu chuẩn. Khi đó điểm nhìn (eyepoint) sẽ được đặt ở gốc tọa độ, hướng

nhìn trùng với hướng âm của trục Z. Trục X chỉ về phía phải và trục Y chỉ lên phía trên trong màn hình. Hệ tọa độ mới này sẽ được gọi là Hệ tọa độ Mắt (Eye Coordinate System). Phép biến đổi từ tọa độ thế giới sang các tọa độ mắt là một phép biến đổi affine, được gọi là phép biến đổi hiển thị (Viewing Transformation). Cả tọa độ thế giới và tọa độ mắt đều được biểu diễn bởi tọa độ đồng nhất (Homogeneous Coordinates) với w=1.

- Bước thứ 2. Tọa độ mắt được chuyển qua tọa độ của thiết bị chuẩn hóa (Nomalized Device Coordinates) để cho vùng không gian mà ta muốn nhìn được đặt trong một khối lập phương tiêu chuẩn:

$$-1 \le x \le +1, -1 \le y \le +1, -1 \le z \le +1$$

Các điểm ở gần điểm nhìn (điểm đặt camera) hơn sẽ có thành phần z

nhỏ hơn.

Bước này sẽ gồm 3 bước con.

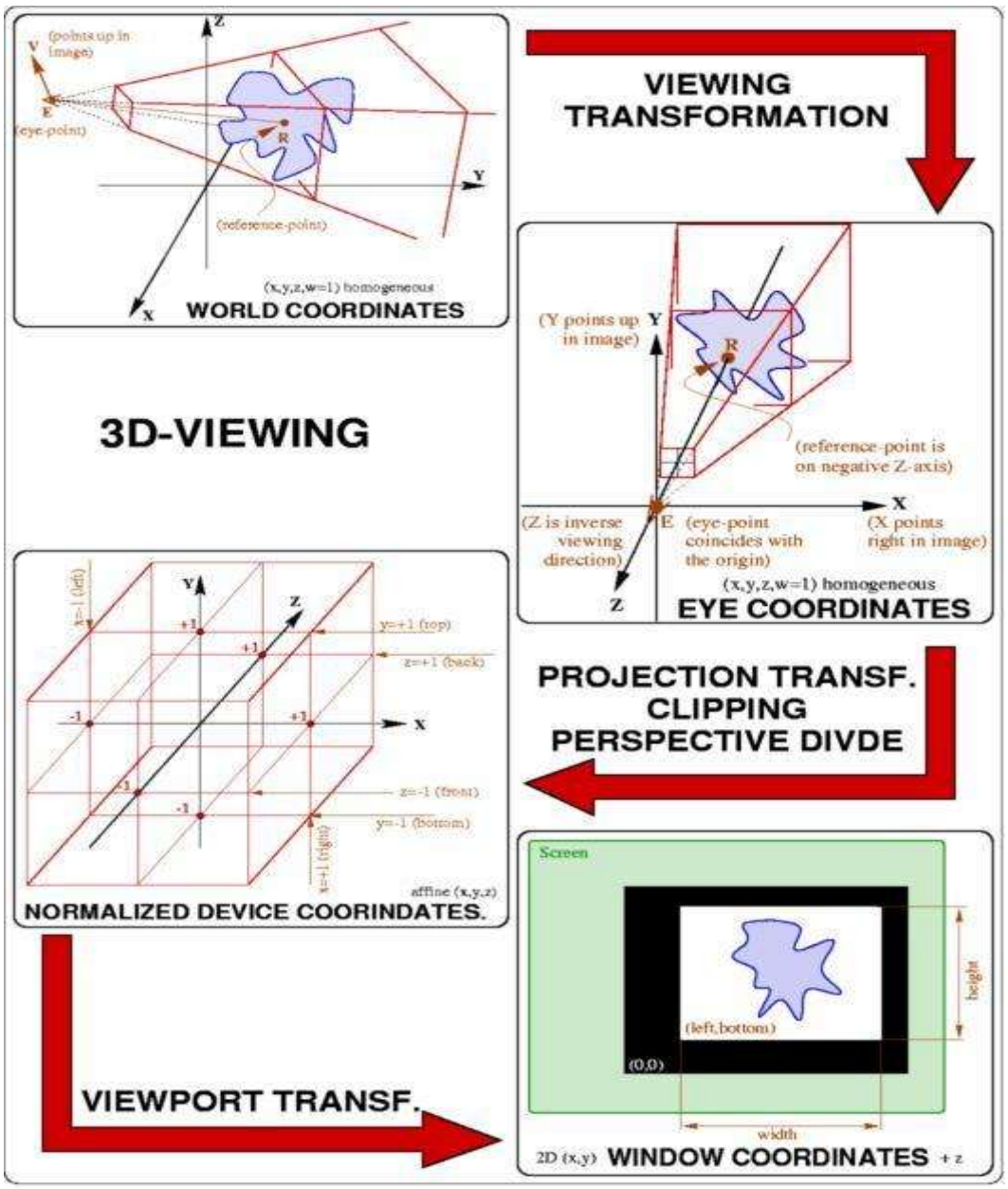
- Bước cuối cùng, phép biến đổi cổng nhìn (Viewport Transformation) là sự kết hợp của 1 phép co giãn tuyến tính và 1 phép tịnh tiến. Sẽ chuyển thành phần x và y của tọa độ thiết bị chuẩn hóa $-1 \le x \le 1, -1 \le y \le 1$ sang tọa độ Pixel của màn hình. Thành phần z ($-1 \le z \le 1$) được chuyển sang đoạn [0,1] và sẽ được sử dụng như là giá trị chiều sâu (Depth-Value) trong thuật toán Z-Buffer (bộ đệm Z) được sử dụng cho việc xác định mặt sẽ được hiển thị.

  Bước thứ 2 bao gồm 3 bước con.

  - Một phép chiếu chuyển từ vùng nhìn sang 1 khối lập phương tiêu chuẩn với tọa độ đồng nhất: $-1 \le x \le 1, -1 \le y \le 1, -1 \le z \le 1$. Trong trường hợp sử dụng phép chiếu trực giao, vùng nhìn này sẽ có dạng một ống song song 3D với các mặt song song với các mặt của hệ tọa độ mắt. Trong trường hợp sử dụng phép chiếu đối xứng, vùng nhìn sẽ là một hình tháp cụt với đầu mút là gốc tọa độ của hệ tọa độ mắt. Hệ tọa độ đồng nhất (4 thành phần) thu được sau phép chiếu được gọi là hệ tọa độ cắt (Clipping Coordinate System). Phép chiếu sẽ là một phép biến đổi affine trong trường hợp phép chiếu là phép chiếu trực giao. Nếu phép chiếu là phép

chiếu phối cảnh sẽ không phải là một phép biến đổi affine (Vì w sẽ nhận một giá trị khác 1)

- Bước tiếp theo, các vùng của không gian hiển thị mà không nằm trong khối tiêu chuẩn đó (Khối này còn được gọi là khối nhìn tiêu chuẩn) sẽ bị cắt đi. Các đa giác, các đường thẳng được chứa trong hoặc là có một phần ở trong sẽ được thay đổi để chỉ phần nằm trong khối nhìn tiêu chuẩn mới được giữ lại. Phần còn lại không cần quan tâm nhiều nữa.
- Sau khi cắt gọt, các tọa độ đồng nhất sẽ được chuyển sang tọa độ của thiết bị bằng cách chia x,y,z cho w. Nếu w nhận 1 giá trị đúng qua phép chiếu, thì phép chia này sẽ cho các động phối cảnh mong muốn trên màn hình. Vì lý do đó., phép chia này còn được gọi là phép chia phối cảnh (Perspective Division)



Hình 1.4: Tổng quan về hiển thị 3D và các phép chiếu.

1.2.2. Biểu diễn điểm và các phép biến đổi

Sự chuyển đổi từ tọa độ thế giới sang tọa độ của thiết bị là một chuỗi của các phép biến đổi affine và các phép chiếu. trong không gian Decarts 3 chiều.

Các phép biến đổi affine và các phép chiếu trong không gian Decarts 3 chiều có thể được biểu diễn tốt nhất bởi các ma trận 4×4 tương ứng với các tọa độ đồng nhất (Homogeneous coordinates) (x,y,z,w). Điểm 3D với tọa độ đồng nhất (x,y,z,w) sẽ có tọa độ affine là (x/w,y/w,z/w).

Mối quan hệ giữa tọa độ affine và tọa độ đồng nhất không phải là quan hệ 1-1. Cách đơn giản nhất để chuyển từ tọa độ affine (x,y,z) của một điểm sang tọa độ đồng nhất là đặt w=1: (x,y,z,1). Chúng ta thừa nhận rằng tất cả các tọa độ thế giới được biểu diễn bằng cách này.

Ta sẽ biểu diễn các phép biến đổi affine (như là co giãn (scaling transformations), phép quay (rotations), và phép tịnh tiến (translations)) bằng các ma trận mà sẽ không làm thay đổi thành phần w (w=1).

● Tịnh tiến bởi véc tơ $\vec{T} = (T_x, T_y, T_z)$:

$$\mathbf{M_t}(\vec{T}) = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{M_t}(\vec{T}) \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}$$

● Phép co giãn theo các nhân tố $\vec{S} = (S_x, S_y, S_z)$

$$\mathbf{M_s}(\vec{S}) = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{M_s}(\vec{S}) \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S_x \cdot x \\ S_y \cdot y \\ S_z \cdot z \\ 1 \end{pmatrix}$$

● Phép quay quanh gốc tọa độ mà theo đó tập các véc tơ chuẩn tắc là $\{\vec{u}, \vec{v}, \vec{n}\}$, trực giao từng đôi một, sẽ được chuyển về $\{\vec{X}, \vec{Y}, \vec{Z}\}$.

$$\mathbf{M_r}(\vec{u}, \vec{v}, \vec{n}) = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

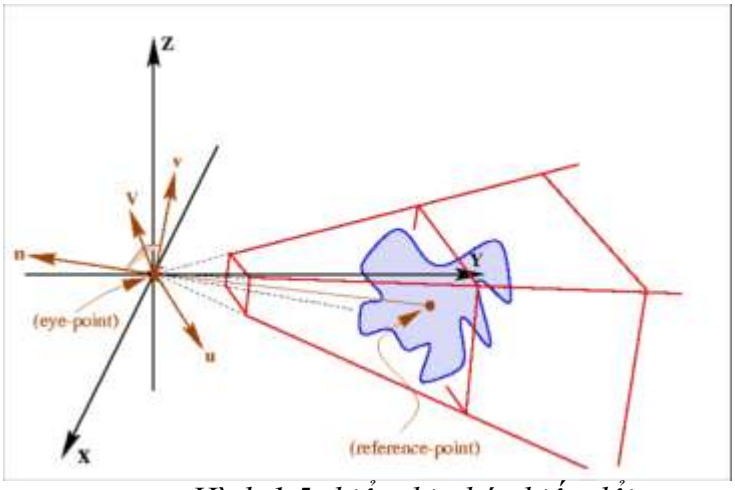
1.2.3. Phép biến đổi hiển thị

Phép biến đổi hiển thị sẽ đưa một camera ảo được cho tùy ý về một camera với điểm nhìn trùng với gốc tọa độ và hướng nhìn dọc theo chiều âm của trục Z (xem hình 2.1) Trục Y sau phép biến đổi tương ứng sẽ chỉ lên phía trên của màn hình. Trục X sẽ chỉ về phía phải.

Một cách thuận tiện để xác định vị trí của camera ảo là cho sãn vị trí của điểm nhìn $\vec{E}$, Một điểm trong khung nhìn $\vec{R}$ (điểm tham chiếu) và một hướng $\vec{V}$ sẽ chỉ lên phía trên trong màn hình.

Phép biến đổi hiển thị sẽ gồm 2 bước:

● Một phép tịnh tiến sẽ đưa điểm nhìn $\vec{E}$ về gốc tọa độ. Ma trận biến đổi tương ứng sẽ là $M_t(-\vec{E})$. Kết quả sẽ như sau:



Hình 1.5: hiển thị phép biến đổi.

● Một phép quay sẽ chuyển hướng nhìn ngược về trục Z, quay vectơ $\vec{V}$ về mặt phẳng YZ. Vector $\vec{V}$ sẽ chỉ được quay về trùng với trục Y nếu $\vec{V}$ vuông góc với hướng nhìn. Trước hết ta sẽ xây dựng tập các véc tơ chuẩn tắc phù hợp trong tọa độ thế giới.

$$\vec{n} = \frac{\vec{E} - \vec{R}}{\left\|\vec{E} - \vec{R}\right\|}$$ Ngược với hướng nhìn $\rightarrow \vec{Z}$ $(\overrightarrow{Oz})$

$$\vec{u} = \frac{\vec{V} \times \vec{n}}{\left\|\vec{V} \times \vec{n}\right\|}$$ Chỉ về phía phải, vuông góc với $\vec{n} \rightarrow \vec{X}$

$\vec{v} = \vec{n} \times \vec{u}$ Chỉ lên giống $\vec{V}$, nhưng vuông góc với $\vec{n}$ và $\vec{u} \rightarrow \vec{Y}$

Như vậy ma trận của phép quay sẽ là: $M_r(\vec{u},\vec{v},\vec{n})$

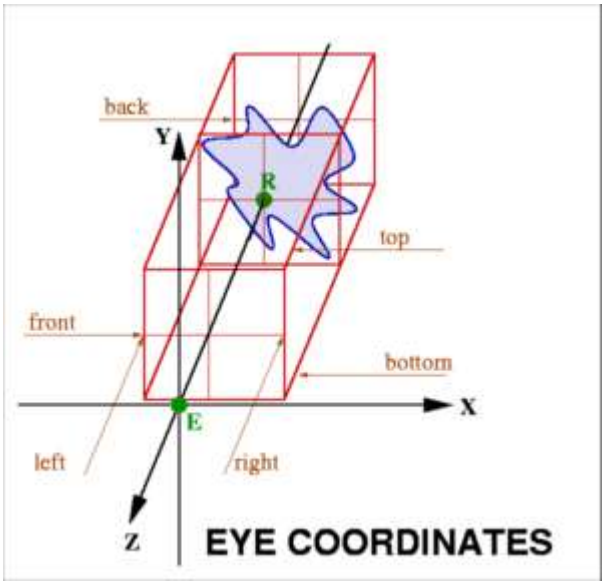
Và do đó ma trận của phép biến đổi sẽ là:

$$\mathbf{M_r}(\vec{u},\vec{v},\vec{n}) \cdot \mathbf{M_t}(-\vec{E}) = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -E_x \\ 0 & 1 & 0 & -E_y \\ 0 & 0 & 1 & -E_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z & -\vec{u}\cdot\vec{E} \\ v_x & v_y & v_z & -\vec{v}\cdot\vec{E} \\ n_x & n_y & n_z & -\vec{n}\cdot\vec{E} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Trong đó $\vec{u},\vec{v}$ và $\vec{v}$ được tính từ $\vec{E}$, $\vec{R}$ và $\vec{V}$

1.2.4. Phép chiếu trực giao

Trong trường hợp phép chiếu trực giao, vùng không gian hiển thị là một ống song song trong hệ tọa độ mắt. Các mặt của ống song song này song song với các mặt của hệ tọa độ mắt. Kích thước và vị trí của vùng không gian hiển thị được xác định bởi tọa độ mắt x_{left}, x_{right}, y_{bottom}, y_{top}, z_{front} và z_{back} . (x_{left}, y_{bottom}) và (x_{right}, y_{top}) xác định một cửa sổ trong mặt phẳng chiếu (hoặc là bất kỳ mặt nào song song với mặt XY) mà vùng không gian hiển thị sẽ được hiển thị trên đó. Cửa sổ này phải được đưa về dạng hình vuông $[-1,+1]^2$. z_{front} và z_{back} định nghĩa 2 mặt phẳng cắt trước và cắt sau. Tọa độ của tất cả các điểm trong không gian (hoặc ít nhất là những điểm ta muốn nhìn) phải thỏa mãn $z_{back} \leq z \leq z_{front}$. Khoảng giá trị của z phải được đưa về các giá trị chiều sâu (depth value) nằm trong đoạn [-1,+1]. Các điểm gần mắt hơn sẽ có giá trị chiều sâu nhỏ hơn.



Hình 1.6: Vùng không gian hiển thị của phép chiếu trực giao

Phép chiếu trực giao thu được bằng cách thực hiện các phép biến đổi sau theo thứ tự:

● Phép tịnh tiến $M_t(-\vec{M})$ sẽ đưa tâm của vùng không gian hiển thị về gốc tọa độ của hệ tọa độ mắt.

$$\vec{M} = \left(\frac{x_{\text{right}} + x_{\text{left}}}{2}, \frac{y_{\text{top}} + y_{\text{bottom}}}{2}, \frac{z_{\text{front}} + z_{\text{back}}}{2} \right)$$

● Một phép co giãn để đưa kích thước của vùng hiển thị về 2 đơn vị mỗi chiều.

● Một phép đối xứng qua mặt XY để các điểm nằm gần hơn sẽ nhận giá trị z nhỏ hơn.

Phép co giãn và phép đối xứng ở trên có thể thu được chỉ bằng một phép biến đổi đơn: $M_s(\vec{S})$ với:

$$\vec{S} = \left(\frac{2}{x_{\text{right}} - x_{\text{left}}}, \frac{2}{y_{\text{top}} - y_{\text{bottom}}}, \frac{-2}{z_{\text{front}} - z_{\text{back}}} \right)$$

Như vậy ma trận của phép chiếu trực giao sẽ là:

$$\mathbf{M_s}(\vec{S}) \cdot \mathbf{M_t}(-\vec{M}) = \begin{pmatrix} \frac{2}{x_{\text{right}} - x_{\text{left}}} & 0 & 0 & -\frac{x_{\text{right}} + x_{\text{left}}}{x_{\text{right}} - x_{\text{left}}} \\ 0 & \frac{2}{y_{\text{top}} - y_{\text{bottom}}} & 0 & -\frac{y_{\text{top}} + y_{\text{bottom}}}{y_{\text{top}} - y_{\text{bottom}}} \\ 0 & 0 & \frac{-2}{z_{\text{front}} - z_{\text{back}}} & \frac{z_{\text{front}} + z_{\text{back}}}{z_{\text{front}} - z_{\text{back}}} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Thành phần z không thay đổi, bởi vì phép chiếu trực giao là một phép biến đổi affine. Phép chiếu này được sử dụng trong các ứng dụng cần đến các quan hệ hình học (các tỉ số khoảng cách) như là trong CAD.

1.2.5. Phép chiếu phối cảnh

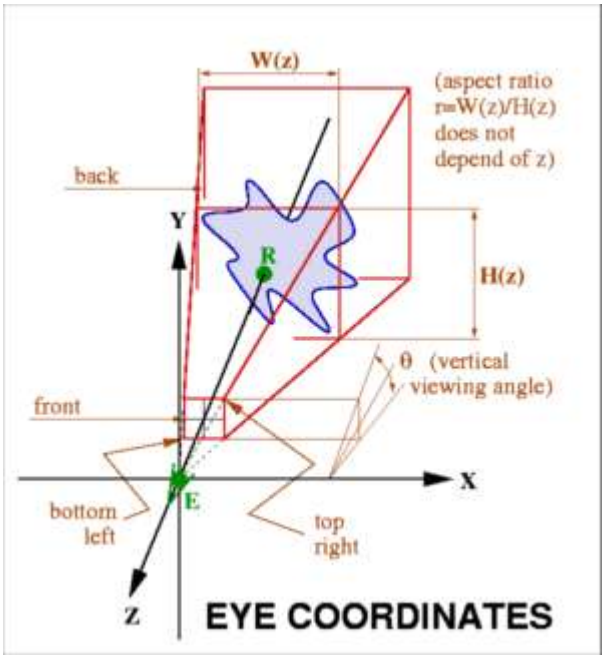
Phép chiếu phối cảnh phù hợp và gần hơn với quan sát của con người (bằng một mắt) trong thế giới 3D. Tất cả các điểm trên một đường thẳng đi qua điểm nhìn sẽ được ánh xạ lên cùng một điểm trong màn hình 2D. Điểm ảnh này được xác định bởi tọa độ thiết bị chuẩn hóa x và y. Nếu 2 điểm được ánh xạ vào cùng một điểm trên màn hình, ta cần phải xác định điểm nào sẽ được hiển thị bằng thuật toán Z-buffer, nghĩa là so sánh chiều sâu của chúng. Vì lý do này chúng ta cần định nghĩa một thành phần tọa độ khác của thiết bị chuẩn hóa là z sao cho nó là một hàm tăng đơn điệu của khoảng cách từ điểm đó đến mặt phẳng mắt XY. Khoảng cách từ một điểm trong không gian

đến mặt phẳng XY không bằng với khoảng cách từ điểm đó đến điểm nhìn (được đặt ở gốc tọa độ), nhưng nó sẽ được tính toán đơn giản hơn và cũng đủ để xác định được các mặt sẽ được hiển thị.

Như vậy, phép chiếu trực giao sẽ đưa một điểm (với tọa độ đồng nhất) trong hệ tọa độ mắt (x,y,z,1) về một điểm (tọa độ đồng nhất) trong hệ tọa độ cắt ($x^{'}$,$y^{'}$,$z^{'}$,$w^{'}$). Sau đó các tọa độ của thiết bị chuẩn hóa (affine) ($x^{''}$,$y^{''}$,$z^{''}$) sẽ thu được bằng cách chia $x^{'}$,$y^{'}$,$z^{'}$ cho $w^{'}$ (Phép chia phối cảnh):

$$(x'', y'', z'') = (\frac{x'}{w'}, \frac{y'}{w'}, \frac{z'}{w'})$$

Với phép chiếu phối cảnh, vùng không gian hiển thị là một hình tháp cụt với đầu mút là gốc tọa độ.



Hình 1.7: Vùng không gian hiển thị của phép chiếu phối cảnh cân xứng (Symmetrical Perspective Projection)

Trong trường hợp tổng quát, vùng này được xác định hoàn toàn bởi các thành phần tọa độ z (z_{front} và z_{back}) của các mặt cắt trước và cắt sau và một mặt cắt bất kỳ của vùng nhìn mà vuông góc với trục Z (Ví dụ đó là mặt $z = z_{front}$). Mặt cắt này là một hình chữ nhật được xác định bởi điểm trái dưới (x_{left},y_{bottom}) và điểm phải trên (x_{right}, y_{top}). Các mặt cắt trước và cắt sa phải được xác định sao cho mọi điểm trong vùng hiển thị phải có thành phần z thỏa mãn ($z_{front} \geq z \geq z_{back}$) trong hệ tọa độ mắt.

Phép chiếu phối cảnh đối xứng là rất quan trọng. Trong trường hợp này, điểm tham chiếu được chiếu lên trung tâm của màn hình. Vùng hiển thị sau đó sẽ được xác định một cách dễ dàng hơn bằng cách cho một góc nhìn đứng θ_y và tỉ số r = W(z)/H(z) không phụ thuộc vào z. Trong trường hợp này ta sẽ có:

$$x_{\text{left}} = +\frac{W}{2}z_{\text{front}} \quad x_{\text{right}} = -\frac{W}{2}z_{\text{front}} \quad y_{\text{bottom}} = +\frac{H}{2}z_{\text{front}} \quad y_{\text{top}} = -\frac{H}{2}z_{\text{front}}$$

Với :

$$H = 2\tan(\frac{\theta_y}{2}) \quad W = rH.$$

Chúng ta sẽ tìm ma trận biến đổi (4x4) để đưa vùng hiển thị hình tháp cụt về khối lập phương tiêu chuẩn trong tọa độ đồng nhất:

$$-1 \le x'/w' \le +1, -1 \le y'/w' \le +1, -1 \le z'/w' \le +1$$

● Đầu tiên, chúng ta sẽ xét trường hợp phép chiếu phối cảnh đối xứng với $\theta_y = 90^\circ$ và r = 1 (Cửa sổ hình vuông). Phép chiếu xuyên tâm (Với tâm là gốc tọa độ) của vùng hiển thị lên mặt z = -1 sẽ là hình vuông $[-1, +1]^2$.

Phép chiếu xuyên tâm này được mô tả bằng ma trận biến đổi sau:

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \qquad \mathbf{P} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ -z \end{pmatrix}$$

Điểm qua phép biến đổi sẽ có tọa độ (x/-z, y/-z, -1), đây chính là giao điểm của mặt phẳng z = -1 với đường thẳng nối gốc tọa độ với điểm (x,y,z) bất kỳ trong vùng hiển thị.

Phép biến đổi này chỉ có tác dụng khi ta không cần quan tâm đến z'. Là trường hợp ta không cần quan tâm đến việc xác định xem mặt nào sẽ che

mặt nào.

● Ma trận chiếu P ở trên không làm thay đổi thành phần tọa độ z. Sau phép chia cho w' = -z' chúng ta luôn thu được z'' = -1 bởi vì phép chia phối cảnh không còn có khả năng xác định z'' như là một hàm tuyến tính của z. Tuy nhiên ta vẫn có cách để xây

dựng ma trận chiếu để $z'' = x'/w'$ là một hàm tăng đơn điệu (không tuyến tính) của chiều sâu –z của một điểm trong khoảng [-1, +1]. Và như vậy ta vẫn có thể xác định được các bề mặt được hiển thị.

Thấy rằng z' được xác định bởi các thành phần trong hàng thứ 3 của ma trận P. Chúng ta phải xác định các thành phần này để thu được các tác dụng mong muốn. Ma trận biến đổi mới sẽ được KH là Q. z' không cần phụ thuộc vào x và y, do đó 2 thành phần đầu ta cho bằng 0. Chúng ta gọi 2 thành phần còn lại trong hàng thứ 3 là a và b. Một điểm bất kỳ trong hệ tọa độ mắt (x,y,z,1) sẽ được biến đổi thành:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \mathbf{Q} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

với:

$$\mathbf{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Từ đó ta suy ra:

$$\frac{z'}{w'} = \frac{a \cdot z + b}{-z}$$

Chúng ta muốn ánh xạ z_{front} vào -1, z_{back} vào +1 tức là:

$$\frac{a \cdot z_{front} + b}{-z_{front}} = -1$$

$$\frac{a \cdot z_{back} + b}{-z_{back}} = +1$$

Giải hệ phương trình trên ta có:

$$a = \frac{z_{front} + z_{back}}{z_{front} - z_{back}}$$

$$b = \frac{-2 \cdot z_{front} \cdot z_{back}}{z_{front} - z_{back}}$$

Với a và b thu được ở trên ta hoàn toàn có thể chắc chắn rằng z" = z'/w' là một hàm tăng đơn điệu (không tuyến tính) của z.

● Ma trận Q làm việc với $\theta_y = 90^\circ$ và r = 1. Trường hợp tổng quát sẽ được đưa về trường hợp đặc biệt này.

○ Một phép chiếu phối cảnh đối xứng với $\theta_y \neq 90^\circ$ và/hoặc $r \neq 1$ sẽ được đưa về trường hợp trước bằng một phép co giãn x và y bởi ma trận $M_s(\vec{S})$ với:

$$\vec{S} = (\tfrac{2}{W}, \tfrac{2}{H}, 1)$$

$$H = 2\tan(\frac{\theta_y}{2}) \quad W = rH.$$

Khi đó ma trận chiếu hoàn thiện cho phép chiếu đối xứng là:

$$\mathbf{Q} \cdot \mathbf{M_s}(\vec{S}) = \begin{pmatrix} \frac{1}{r \cdot \tan(\theta_y/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\theta_y/2)} & 0 & 0 \\ 0 & 0 & \frac{z_{front}+z_{back}}{z_{front}-z_{back}} & \frac{-2 \cdot z_{front} \cdot z_{back}}{z_{front}-z_{back}} \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$

○ Với phép chiếu phối cảnh không đối xứng vùng hiển thị đầu tiên được biến đổi để trục của nó trùng với trục Z. Để thực hiện việc này cần một phép tịnh tiến vuông góc với trục Z, qua một khoảng cách tương xứng với –z. Đầu mút của vùng hiển thị vẫn nằm ở gốc tọa độ và phải luôn ở đó.

Trung tâm $\left(\frac{x_{right} + x_{left}}{2}, \frac{y_{top} + y_{bottom}}{2}, z_{front} \right)$ của mặt trước của vùng hiển thị phải được ánh xạ vào điểm (0, 0, z_{front}). Phép biến đổi này được gọi là biến đổi cắt (Shearing Transformation). Ma trận cho phép biến đổi này là:

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & \frac{x_{right}+x_{left}}{-2z_{front}} & 0 \\ 0 & 1 & \frac{y_{top}+y_{bottom}}{-2z_{front}} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{S} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + \frac{x_{right}+x_{left}}{-2z_{front}} \cdot z \\ y + \frac{y_{top}+y_{bottom}}{-2z_{front}} \cdot z \\ z \\ 1 \end{pmatrix}.$$

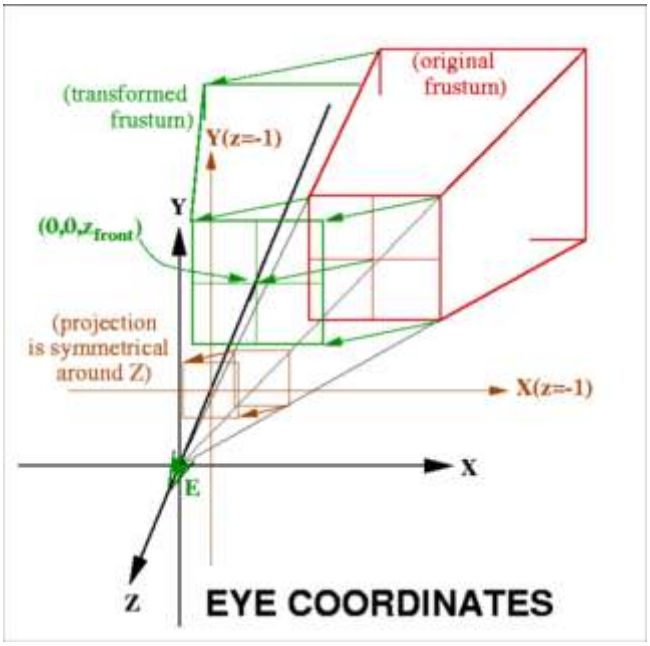
Phép chiếu lên mặt z = -1 giờ sẽ đối xứng qua trục Z.

Công việc cuối cùng cần làm là biến đổi các độ dài $\frac{x_{right} - x_{left}}{-z_{front}}$ và $\frac{y_{top} - y_{bottom}}{-z_{front}}$ của phép chiếu trong mặt z = -1 của x và y về 2 đơn vị bằng một phép co giãn bằng ma trận $M_s(\vec{S})$ với:

$$\vec{S} = \left(\frac{-2z_{\text{front}}}{x_{\text{right}} - x_{\text{left}}}, \frac{-2z_{\text{front}}}{y_{\text{top}} - y_{\text{bottom}}}, 1 \right).$$

Và cuối cùng ta có ma trận cho phép chiếu phối cảnh không đối xứng hoàn thiện là:

$$\mathbf{Q} \cdot \mathbf{M}_s(\vec{S}) \cdot \mathbf{S} = \begin{pmatrix} \frac{-2z_{\text{front}}}{x_{\text{right}} - x_{\text{left}}} & 0 & \frac{x_{\text{right}} + x_{\text{left}}}{x_{\text{right}} - x_{\text{left}}} & 0 \\ 0 & \frac{-2z_{\text{front}}}{y_{\text{top}} - y_{\text{bottom}}} & \frac{y_{\text{top}} + y_{\text{bottom}}}{y_{\text{top}} - y_{\text{bottom}}} & 0 \\ 0 & 0 & \frac{z_{\text{front}} + z_{\text{back}}}{z_{\text{front}} - z_{\text{back}}} & \frac{-2 \cdot z_{\text{front}} \cdot z_{\text{back}}}{z_{\text{front}} - z_{\text{back}}} \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$



Hình 1.8: Một phép chiếu phối cảnh không đối xứng được đưa về đối xứng bởi một phép biến đổi cắt (là một phép tịnh tiến vuông góc với trục Z qua một khoảng cách tương ứng với –z). Phép biến đổi này đưa trục của vùng hiển thị trùng vớihướng âm của trục Z

1.2.6. Phép biến đổi cổng nhìn

Phép biến đổi cổng nhìn chỉ gồm một phép tịnh tiến và một phép thay đổi tỉ lệ để:

● Tọa độ thiết bị chuẩn hóa (x, y) với $-1 \le x \le 1, -1 \le y \le 1$ được chuyển qua tọa độ pixel.

$$left \le x_w \le left + width$$

$$bottom \le y_w \le bottom + height$$

● Thành phần z với $-1 \le z \le 1$ được co lại trong đoạn $0 \le z_w \le 1$.

Giá trị z_w này sẽ được sử dụng để loại bỏ những bề mặt bị ẩn. Những điểm có giá trị z_w nhỏ sẽ nằm trước những điểm có giá trị z_w lớn hơn.

Xây dựng ma trận biến đổi là công việc đơn giản. Tuy nhiên sẽ hiệu quả hơn nếu ta thực hiện phép biến đổi một cách trực tiếp:

$$x_w = left + width \cdot \frac{x+1}{2}$$

$$y_w = bottom + height \cdot \frac{y+1}{2}$$

$$z_w = \frac{z+1}{2}$$

1.3. BỘ ĐỆM VÀ CÁC PHÉP KIỂM TRA

Một mục đích quan trọng của hầu hết các chương trình đồ họa là vẽ được các bức tranh ra màn hình. Màn hình là một mảng hình vuông của các pixel. Mỗi pixel đó có thể hiển thị được 1 màu nhất định. Sau các quá trình quét (bao gồm Texturing và fog…), dữ liệu chưa trở thành pixel, nó vẫn chỉ là các "mảnh" (Fragments). Mỗi mảnh này chứa dữ liệu chung cho mỗi pixel bên trong nó như là màu sắc là giá trị chiều sâu. Các mảnh này sau đó sẽ qua một loạt các phép kiểm tra và các thao tác khác trước khi được vẽ ra màn hình.

Nếu mảnh đó qua được các phép kiểm tra (test pass) thì nó sẽ trở thành các pixel. Để vẽ các pixel này, ta cần phải biết được màu sắc của chúng là gì, và thông tin về màu sắc của mỗi pixel được lưu trong bộ đệm màu (Color Buffer).

Nơi lưu trữ dữ liệu cho từng pixel xuất hiện trên màn hình được gọi là *bộ đệm* (Buffer). Các bộ đệm khác nhau sẽ chưa một loại dữ liệu khác nhau cho pixel và bộ nhớ cho mỗi pixel có thể sẽ khác nhau giữa các bộ đệm. Nhưng trong một bộ đệm thì 2 pixel bất kỳ sẽ được cấp cùng một lượng bộ nhớ giống nhau. Một bộ đệm mà lưu trữ một bít thông tin cho mỗi pixel được gọi là một *bitplane*. Có các bộ đệm phổ biến như Color Buffer, Depth Buffer, Stencil Buffer, Accumulation Buffer.

1.3.1. Bộ đệm chiều sâu

1.3.1.1. Khái niệm: Là bộ đệm lưu trữ giá trị chiều sâu cho từng Pixel. Nó được dùng trong việc loại bỏ các bề mặt ẩn. Giả sử 2 điểm sau các phép chiếu được ánh xạ vào cùng một pixel trên màn hình. Như vậy điểm nào có giá trị chiều sâu (z) nhỏ hơn sẽ được viết đè lên điểm có giá trị chiều sâu lớn hơn. Chính vì vậy nên ta gọi bộ đệm này là Z-buffer.

1.3.1.2. Depth test: Với mỗi pixel trên màn hình, bộ đệm chiều sâu lưu khoảng cách vuông góc từ điểm nhìn đến pixel đó. Nên nếu giá trị chiều sâu của một điểm được ánh xạ vào pixel đó nhỏ hơn giá trị được lưu trong bộ đêm chiều sâu thì điểm này được coi là qua Depth test (depth test pass) và giá trị chiều sâu của nó được thay thế cho giá trị lưu trong bộ đệm. Nếu giá trị chiều sâu của điểm đó lớn hơn giá trị lưu trong Depth Buffer thì điểm đó "trượt" phép kiểm tra chiều sâu. (Depth test Fail)

1.3.2. Bộ đệm khuôn

1.3.2.1. Khái niệm: Bộ đệm khuôn dùng để giới hạn một vùng nhất định nào đó trong khung cảnh. Hay nói cách khác nó đánh dấu một vùng nào đó trên màn hình. Bộ đệm này được sử dụng để tạo ra bóng hoặc để tạo ra ảnh phản xạ của một vật thể qua gương…

1.3.2.2. Stencil Test: Phép kiểm tra Stencil chỉ được thực hiện khi có bộ đệm khuôn. (Nếu không có bộ đệm khuôn thì phép kiểm tra Stencil được coi là luôn pass). Phép kiểm tra Stencil sẽ so sánh giá trị lưu trong Stencil Buffer tại một Pixel với một giá trị tham chiếu theo một hàm so sánh cho trước nào đó. OpenGL cung cấp các hàm như là

GL_NEVER, GL_ALWAYS, GL_LESS, GL_LEQUAL, GL_EQUAL, GL_GEQUAL, GL_GREATER hay là GL_NOTEQUAL. Giả sử hàm so sánh là GL_LESS, một "mảnh" (Fragments) được coi là qua phép kiểm tra (pass) nếu như giá trị tham chiếu nhỏ hơn giá trị lưu trong Stencil Buffer.

Ngoài ra OpenGL còn hỗ trợ một hàm là

glStencilOp(GLenum fail, GLenum zfail, GLenum zpass);

Hàm này xác định dữ liệu trong stencil Buffer sẽ thay đổi thế nào nếu như một "mảnh" pass hay fail phép kiểm tra stencil. 3 hàm fail, zfail và zpass có thể là GL_KEEP, GL_ZERO, GL_REPLACE, GL_INCR, GL_DECR …Chúng tương ứng với giữ nguyên giá trị hiện tại, thay thế nó với 0, thay thế nó bởi một giá trị tham chiếu, tăng và giảm giá trị lưu trong stencil buffer. Hàm fail sẽ được sử dụng nếu như "mảnh" đó fail stencil test. Nếu nó pass thì hàm zfail sẽ được dùng nếu Depth test fail và tương tự, zpass được dùng nếu như Depth test pass hoặc nếu không có phép kiểm tra độ sâu nào được thực hiện. Mặc định cả 3 tham số này là GL_KEEP.

1.4. KHÁI QUÁT CÁC KỸ THUẬT SINH ẢNH

Trong đồ họa máy tính 3D, **Rendering** - kết xuất đồ họa - là một quá trình sinh tạo một hình ảnh từ một mô hình bằng cách sử dụng một chương trình ứng dụng phần mềm. Nhiều thuật toán kết xuất đồ họa đã được nghiên cứu và phần mềm dùng trong quá trình kết xuất có thể áp dụng một số những kỹ thuật kết xuất để đạt được hình ảnh cuối cùng. Các kỹ thuật kết xuất đồ họa phổ biến được sử dụng là tạo ảnh điểm (rasterization), Chiếu tia (Ray casting) và Dò tia (Ray tracing).

Ray tracing là một kỹ thuật để sinh ảnh bằng cách tìm đường đi của ánh sáng qua các điểm ảnh trong một mặt phẳng ảnh và mô phỏng các hiệu ứng khi ánh sáng chạm vào bề mặt các đối tượng ảo. Kỹ thuật này dò theo đường đi của các tia sáng, bắt đầu từ Camera, tới bề mặt đầu tiên và sau đó phụ thuộc vào tính trong suốt hay phản xạ của bề mặt, xác định hướng đi tiếp theo của tia sáng. Ray tracing lần đầu tiên cho phép tính đến môi trường xung quanh trong sự chiếu sáng vật thể, cho phép tạo ra các khung hình có độ chân thực rất cao so với phương pháp kết xuất quét dòng thông thường. Ray tracing đặc biệt phù hợp với các ứng dụng có các ảnh được kết xuất chậm như ảnh tĩnh, phim hay các hiệu ứng truyền hình đặc biệt. Ray tracing có khả năng mô phỏng nhiều hiệu ứng quang học như phản xạ, khúc xạ, tán xạ, và quang sai màu.

Chương 2:

KỸ THUẬT SINH ẢNH DỰA VÀO RAYTRACING

2.1. KỸ THUẬT SINH ẢNH RAYTRACING

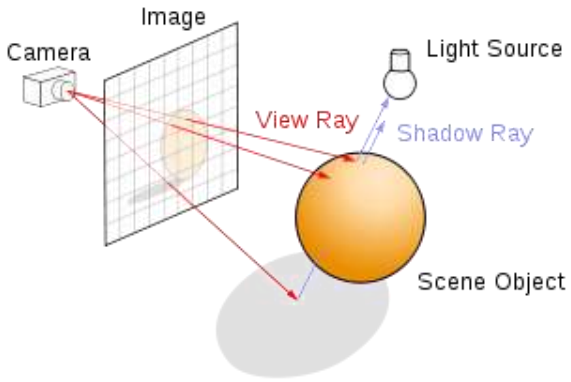
2.1.1. Nguyên lý giải thuật

Ray tracing là một phương pháp tạo ra các ảnh giống thật bằng máy tính, trong đó đường đi của mỗi tia riêng rẽ được lần theo từ người nhìn tới điểm tới đầu tiến.

Giải thuật Ray tracing được mô tả như sau:

1. Với mỗi điểm trên trên mặt phẳng của ảnh cần dựng, phóng một tia từ mắt người tới điểm đó.
2. Nếu tia đó không cắt vật nào thì màu của điểm đó trên ảnh là

   màu nền.
3. Nếu tia đó có cắt một vật nào đó, thì tìm điểm cắt gần điểm nhìn nhất. Tia sáng tại điểm cắt được tách làm hai tia là tia phản xạ và tia khúc xạ.



Hình 2.1: Giải thuật Ray tracing

4. Với từng tia phản xạ và khúc xạ lại tiếp tục thực hiện lần theo (tracing) bằng cách tính đệ quy từ bước 2.
5. Nếu độ xâu của của việc tính đệ quy đã đạt một tới giá trị cho trước, không tiếp tục lần theo nữa.
6. Ánh sáng tại điểm cắt được tính bằng phương trình:

$$I = I_{local} + K_r R + K_t T \qquad (2)$$

Với I: Cường độ sáng của điểm giao cắt.

I_{local}: Cường độ sáng nội tại của điểm giao cắt.

K_r: Hệ số phản xạ của bề mặt.

R: Lượng ánh sáng nhận được bằng cách lần theo tia phản xạ.

K_t: Hệ số truyền qua của bề mặt.

T: Lượng ánh sáng nhận được bằng cách lần theo tia khúc xạ.

7. Tính I_{local} bằng cách nối điểm giao cắt với tất nguồn sáng, kiểm tra xem đoạn đó có cắt bất cứ vật nào không, nếu có thì vật đó bị che khỏi nguồn sáng đó, nếu không thì I_{local} trong phương trình (2) được xác định nhờ vào tổng cường độ chiếu sáng của các nguồn sáng chiếu đến điểm đó và đặc tính của bề mặt tại điểm đó.

2.1.2. Đặc điểm giải thuật

Hình 2.2: hình ảnh Ray Tracing

Ray Tracing là phương pháp để tạo ta những hình ảnh giống như thật:

- Sự tương tác giữa ánh sáng và bóng tối với các vật thể được thể hiện giống như ta thấy trong tự nhiên.
- Sự kết hợp đơn giản của các hiệu ứng như bóng, sự phản xạ, khúc xạ.
- Mô phỏng theo đường đi giữa tia sáng và vật thể theo quy luật của quang hình học.

Những hình ảnh mà chúng ta thấy được là sự tổ hợp màu của hàng tỉ tia sáng đi vào mắt chúng ta.

Vì vậy, Ray Tracing đề ra phương pháp để tính toán màu sắc của tia sáng.

2.1.3. Ưu điểm

Ray Tracing là phương pháp tổng quát rất hữu hiệu để hiện thị các mặ cong bởi vì:

- Là tính đơn giản của thuật toán.
- Sử dụng nguyên lý quang hình nên tạo ra những hình ảnh rất giống thực tế.
- Áp dụng cho nhiều loại mặt cong.

2.1.4. Nhược điểm

- Xử lý nhiều hình ảnh đổ bóng và phản chiếu phức tạp khiến cho bộ xử lý phải thực hiện một khối lượng công việc khổng lồ.

Tuy nhiên, bộ xử lý đa nhân hiện nay sẽ giúp quá trình **ray-tracing** nhanh chóng được thực hiện

Hình 2.3: Multi Ray Tracing.

2.2. THUẬT TOÁN KẾT HỢP RAYTRACING VÀ RADIOSITY

Radiosity là giải thuật đầu tiên và nổi tiếng nhất, phương pháp này mô phỏng sự tương tác ánh sáng giữa các bề mặt với nhau. Giải thuật Ray tracing mô phỏng lần theo đường đi của tia sáng qua phản xạ, tán xạ của ánh sáng mỗi khi giao cắt với vật. Hai giải thuật này là hai nửa của việc chụp ảnh một căn phòng được chiếu sáng trong thực tế nên kết hợp được cả hai phương pháp này sẽ tạo ra được những bức ảnh có mày sắc rất gần với các bức ảnh thật.

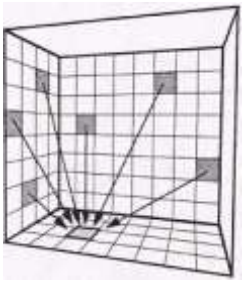
Vấn đề của việc kết hợp hai giải thuật này là mỗi giải thuật thường được triển khai trên các mô hình vật thể khác nhau để dựng ảnh đạt hiểu quả tính toán nhanh vì

hai mô hình này đều đòi hỏi khối lượng tính toán rất lớn. Theo đó, khi kết hợp hai phương pháp cần tìm những mô hình vật thể sao cho làm giảm khối lượng tính toán cũng như khắc phục những dị vật về hình ảnh khi kết hợp mô hình vật thể này với giải thuật này.

2.2.1. Radiosity

Radiosity là kỹ thuật dựng ảnh cho phép các vật thể có thể tương tác năng lượng ánh sáng với nhau giúp cho các vật trở nên nhìn thấy (có phát xạ năng lượng). Một miếng sẽ được chiếu sáng bằng năng lượng ánh sáng các miếng xung quanh chiếu đến. Sau đó phải sử dụng một số giải thuật như Z-buffer hoặc Ray tracing để hiển thị những thông số về các vật thể vừa được tính toán.

Hình 2.4: Bề mặt vật thể được chia nhỏ

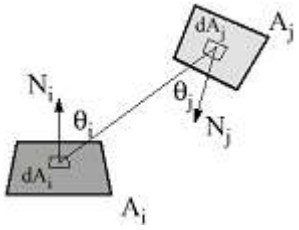


Hình 1.5: Miếng nhận ánh sáng từ các miếng xung quanh

Nguyên lý giải thuật

Giải thuật radiosity được mô tả như sau:

- Chia nhỏ các bề mặt vật thể thành những miếng nhỏ.
- Với mỗi cặp hai miếng bất kỳ trong cảnh, tính tỷ lệ giữa năng lượng rời miếng nguồn mà tới được miếng đích trên toàn bộ năng lượng rời khỏi miếng nguồn gọi là *form factor*.



Hình 2.6:Tính Form factor giữa hai miếng của cảnh'

- Áp dụng phương trình (1) để tính lần lượt tính năng lượng của mỗi miếng.

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F \qquad (1)$$

Với B_i : Tổng năng lượng phát xạ của miếng thứ i

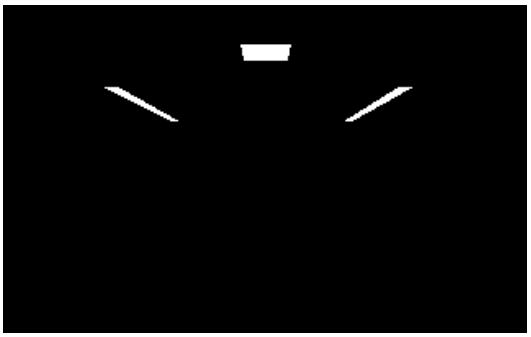
E_i : Năng lượng phát xạ nội tại của miếng đó.

ρ_i : Độ phản xạ của miếng thứ i.

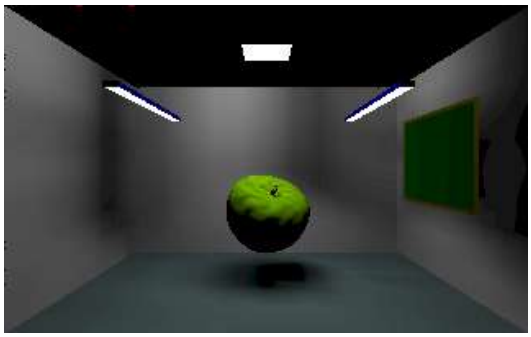
E_i : form factor giữa miếng thứ i và thứ j.

- Lặp lại nhiều lần thuật toán trên để các bề mặt tương tác năng lượng với nhau nhiều lần như trong thực tế.

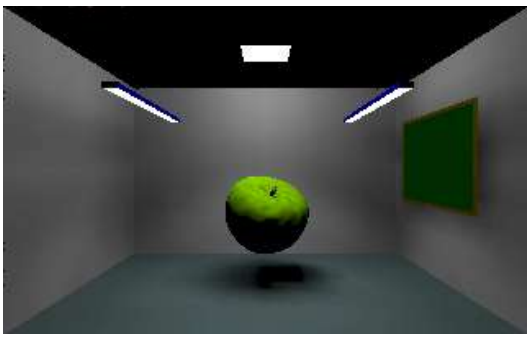
Hình (2.7) (2.8) (2.9) (2.10) thể hiện kết quả của việc tính Radiosity nhiều lần tạo nên bức ảnh có độ sáng gần với ảnh thật. Nếu số lần tính toán Radiosity tăng dần, màu sắc và độ sáng của những vật trong ảnh càng trở nên hài hòa và gần với thực tế hơn. Điều này có được là do giải thuật Radiosity mô phỏng sự tương tác giữa các bề mặt trong thực tế nên sau càng nhiều lần tương tác, các vật càng có độ sáng hợp lý.



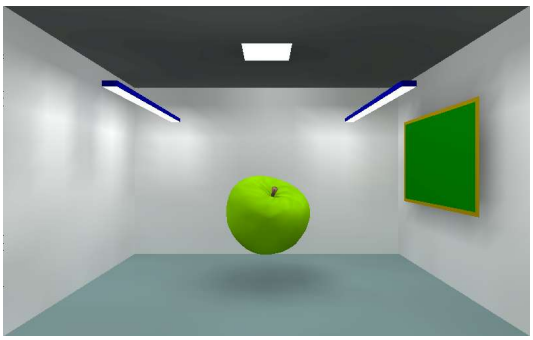
Hình 2.7: Ảnh trước khi thực hiện tính toán Radiosity



Hình 2.8: Ảnh khi thực hiện tính Radiosity 1 lần



Hình 2.9: Ảnh khi thực hiện tính Radiosity 2 lần



Hình 2.10: Ảnh khi thực hiện tính Radiosity nhiều lần

Đặc điểm của giải thuật Radiosity

Giải thuật Radiosity tạo cho bức ảnh có bóng mờ tốt giống như tương tác năng lượng ánh sáng giữa bề mặt các vật với nhau, không phụ thuộc góc nhìn hay không cần tính toán lại khi thay đổi góc nhìn. Những bức ảnh thực hiện theo giải thuật Radiosity có phân bố độ sáng hợp lý, các vật có bóng mờ nhưng thiếu bóng phản xạ (xem hình 2.10).

2.2.2. Thuật toán kết hợp hai giải thuật

Dựa trên đặc tính của hai giải thuật, giải thuật kết hợp của hai giải thuật này được đề ra như sau:

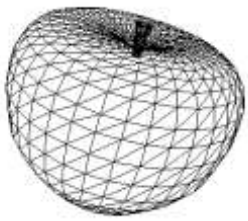
- Thực hiện việc tính toán theo giải thuật Radiosity trên các bề mặt của các vật thể.
- Bước tiếp theo thực hiện theo giải thuật Ray tracing trên các vật thể sau khi đã thực hiện tính toán theo giải thuật Radiosity.
- Bước 7 trong giải thuật Ray tracing không cần tính mà I_{local} trong phương trình (2) đã được tính với giải thuật Radiosity trước đó.

Mô hình vật thể

Có hai mô hình các vật thể cơ bản trong kỹ thuật dựng ảnh. Mô hình mà các vật thể có các mặt được xác định bởi các phương trình toán học như mặt cầu, mặt trụ, ... Mô hình này rất tốt cho việc tính toán theo giải thuật Ray tracing vì việc tính toán giao cắt của một tia với một bề mặt được tính toán đơn giản và số lượng bề mặt cần xét xem có giao cắt với một tia tương đối nhỏ. Tuy nhiên, mô hình vật thể này rất khó

triển khai khi thực hiện giải thuật Radiosity do giải thuật Radiosity yêu cầu các bề mặt cần được chia nhỏ thì việc tính toán phân bố nguồn sáng trên bề mặt mới đạt được gần thực tế. Nhưng việc chia nhỏ các bề mặt được mô tả bởi các phương trình toán học là rất phức tạp và chưa có giải thuật

hiệu quả.



Hình 2.11: Mô hình vật thể là lưới điểm

Mô hình vật thể thứ hai được xem xét là mô hình vật thể mà các bề mặt là một lưới điểm, các bề mặt đã được chia nhỏ thành các miến nhỏ hơn để tiện cho việc tình toán giải thuật theo Radiosity. Mô hình này không có lợi cho việc tính toán theo giải thuật Ray tracing vì với bề mặt được chia nhỏ thành nhiều miếng, việc tìm giao cắt của một tia với miếng đầu tiên đòi hỏi phải kiểm tra giao cắt với tất cả các miếng và tìm giao cắt với miếng gần nhất. Số lượng các miếng tăng lên cúng đồng nghĩa với việc làm giảm tốc độ của giải thuật. Tuy nhiên những hạn chế về tốc độ có thể khắc phục nhờ áp dụng một số giải thuật cải thiện nhỏ.

Từ những nhận xét trên, mô hình các vật thể có bề mặt là lưới các điểm được chọn. Cấu trúc cụ thể của các mô hình vật thể như sau:

- Mỗi *vật thể* bao gồm một tập hợp các *bề mặt*, với mỗi bề mặt được cho thông số phản xạ và phát xạ với các màu R, G, B.
- Mỗi bề mặt đó lại được chia nhỏ thành một lưới của một hay nhiều *miếng* tam giác hay tứ giác.
- Một miếng sau đó lại được chia nhỏ thành một hay nhiều *thành phần*

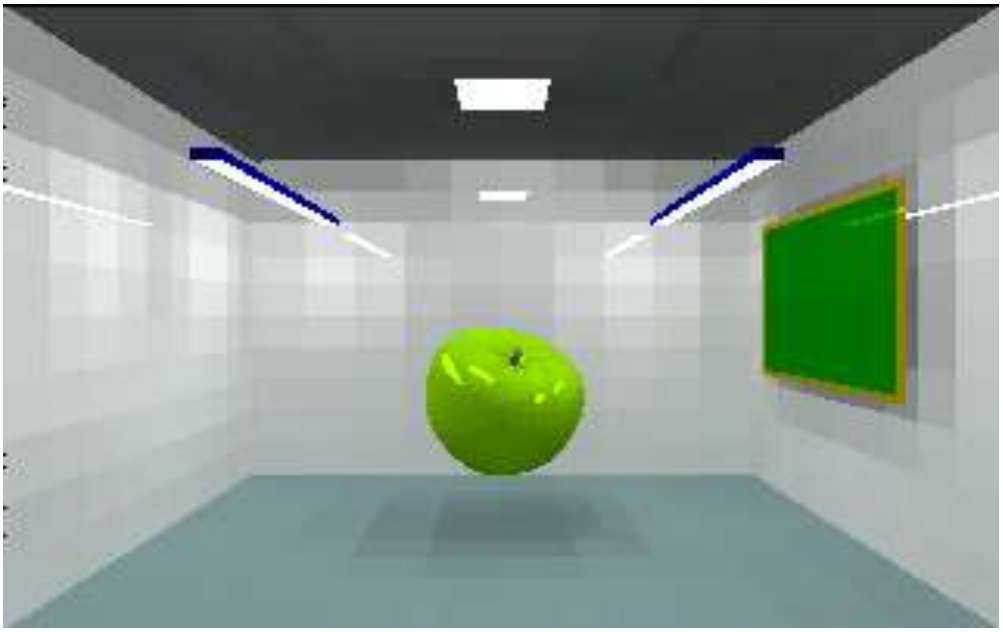
nhỏ hơn.

- Mỗi miếng và mỗi thành phần được định nghĩa bởi các đỉnh. Bởi vì hầu hết các đỉnh có thể thuộc nhiều miếng hay thành phần, do vậy các đỉnh được định nghĩa riêng. Các thành phần và miếng chỉ ra các đỉnh của mình thông qua chỉ số của các đỉnh trong dãy các đỉnh.

Khắc phục vấn đề gặp phải

Sau khi thực hiện Radiosity, các miếng của cùng bề mặt có độ sáng tối khác nhau giống với phân bố ánh sáng trong thực tế. Các miếng tại gần nguồn sáng sẽ có sáng

hơn, những miếng bị khuất nguồn sáng sẽ tối hơn. Như vậy bề mặt sẽ có hình kẻ sọc (xem hình 2.12), ảnh sẽ không đẹp. Để khắc phục vấn đề trên, thực hiện đánh bóng giữa các miếng. Chương trình thực hiện theo cả hai giải thuật đánh bóng nổi tiếng là Gouraud và Phong.



Hình 2.12: Bức ảnh khi chưa thực hiện đánh bóng

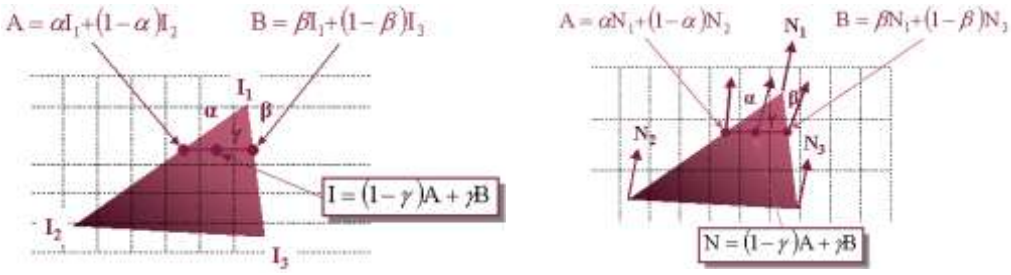
Triển khai giải thuật đánh bóng Gouraud

Triển khai giải thuật này bằng cách gán độ sáng tại mỗi đỉnh được tính bằng trung bình độ sáng của các miếng có chung đỉnh đó. Độ sáng của một điểm trong miếng được nội suy ra từ độ sáng của các đỉnh của miếng đó theo phương pháp nội suy nhị phân như trong hình (2.13) và phương trình (3) (4) (5).

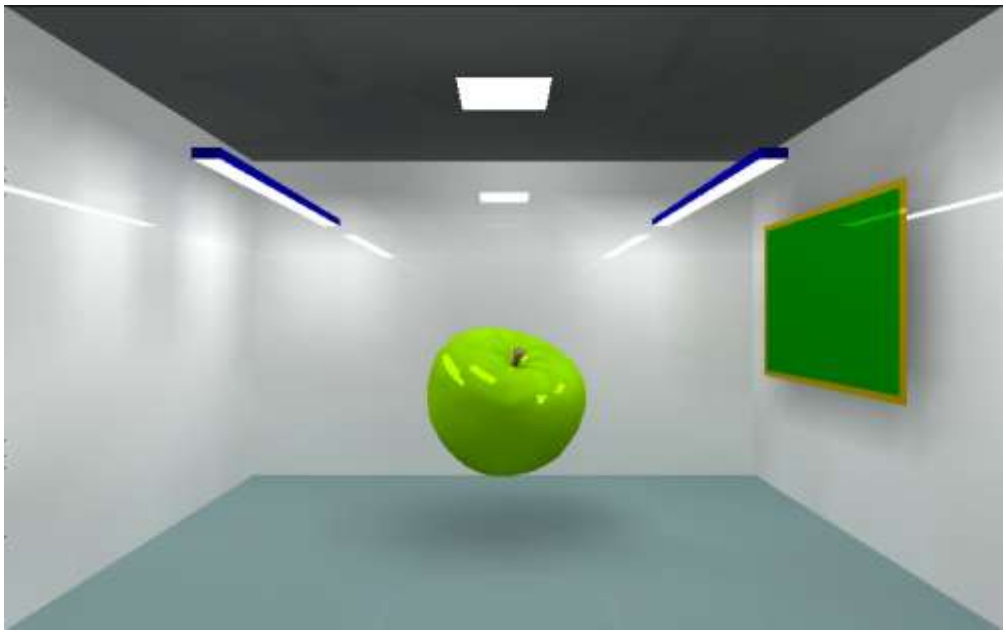
$$A = \alpha I_1 + (1-\alpha) I_2 \quad (3)$$

$$B = \beta I_i + (1-\beta) I_3 \quad (4)$$

$$I = (1-\gamma) A + \gamma B \quad (5)$$



Hình 2.13: Phương pháp đánh bóng Gouraud và đánh bóng Phong



Hình 2.14: Ảnh khi thực hiện đánh bóng theo phương pháp Gouraud

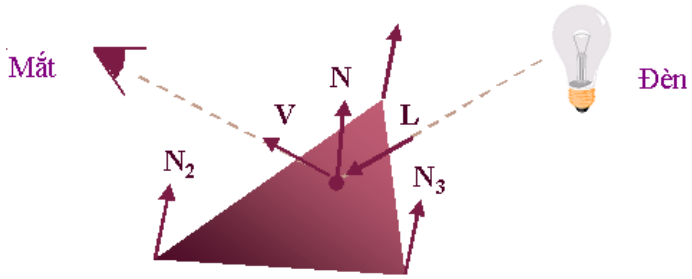
Triển khai giải thuật đánh bóng Phong

Với giải thuật này thì vector pháp tuyến tại mỗi đỉnh được tính bằng trung bình các vector pháp tuyến của các miếng có chung đỉnh đó. Vector pháp tuyến của một điểm bên trong một miếng được nội suy ra từ vector pháp tuyến của các đỉnh của miếng đó theo phương pháp nội suy nhị phân như trong hình (2.14) và phương trình (6) (7) (8). Độ sáng tại mỗi điểm bây giờ cón phụ thuộc vào góc giữa vector pháp tuyến tại điểm đó và vector từ điểm nhìn tới điểm sáng đó.

$$A = \alpha N_1 + (1-\alpha) N_2 \quad (6)$$

$$B = \beta N_i + (1-\beta) N_3 \quad (7)$$

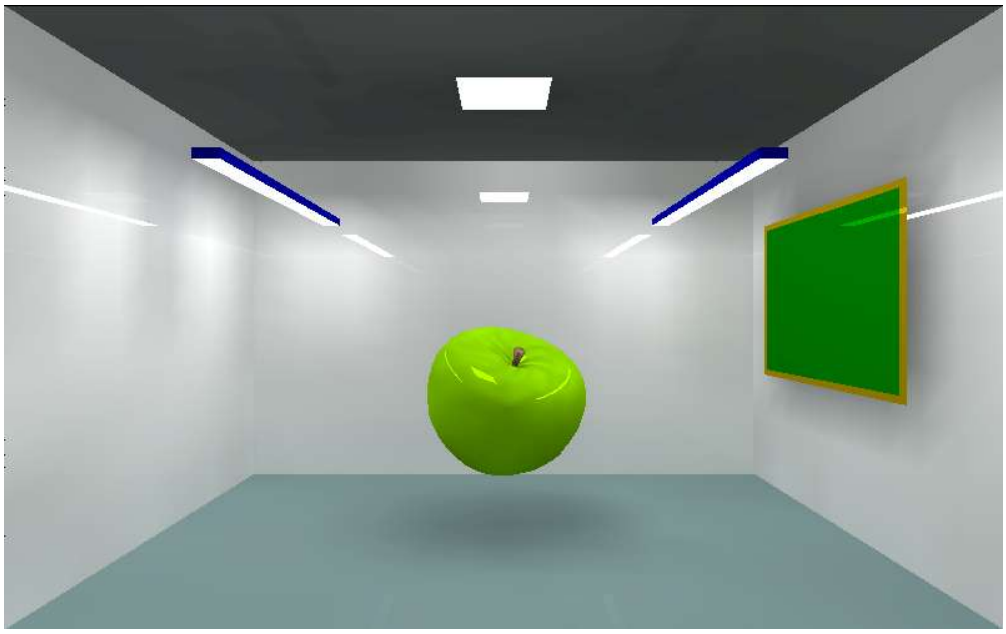
$$N = (1-\gamma) A + \gamma B \quad (8)$$



Hình 2.15: Độ sáng của một điểm phục thuộc vào

So sánh hình (2.14) và (2.16) ta có thể thấy phương pháp đánh bóng Phong làm cho ảnh có bóng phản xạ được định hướng chính xác hơn với những bề mặt cong, chú

ý bóng phản xạ trên quả táo của hai ảnh này để thấy rõ điều đó. Điều này có được là do độ sáng còn phụ thuộc vào vector pháp tuyến tại điểm đó cũng như vector phản xạ được tính chính xác hơn khi có được vector pháp tuyến chính xác tại điểm cắt.

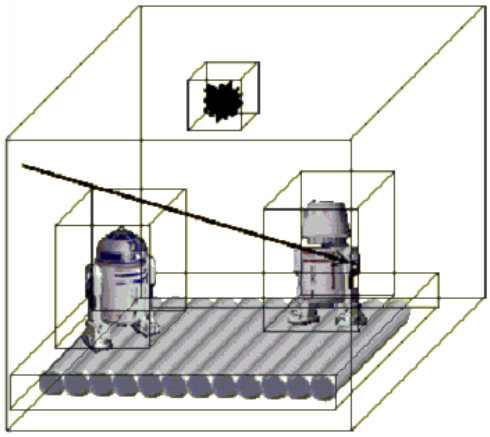


Hình 2.16: Ảnh thực hiện đánh bóng theo phương pháp Phong

Giải thuật cải thiện tốc độ

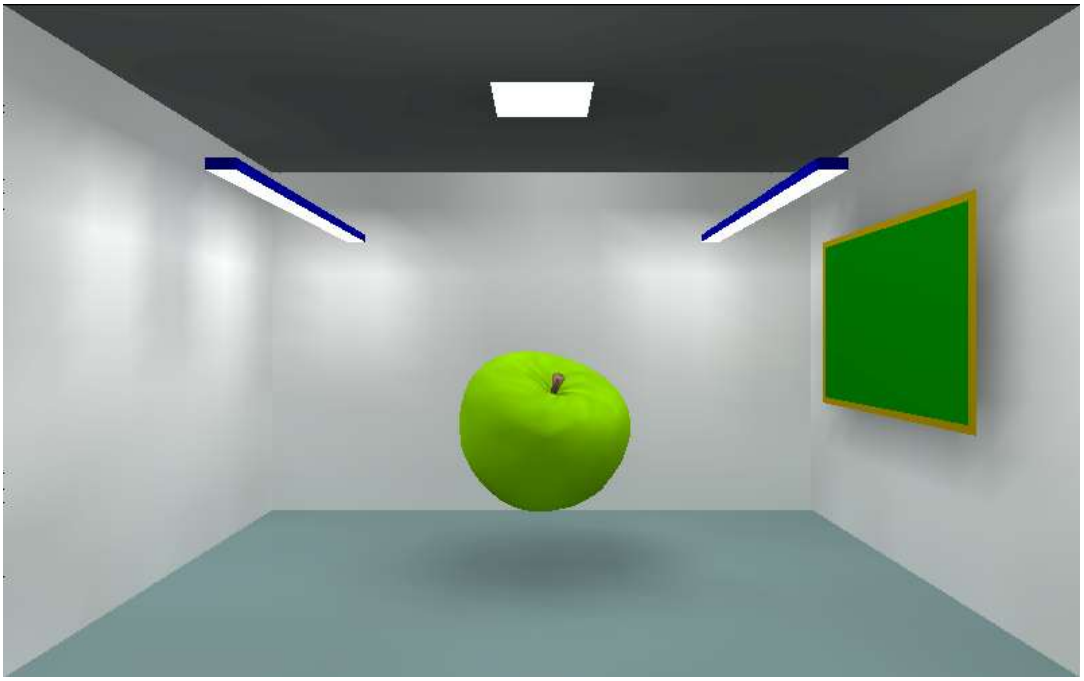
Vấn đề gặp phải khi triển khai giải thuật kết hợp này là với mô hình vật thể được chia thành nhiều miếng khi thực thi giải thuật Ray tracing là khi tìm giao cắt của một tia với vật thể đầu tiên phải kiểm tra giao cắt với tất cả các miếng để tìm điểm cắt đầu tiên. Nếu số lượng miếng tăng thì cũng đồng nghĩa với khối lượng tính toán tăng dẫn đến tốc độ dựng ảnh chậm.

Để cải thiện tốc độ, trước khi kiểm tra xem một tia có cắt một miếng nào của một vật thể không, ta kiểm tra xem tia đó có cắt hình hộp chữ nhật bao quanh vật đó không. Chỉ khi tia đó cắt một mặt nào đó của hình hộp bao quanh thì mới kiểm tra tìm giao cắt của tia đó với các miếng của vật thể đó.

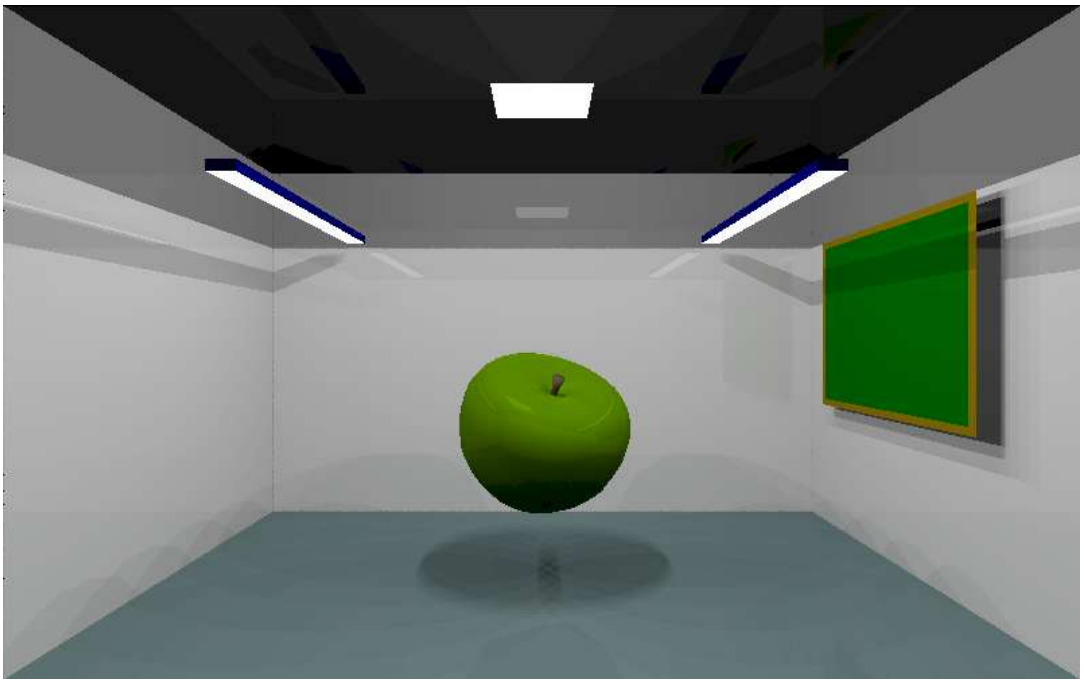


Hình 2.17: Kiểm tra giao cắt của một tia với hộp bao trước

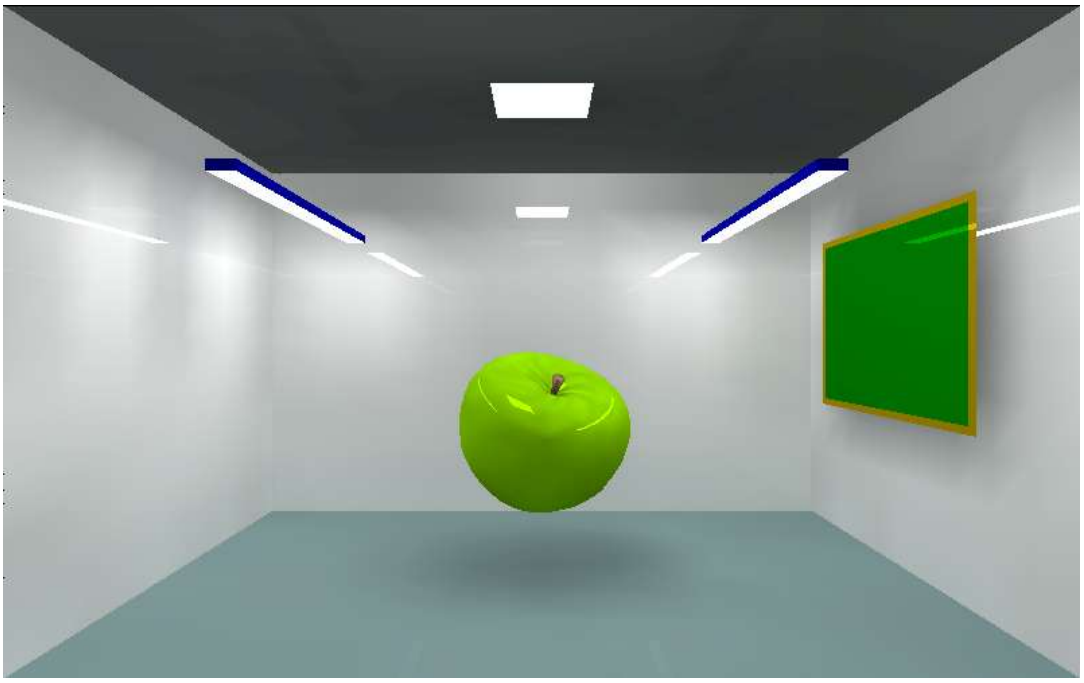
Hình ảnh đã được hoàn thành có thể dựng ảnh với giải thuật kết hợp cả hai giải thuật Radiosity và Ray tracing và cũng có thể dựng ảnh với hai giải thuật này một cách riêng rẽ.



Hình 2.18: Bức ảnh thực hiện với giải thuật Radiosity



Hình 2.19: Ảnh thực hiện với ký thuật Ray tracing



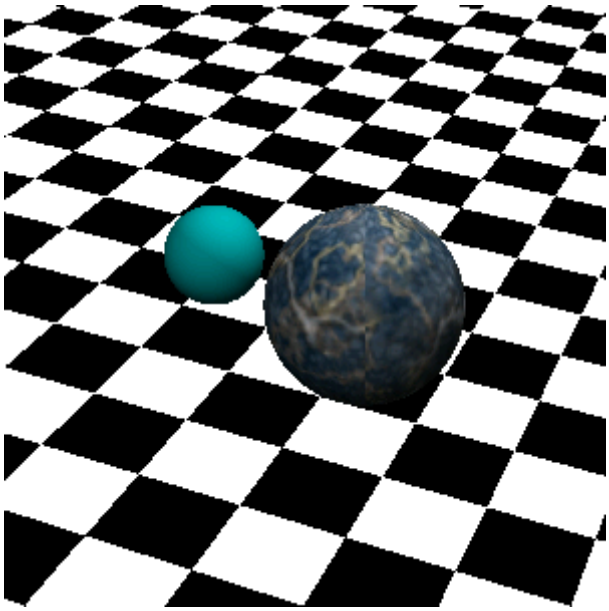
Hình 2.20: Bức ảnh khi kết hợp cả hai giải thuật

Qua các ảnh trên ta thấy ảnh sử dụng giải thuật kết hợp có hình ảnh rất gần với thực tế. Ảnh của giải thuật kết hợp có bóng mờ, độ sáng tối phân bố hợp lý và có cả bóng phản xạ trên các vật. Ảnh thực hiện chỉ với giải thuật Radiosity thiếu bóng phản xạ trên các vật còn ảnh thực hiện chỉ với giải thuật Ray tracing thì độ sáng tối phân bố không hợp lý. Bức ảnh kết hợp của cả hai giải thuật có màu sắc cũng như bóng phản xạ rất gần với thực tế và rất khó phân biệt với một bức ảnh thật.

Chương 3: CHƯƠNG TRÌNH THỬ NGHIỆM

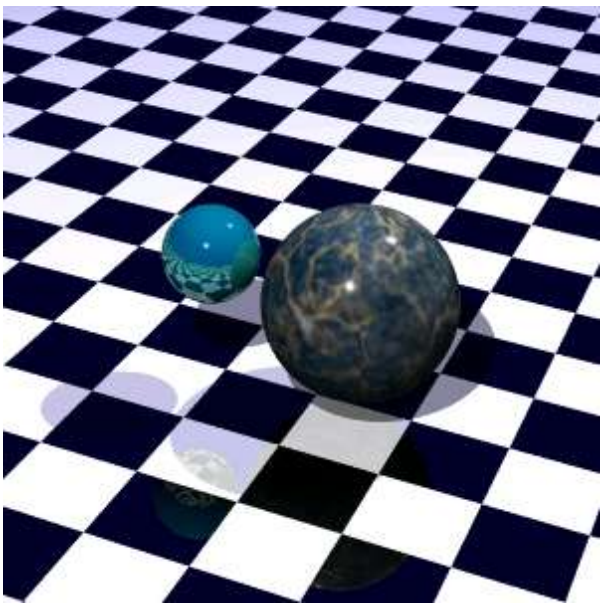
3.1. BÀI TOÁN

Các cảnh trong phép dò tia ray tracing được mô tả toán học bởi một lập trình viên hoặc một nghệ sĩ thể hiện trực quan (thường sử dụng những công cụ trung gian). Các cảnh cũng có thể kết hợp chặt chẽ các dữ liệu từ hình ảnh và mô hình ghi lại bởi các phương tiện ví dụ như máy ảnh kỹ thuật số.



Hình 3.1: Vật thể chưa qua xử lý

Mỗi tia phải được kiểm tra sự giao nhau với một vài tập con của toàn bộ các đối tượng trên cảnh. Một khi đối tượng gần nhất được xác định, giải thuật sẽ ước lượng ánh sáng tới tại giao điểm khảo sát, xem xét tính chất vật liệu của đối tượng, và tổng hợp thông tin để tính toán màu sắc cuối cùng (chuẩn) của điểm ảnh tương ứng. Các giải thuật tổng thể cụ thể và các vật liệu phản xạ, khúc xạ có thể đòi hỏi nhiều tia hơn để lấp đầy cảnh.

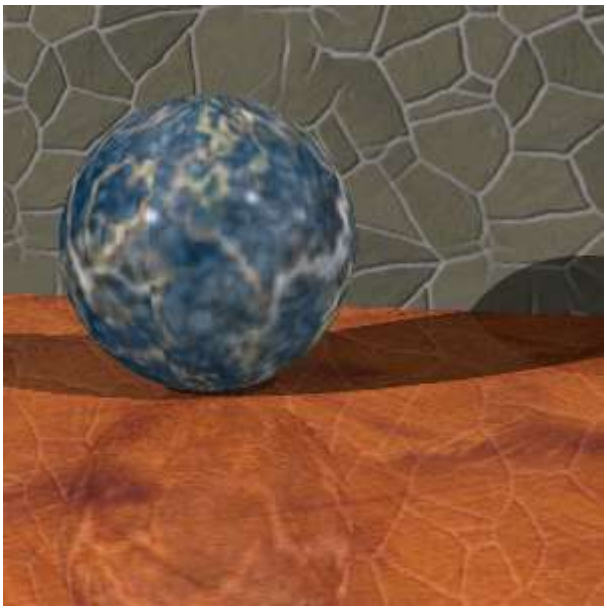


Hình 3.2: hình ảnh vật thể sau xử lý

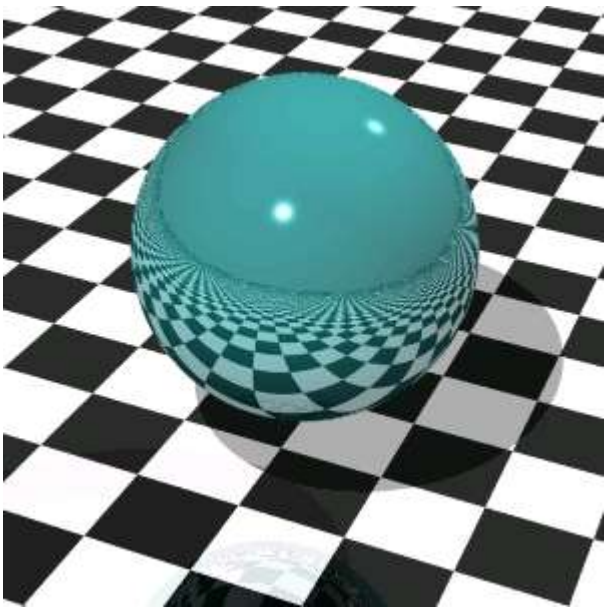
3.2. MỘT SỐ KẾT QUẢ CHƯƠNG TRÌNH

Chương trình của em được viết bằng C# 2.0

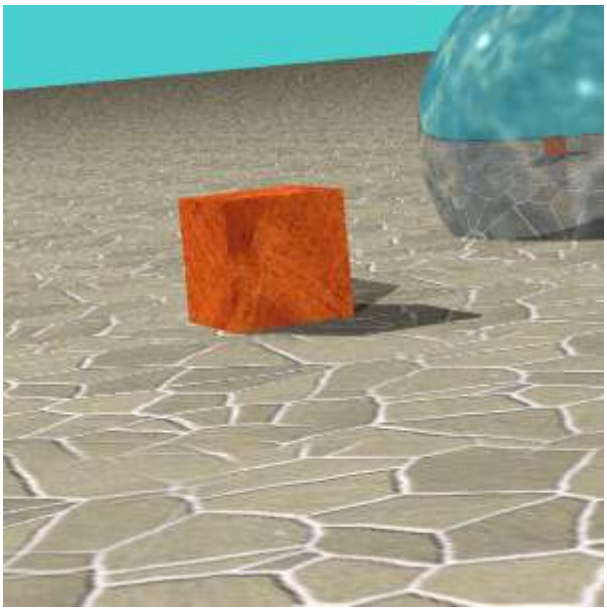
Một số hình ảnh sau khi thực hiện chương trình



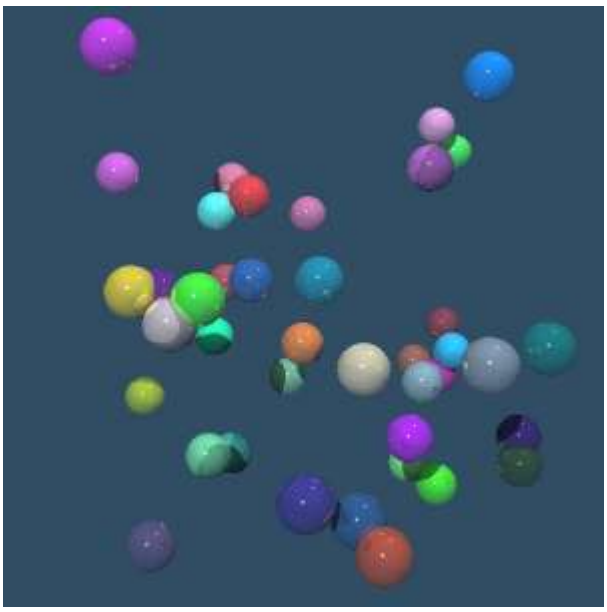
Hình 3.3: khối cầu trên nền gỗ



Hình 3.4: Quả cầu trên nền đá hoa



Hình 3.5: hộp gỗ và quả cầu pha lê



Hình 3.6: nhiều quả cầu trong không trung

KẾT LUẬN

Ít ai có thể phủ nhận vai trò của Ray Tracing trong đời sống công nghệ hiện nay, khi mà việc tạo ảnh và các hiệu ứng như bóng, trong suốt và hình phản chiếu đã trở nên rất phổ biến. Có rất nhiều lĩnh vực thiết kế đồ họa trong đó Ray Tracing đóng vai trò trợ giúp đắc lực làm đơn giản hoá những thiết kế như: ánh sáng, tạo bóng , khử răng cưa – tất cả đều có thể được lập trình bằng một cách đơn giản hơn với Ray Tracing.

Có những lĩnh vực đã áp dụng Ray Tracing và các nhà phát triển đã nhận ra lợi ích của công nghệ này. Trong đó lãnh vực rõ rệt nhất là phim ảnh – hết các hiệu ứng hình cảnh chất lượng cao của Hollywood đều được làm bằng Ray Tracing.

Một lĩnh vực khá cũng rất thú vị có sử dụng Ray Tracing chính là quá trình phát triển game. Các studio game vẫn thường dùng Ray Tracing để sản xuất dữ liệu thô để tạo hiệu ứng thị giác, như trong bản đồ ánh sáng chẳng hạn.

Chính vì vậy đề tài nghiên cứu của em đã tìm hiểu về ray tracing và các kỹ thuật sinh ảnh từ raytracing. Qua quá trình nghiên cứu đề tài, em đã hiểu thêm về các phương pháp xử lý ảnh và đã thực hiện một số phương pháp xử lý ảnh đơn giản. Mong rằng những kiến thức tiếp thu được trong quá trình làm đồ án có thể giúp ích cho công việc và nghiên cứu sau này.

TÀI LIỆU THAM KHẢO

[1] Lương Chi Mai, Huỳnh Thị Thanh Bình (2000), *Nhập môn đồ họa máy tính*, Nhà xuất bản KH&KT, 2000.

[2] Lê Tấn Hùng, Huỳnh Quyết Thắng (2004), *Kỹ thuật đồ họa*, Nhà xuất bản KH&KT, 2000.

[3] Peter Shirley, *Realistic Ray Tracing*, AK Peters Press,2000.

[4] Andrew V.Nealen, *"Shadow Volume and Shadow Mapping, Recent Development in Real-time Shadow Rendering"*, University of British Columbia , 2000.

[5] Henrik Van Jensen, *Realistic Image Synthesis using Photon mapping*, AK Peters Press,2001.

[6] Philip Dutré, *Global Illumination Compendium*, Cornell University,August, 2001.

[7] Dietrich, Sim, "*Shadow Techniques*", Game Developers Convention 2001,

[8] Philip Dutré, Kavita Bala, Philippe Bekaert, 'Advanced Global Illumination', *SIGGRAPH 2002 Course 2*, 2002.

[9] Mark Kilgard, "Shadow Mapping with Today's Hardware", Technical Presentation: http://developer.nvidia.com/view.asp?IO=cedec_shadowmap.

[10] Everitt, CassRege, Ashu Cebenoyan, Cem, "Hardware Shadow Mapping",

[11] http://nehe.gamedev.net

[12] http://paulprojects.net

[13] http://www.cs.kuleuven.ac.be/cwis/research/graphics/INFOTEC/viewing-in-3d

[14] http://www.povray.org/

[15] http://developer.nvidia.com/attach/1308,