BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG
-------o0o-------

ISO 9001:2008

TÌM HIỂU KỸ THUẬT TẠO BÓNG CỨNG SHADOW MAPPING

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ Thông tin

Sinh viên thực hiện:	Đào Đức Cương
Giáo viên hướng dẫn:	PGS.TS. Đỗ Năng Toàn
Mã số sinh viên:	1351020027

HẢI PHÒNG - 2013

LỜI CẢM ƠN

Em xin gửi lời cảm ơn tới các thầy cô khoa Công nghệ thông tin trường Đại học Dân Lập Hải Phòng, những người đã ân cần dạy dỗ cho chúng em những kiến thức bổ ích và quý giá trong suốt 4 năm học qua, những người đã trang bị cho chúng em hành trang quý giá để bước vào đời.

Em xin gửi lời cảm ơn sâu sắc tới thầy Đỗ Năng Toàn, người đã tận tình chỉ bảo và hướng dẫn chúng em thực hiện tốt đồ án tốt nghiệp này. Chúng em xin gửi lời cảm ơn tới gia đình và bạn bè, hậu phương vững chắc cho tiền tuyến chúng em trong suốt những năm học gian khổ, và gần đây đã cho chúng em nguồn động viên to lớn về tinh thần và vật chất để chúng em có thể hoàn thành tốt đồ án tốt nghiệp này.

MỤC LỤC

PHẦN MỞ ĐẦU 4

Chương 1: KHÁI QUÁT VỀ ĐỒ HỌA 3 CHIỀU ÁNH SÁNG VÀ BÀI TOÁN TẠO BÓNG 6

1.1. Khái quát về đồ họa 3 chiều 6

1.1.1. Giới thiệu 6

1.1.2. Biểu diễn điểm và các phép biến đổi 9

1.1.3. Phép biến đổi hiển thị (Viewing Transformation) 10

1.1.4. Phép chiếu trực giao (Orthographic Projection) 11

1.1.5. Phép chiếu phối cảnh (Perspective Projection) 13

1.1.6. Phép biến đổi cổng nhìn (Viewport Transformation) 14

1.2. Bộ đệm và các phép kiểm tra 15

1.2.1. Bộ đệm chiều sâu (Z-Buffer) 15

1.2.2. Bộ đệm khuôn (Stencil Buffer) 16

1.3. Tạo bóng và phân loại bóng 17

1.3.1. Khái niệm 17

1.3.2. Phân loại bóng 19

1.3.3. Các kỹ thuật tạo bóng cứng 20

1.3.4. Các kỹ thuật tạo bóng mềm 21

Chương 2: KỸ THUẬT TẠO BÓNG CỨNG SHADOW MAPPING VÀ CÁC NGUỒN SÁNG 23

2.1. Các loại nguồn sáng 23

2.1.1. Nguồn sáng xung quanh 23

2.1.2. Nguồn sáng định hướng 23

2.1.3. Nguồn sáng điểm 25

2.2.Ý tưởng chính 26

2.3. Thuật toán 27

2.4. Chuyển tọa độ 35

2.5. Nhận xét 36

Chương 3: CHƯƠNG TRÌNH THỬ NGHIỆM..... 37

3.1. Bài toán..... 37

3.2. Phân tích, thiết kế..... 37

3.2.1: Giới thiệu về ngôn ngữ lập trình…….....37

3.2.2: Chức năng của một số hàm trong chương trình.....38

3.3. Thực nghiệm chương trình và đánh giá kết quả..... 39

3.3.1: Thực nghiệm chương trình.....39

3.3.2: Kết quả thực hiện.....42

PHẦN KẾT LUẬN..... 45

TÀI LIỆU THAM KHẢO……..46

PHẦN MỞ ĐẦU

Trong thực tế, con người cảm nhận thế giới bằng các giác quan của mình. Một vật thể có thể được cảm nhận bằng các xúc giác qua sự sờ mó hay được cảm nhận bằng mùi qua khứu giác , tuy nhiên trong một chừng mực nào đó có thể nói cảm nhận vật thể đó bằng thị giác qua màu sắc, đặc điểm, hình dạng,… sẽ cho con người một cảm nhận đầy đủ, trực quan và rõ ràng nhất. Vì vậy nếu có thể xây dựng được các chương trình trên máy tính mô phỏng được các vật thể, hiện tượng trong thế giới thực thì sẽ cung cấp cho người dùng một cách tiếp cận bằng thị giác trực quan hơn về các vấn đề mà họ đang xem xét.

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.

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.

Với công nghệ phần cứng máy tính hiện nay, các hạn chế cơ bản về phần cứng của các chương trình đồ họa ba chiều phần nào đã được giải quyết, chính vì vậy các công nghệ về đồ họa ba chiều đang rất được quan tâm và phát triển trên thế giới. Các nhóm chương trình ứng dụng của đồ họa ba chiều có thể được kể ra như :

Hỗ trợ thiết kế : Một trong những ứng dụng của đồ họa ba chiều trên máy tính là các chương trình hỗ trợ thiết kế như CAD, 3D Max, Maya, Poser,… Các chương trình này được sử dụng cho các công việc như thiết kế nhà cửa, quần áo, phương tiện giao thông, các dụng cụ, các mô hình và cả con người,…

Giáo dục và đào tạo : Các chương trình mô phỏng (thực tại ảo) : mô phỏng sinh học, hóa học, vật lý học, mô phỏng phóng tàu vũ trụ, lái xe, lái máy bay, các bản đồ thông tin địa lý GIS…

Giải trí và nghệ thuật : Các chương trình thiết kế mỹ thuật, tạo mô hình cho việc quy hoạch,… cho phép tạo dựng và hiệu chỉnh kiến trúc của các công trình, cho phép quan sát ở nhiều góc độ để có một cái nhìn tổng quan về công trình từ đó đưa ra các chỉnh sửa phù hợp. Ngoài ra đồ họa ba chiều còn giúp tạo ra các chương trình trò chơi giải trí; hỗ trợ các kỹ xảo điện ảnh…

Vấn đề quan trọng của đồ họa ba chiều hiện nay là làm thế nào thể hiện được các hình ảnh của thế giới lên màn hình máy tính một cách trung thực nhất.

Xuất phát từ vấn đề này đồ án của em xây dựng gồm 3 chương:

CHƯƠNG 1:CÁC KIẾN THỨC CƠ BẢN ĐỒ HỌA 3D VÀ TẠO BÓNG

Chương này nói về các kiến thức cơ bản về hiển thị 3D, bộ đệm và các phép kiểm tra, về biểu diễn điểm và các phép biến đổi và khái quát các kỹ thuật tạo bóng.

CHƯƠNG 2: KỸ THUẬT TẠO BÓNG CỨNG SHADOW MAPPING

Chương này đi vào chi tiết về kỹ thuật tạo bóng cứng Shadow Mapping và các dạng nguồn sáng.

CHƯƠNG 3:CHƯƠNG TRÌNH THỰC NGHIỆM

Chương 1: KHÁI QUÁT VỀ ĐỒ HỌA 3 CHIỀU VÀ BÀI TOÁN TẠO BÓNG.

1.1. Khái quát về đồ họa 3 chiều.

1.1.1. Giới thiệu.

Hình ảnh được xuất hiện có chiều cao, chiều rộng và chiều sâu được gọi là 3 chiều – 3D(three-dimensional).

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 :

Bước thứ 1: 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:

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

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

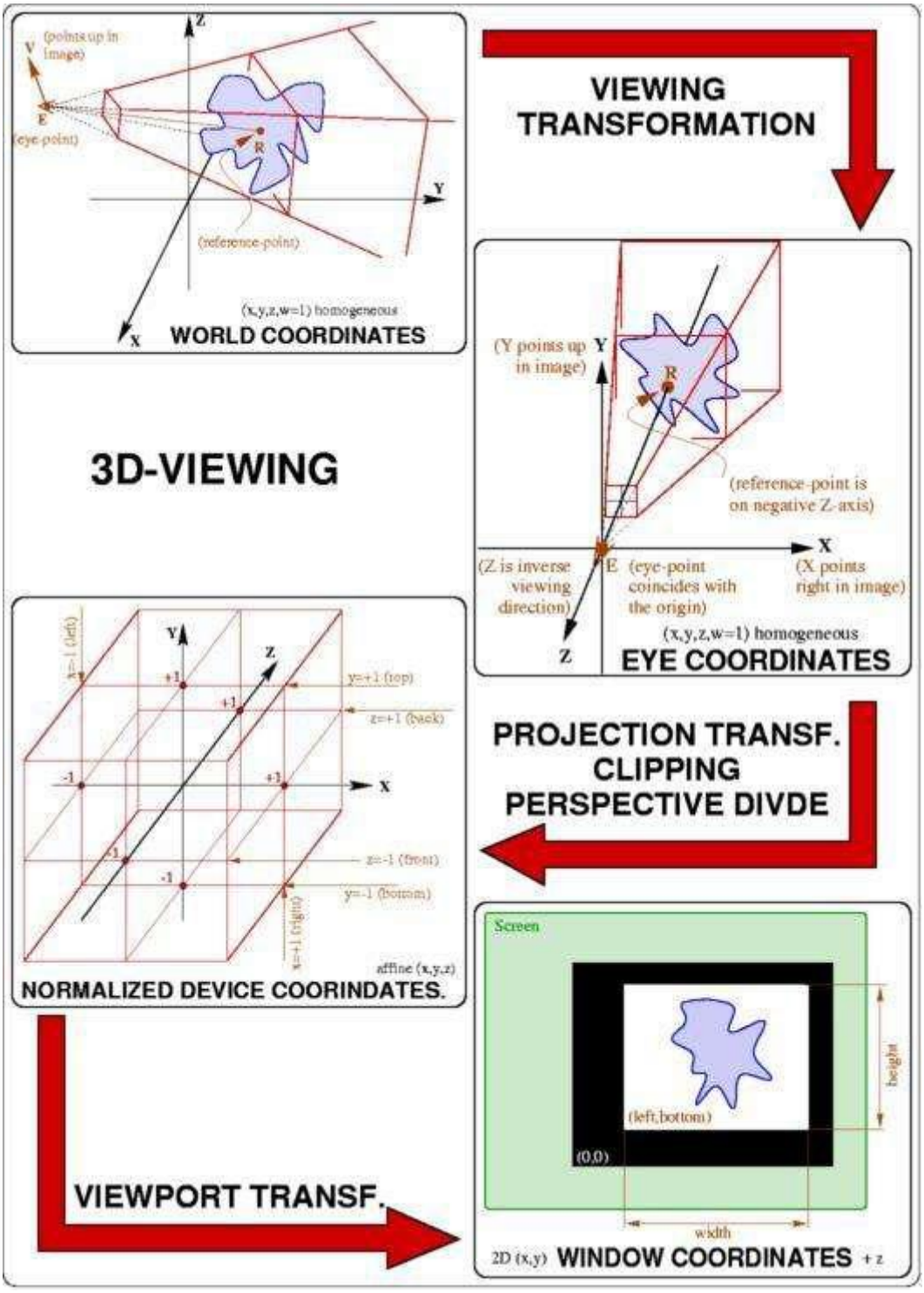
Bước này gồm 3 bước con:

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)

Bước thứa 3: 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.



Hình 1.1: Tổng quan về hiển thị 3D và các phép chiếu.

1.1.2. 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 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 4x4 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 $\{\vec{X}, \vec{Y}, \vec{Z}\}$.

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

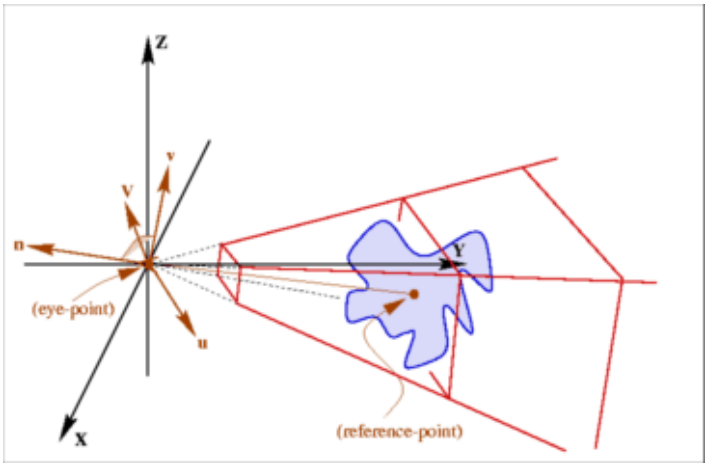
1.1.3. Phép biến đổi hiển thị (Viewing Transformation)

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. 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 sẵn 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 1.2: Phép biến đổi tịnh tiến

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 $\rightarrow \vec{Z}$ $(O\vec{z})$

$\vec{u} = \frac{\vec{V} \times \vec{n}}{\left\|\vec{V} \times \vec{n}\right\|}$ 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 vuống góc với $\vec{n}$ và $\vec{u} \rightarrow \vec{Y}$

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

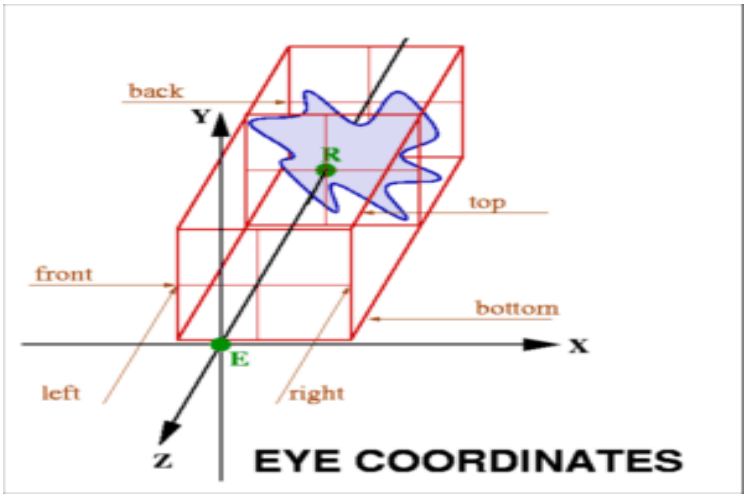
Và do đó ma trận của phép biến đổi sẽ là:

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

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

1.1.4. Phép chiếu trực giao (Orthographic Projection)

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 xleft, xright, ybottom, ytop, zfront và zback . (xleft, ybottom) và (xright, ytop) 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. Zfront và zback 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 zback≤ z ≤ zfront . 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 1.3: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 biển 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.

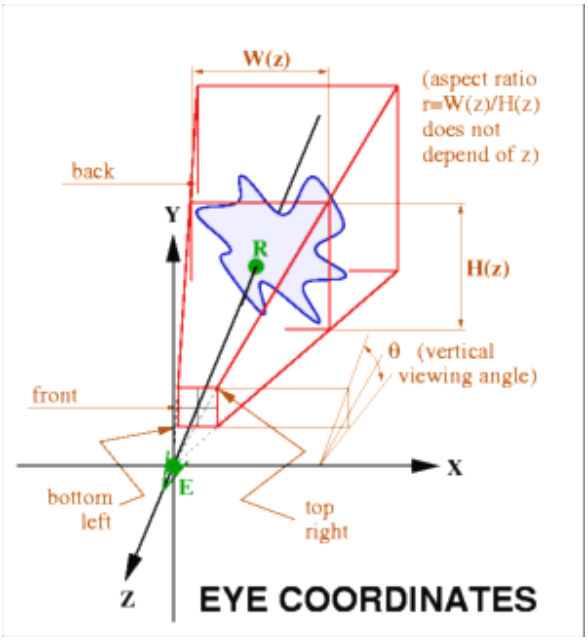
1.1.5. Phép chiếu phối cảnh (Perspective Projection)

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

1.1.6. Phép biến đổi cổng nhìn (Viewport Transformation)

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:

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

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

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

1.2. 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 đượcgọi là một *bitplane*. Có các bộ đệm phổ biến như Color Buffer, Depth Buffer, Stencil Buffer, Accumulation Buffer.

1.2.1. Bộ đệm chiều sâu (Z-Buffer).

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.

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).

1.2.2. Bộ đệm khuôn (Stencil Buffer).

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…

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);

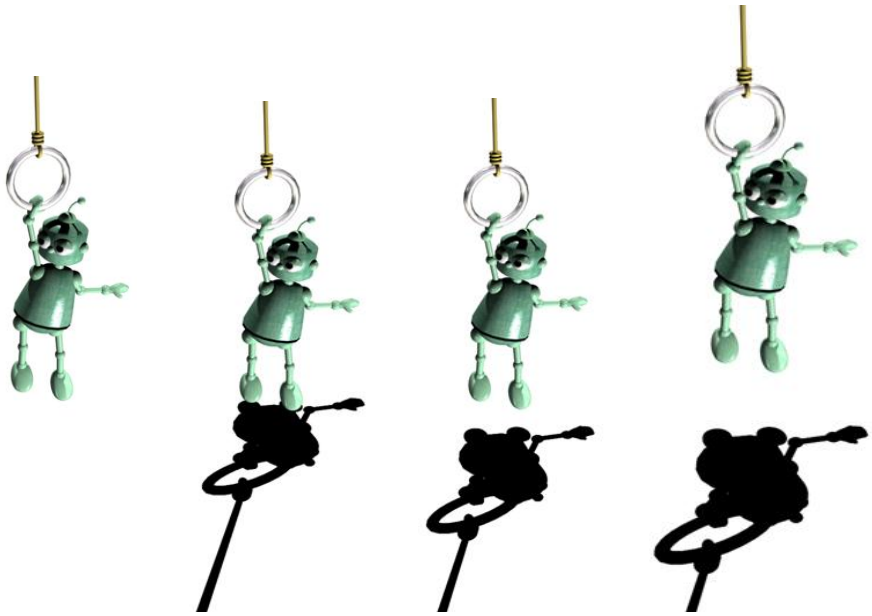
Hàm này xác định dữ liệu trong stencil Buffer sẽ thay đổi thế nào nếu như một "mảnh" pass hay fail phép kiểm tra stencil. 3 hàm fail, zfail và zpass có thể là GL_KEEP, GL_ZERO, GL_REPLACE, GL_INCR, GL_DECR …Chúng tương ứng với giữ nguyên giá trị hiện tại, thay thế nó với 0, thay thế nó bởi một giá trị 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.

1.3. Tạo bóng và phân loại bóng.

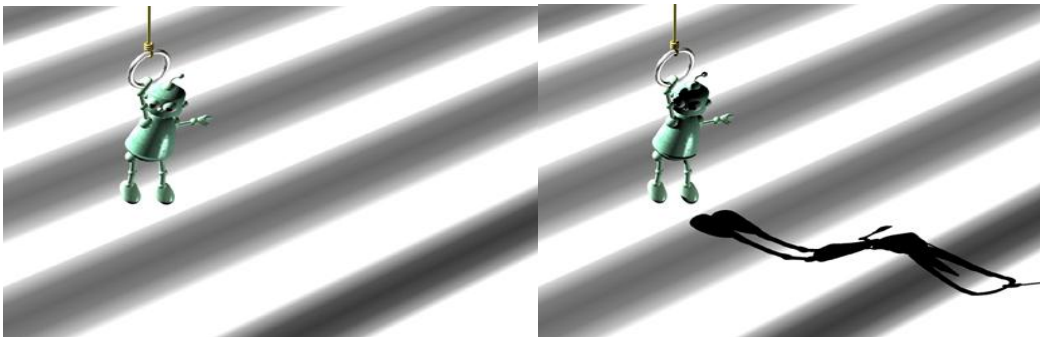
1.3.1. Khái niệm.

"Bóng (Shadow) là một vùng tối nằm giữa một vùng được chiếu sáng, xuất hiện khi một vật thể được chiếu sáng toàn bộ hoặc một phần"

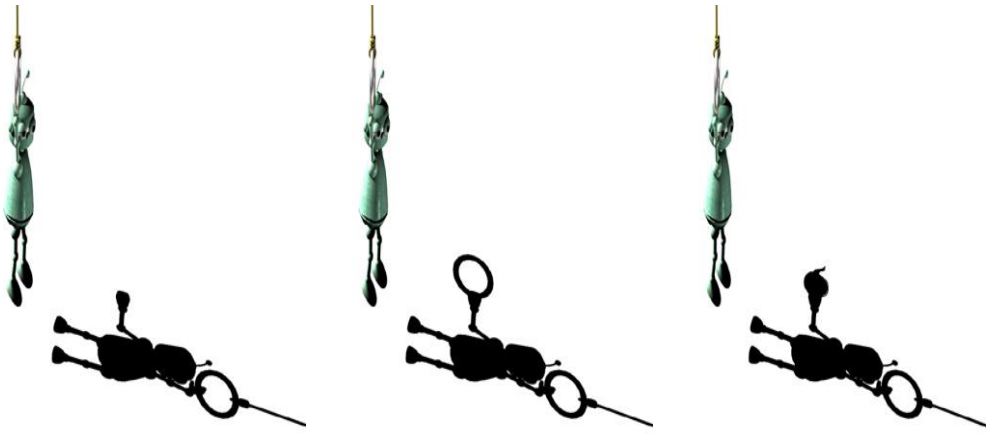
Bóng là một trong những yếu tố quan trọng nhất của tri giác con người về việc nhận biết các vật thể trong thế giới 3 chiều. Bóng giúp cho ta nhận biết được vị trí tương đối của vật đổ bóng (occluder) với mặt nhận bóng (receiver), nhận biết được kích thước và dạng hình học của cả vật đổ bóng và mặt nhận bóng.



Hình 1.5: Bóng cung cấp thông tin về vị trí tương đối của vật thể. Với ảnh ở bên trái ta không thể biết được vị trí của con rối. Nhưng với lần lượt 3 ảnh ở bên phải ta thấy vị khoảng cách của chúng so với mặt đất xa dần



Hình 1.6:Bóng cung cấp thông tin về dạng hình học của mặt tiếp nhận. Hình bên trái ta không thể biết được dạng hình học của mặt tiếp nhận, còn mặt bên phải thì dễ dàng thấy được.



Hình 1.7:Bóng cung cấp thông tin về dạng hình học của con rối. Hình bên trái con rối cầm đồ chơi, ở giữa nó cầm cái vòng, và bên phải nó cầm cái ấm trà

1.2.2. Phân loại bóng

Hầu hết các thuật toán và các phương pháp tạo bóng đều có thể được chia làm 2 loại chính là bóng cứng (Hard shadow) và bóng mềm (Soft shadow), phụ thuộc vào loại bóng mà nó tạo ra.

Vùng bóng được hiển thị được chia làm 2 phần phân biệt: Phần chính mà nằm hoàn toàn trong bóng được gọi là vùng thuần bóng, vùng bao bên ngoài nó và có một phần nằm trong bóng được gọi là vùng nửa bóng. Các thuật toán tạo bóng cứng là nhị phần vi mọi thứ đều chỉ có 2 trạng thái là bóng(1) và được chiếu sáng (0) – Chúng chỉ hiển thị duy nhất phần bóng của bóng. Các thuật toán tạo bóng mềm hiển thị vùng nửa bóng bên ngoài bao trùm vùng thuần bóng trung tâm và phải xử lý tính toán phần mờ đục cho vùng nửa bóng.(Kết quả từ sự phân bố cường độ ánh sáng bất quy tắc trong vùng nửa bóng)

Hình 1.8:Hình bên trái là một ví dụ về bóng cứng, hình bên phải là ví dụ về bóng mềm.

1.2.3. Các kỹ thuật tạo bóng cứng

Tạo bóng giả (Fakes Shadow)

Các thuật toán tạo bóng giả bao gồm các trường hợp đặc biệt tạo bóng không đúng đắn bằng các phương pháp toán học . Nhờ những kỹ thuật này chỉ được sử dụng trong những trường hợp đặc biệt (Ví dụ như bóng chỉ được vẽ cho những đối tượng đặc biệt , hoặc bóng chỉ được vẽ lên một mặt phẳng . Tuy nhiên các phương pháp này cũng tạo ra bóng làm cho ta có cảm giác khá thật.

Bóng khối (Shadow Volume)

Bóng khối là một kỹ thuật tạo bóng cần đến cấu trúc hình học của vật đổ bóng. Vật đổ bóng phải được tạo bởi các khối đa giác . Theo đó ta sẽtìm những đỉnh và cạnh viền , là những cạnh đóng vai trò tạo nên bó ng khối. Một tia sáng chiếu tới vật thể sẽ tiếp xúc với vật thể tại điểm hoặc cạnh viền đó và đi cắt mặt phẳng nhận bóng . Những cạnh viền , và đỉnh viền này sẽ tạo ra các mặt bên đa giác của bóng khối . Từ đó dựa vào các phép kiểm tra ta sẽ kiểm tra được một điểm trong khung cảnh có thuộc bóng khối hay không .

Dùng bản đồ bóng (Shadow Mapping)

Đây là thuật toán dùng đến bộ đệm chiều sâu (Depth Buffer). Ý tưởng chủ yếu là sử dụng bản đồ chiều sâu (hay còn gọi là bản đồ bóng) để lưu trữ các giá trị chiều sâu khi tạo ảnh từ vị trí của ánh sáng rồi sau đó sử dụng các giá trị này để xác định pixel nào được chiếu sáng hay là nằm trong bóng.

Lần theo tia sáng (Ray Tracing)

Thuật toán này sử dụng kỹ thuật Ray Tracing:

Với mỗi tia sáng đi ra từ mắt ta vào một không gian là một đường thẳng sẽ cắt vào cửa sổ (màn hình) và chạm vào vật thể trong không gian (gần nhất từ mắt). Tại điểm chạm vào vật thể đó thì tuỳ ở mỗi điểm chạm của vật thể đó có tính chất như thế nào mà ta chia ra các tia sáng tiếp theo

Nếu điểm chạm đó có tính khúc xạ, phản xạ thì ta lần theo tia sáng đó theo từng tia phản xạ, khúc xạ…

Nếu tại điểm chạm đó vật thể có tính xuyến thấu, phản xạ tức là 1 phần của tia sáng đi qua vật thể đó, một phần tia sáng đó được phản xạ ta lại xét từng tia....tiếp tục mỗi tia lại chạm vào vật thể khác lại chia ra từng tia khúc xạ phản xạ riêng ở mỗi điểm chạm .

Sau khi cắt mọi vật thể có thể trong không gian ta tính màu tại tia từ mắt cắt ở cửa sổ và đặt ở đó 1 giá trị màu. Tương ứng quét tất cả các tia từ mắt đến màn hình...

Bóng tạo bởi kỹ thuật này trông rất thật. Nhưng chi phí để thực hiện nó quá đắt vì phải thực hiện quá nhiều phép tính. Chính vì vậy kỹ thuật này ít được sử dụng trong các ứng dụng thời gian thực.

1.2.4. Các kỹ thuật tạo bóng mềm.

Các kỹ thuật tạo bóng mềm sẽ cho bóng sinh ra trông thật hơn rất nhiều so với bóng được sinh ra bởi các thuật toán tạo bóng cứng. Tính thật của nó được biểu hiện bởi cả vùng nửa bóng và vùng thuần bóng. Hình dạng của bóng sinh ra bởi các thuật toán tạo bóng mềm sẽ phụ thuộc vào hình dạng, kích thước của vật thể tạo bóng , nguồn sáng và cả vị trí tương đối giữa nguồn sáng và vật thể.

Các kỹ thuật tạo bóng mềm phổ biến có thể kể đến là:

Thuật toán bộ đệm khung (Frame Buffer Algorithms)

Được đề xuất bởi Brotman và Badler dựa trên việc sinh ra các đa giác thuần bóng trong suốt quá trình tiền xử lý. Bộ đệm chiều sâu 2D mà được sử dụng để xác định mặt được hiển thị sẽ được mở rộng để lưu bộ đếm nắm giữ các thông tin để xác định xem một pixel bất kỳ là nằm trong vùng nửa bóng hay vùng thuần bóng.

Dõi quang tia 2 chiều và phân bố (Distributed and Bidirectional Ray Tracing)

Rất nhiều mở rộng của thuật toán Ray-Tracing được sử dụng để tạo bóng mềm. Dõi quang tia phân bố cung cấp một kỹ thuật tạo bóng láng, mờ và chuyển động mờ trong khi Dõi quang tia 2 chiều cung cấp một phương pháp tạo bóng mềm rất nhanh.

Ánh sáng nâng cao (Radiosity)

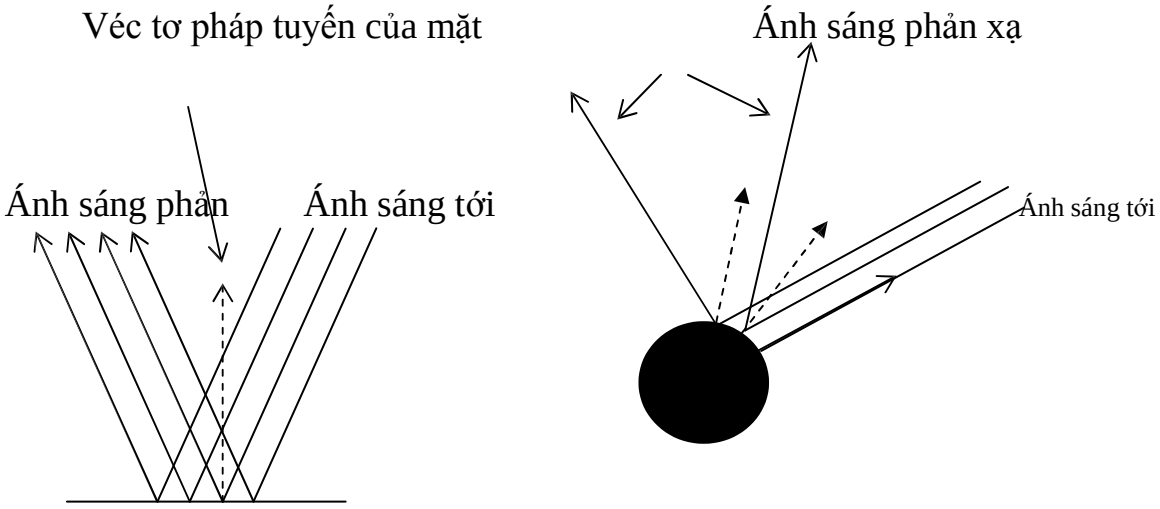
Radiosity là một kỹ thuật tạo bóng mềm bằng cách tính toán tất cả các phản xạ, khuếch tán ánh sáng giữa các mặt khác nhau của tất cả các vật thể trong khung cảnh. Nó hầu như chỉ được sử dụng cho các mặt đa giác bởi vì chi phí tính toán của phương pháp này rất lớn.

CHƯƠNG 2: KỸ THUẬT TẠO BÓNG CỨNG SHADOW MAPPING VÀ CÁC LOẠI NGUỒN SÁNG.

2.1. Các loại nguồn sáng.

2.1.1. Nguồn sáng xung quanh.

Ánh sáng xung quanh là mức sáng trung bình, tồn tại trong một vùng không gian. Một không gian lý tưởng là không 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

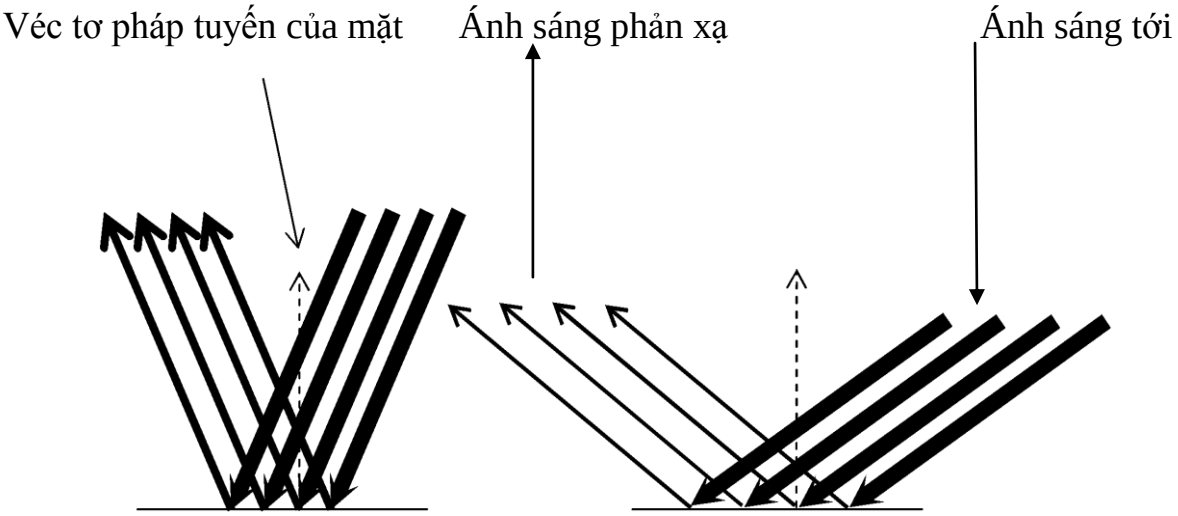


Hình 2.1: Sự phản xạ ánh sáng

2.1.2. 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 chiếu đến bề mặt là: khuyếch tán và phản chiếu. Nếu bề mặt phản xạ toàn bộ (giống như mặt 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 nhám, xù xì) thì một phần các tia sáng sẽ bị toả 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 (vì chỉ có những đối tượng được cấu tạo từ những mặt như mặt gương mới xảy ra hiện tượng phản xạ toàn phần), và đồng thời tìm cách tính cường độ của ánh sáng phản xạ trên bề mặt.



Hình 2.2: Sự phản xạ không toàn phần của ánh sáng

Trong hình 2.2 thể hiện sự phản xạ ánh sáng không toàn phần. Độ đậm nét của các tia ánh sáng tới thể hiện 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 (hay tạm gọi là tia phản xạ) luôn bé hơn so với cường độ của ánh sáng tới (hay gọi là tia 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, nếu góc này lớn thì cường độ phản xạ rất thấp. Ở đây ta chỉ quan tâm đến thành phần ánh sáng khuyếch tán và tạm bỏ qua hiện tượng phản xạ toàn phần. Để cho tiện trong việc tính toán ta tạm đổi hướng của tia tới thực sự, vậy bây giờ hướng của tia tới được xem là hướng ngược lại của tia sáng tới.

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:

$$\text{Cos}(\theta)=\frac{\vec{a}.\vec{n}}{|\vec{a}|.|\vec{n}|} \quad (2.1)$$

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 hoá 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 ánh sáng phản xạ =Cường độ của ánh sáng định hướng * [(Cos(θ)+1)/2]

Trong đó [(Cos(θ)+1)/2] có giá trị trong khoảng từ 0 đến 1. Vậy qua công 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.

2.1.3. 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 toả ra theo mọi hướng theo dạng 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 toạ độ là (x0, y0, z0) và nguồn sáng điểm có tọa độ là (plx, ply, plz) thì ánh sáng sẽ rọi đến điểm đang sét theo vector (x0- plx, y0-ply, z0-plz), hay tia tới: a = (plx - x0, ply - y0,plz - z0).

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

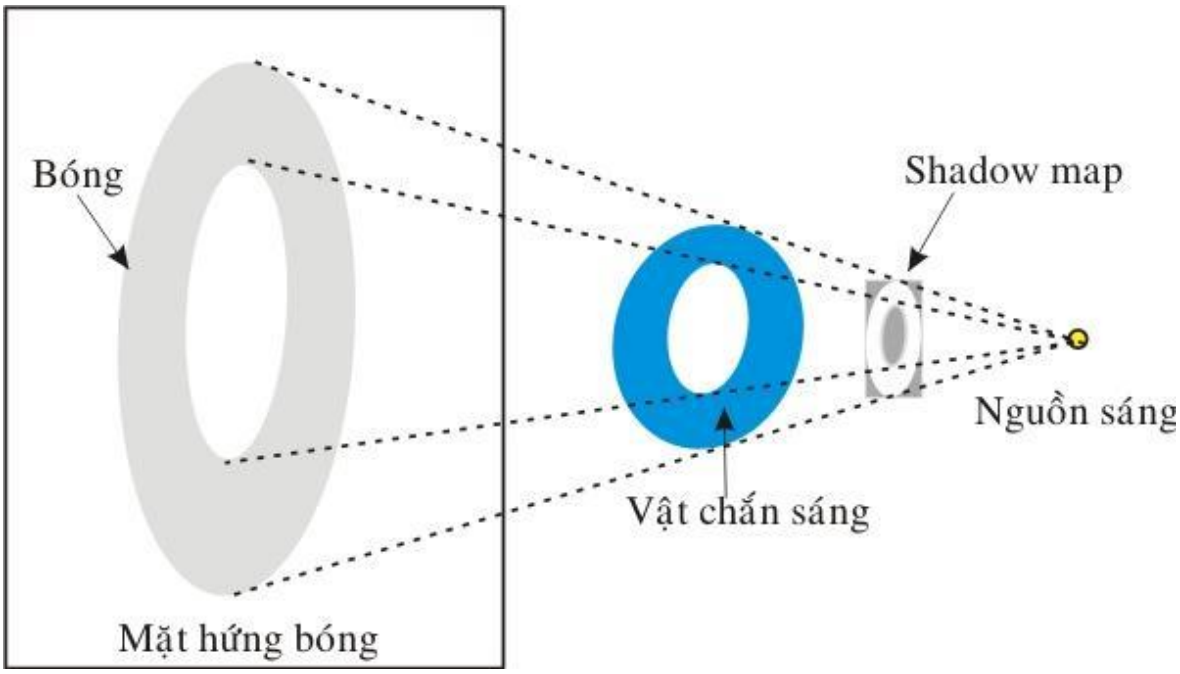
Vậy với nguồn sáng định hướng, chúng ta cần tính tia tới cho mọi điểm trên mặt, từ đó kết hợp với vector pháp tuyến của mặt để tính được cường độ sáng tại điểm đó, nếu tính toán trực tiếp thì có thể mất khá nhiều thời gian do phải tính vector a và tính Cos(θ) thông qua công thức (2.1) với tất cả các điểm trên mặt. Nên nhớ rằng trong tình hướng nguồn sáng điểm thì chúng ta buộc lòng phải tính Cos(θ) thông qua công thức (2.1) vì vector a sẽ thay đổi khi mặt hay nguồn sáng thay đổi (trừ khi mặt tĩnh, song nếu mặt tĩnh và nguồn sáng cố định thì suy ra chúng ta chỉ cần tính cường độ sáng một lần).

2.2. Ý tưởng chính.

Shadow Mapping được giới thiệu đầu tiên bởi Lance Williams năm 1978. Từ đó nó được sử dụng rất rộng rãi cả trong tạo ảnh offline lẫn trong các ứng dụng thời gian thực. Shadow mapping được sử dụng bởi Pixar’s RenderMan và là kỹ thuật tạo bóng chính thức sử dụng trong các bộ phim lớn như “Toy Story”.

Shadow mapping là kỹ thuật tạo bóng trên không gian ảnh (imagespace algorithm) nên ta những thông tin về hình dạng của vật thể hay những kiến thức hình học là không cần thiết. Thuật toán này cũng giống như thuật toán bóng khối, thực hiện phép kiểm tra “trong bóng” (in shadow) trên từng pixel. Một pixel được chiếu sáng nếu không có vật nào chắn giữa đường nối nó và nguồn sáng. Chìa khóa để hiểu được thuật toán Shadow Mapping (bản đồ bóng) là những điểm nằm trên bản đồ bóng chính là những điểm sẽ được hiển thị ra màn hình nếu như điểm nhìn đặt ở vị trí của ánh sáng.

Chúng ta có một thuật toán để xét xem điểm nào sẽ được hiển thị với người quan sát. Đó là sử dụng Z-buffer. Vì thế những điểm có depth test pass nếu chúng ta vẽ toàn bộ khung cảnh từ vị trí của nguồn sáng thì đó chính là những điểm không nằm trong bóng.



Hình 2.3: Chiếu Shadow Map

2.3. Thuật toán.

Thuật toán sẽ gồm 2 bước chính:

- Đầu tiên ta sẽ vẽ toàn bộ khung cảnh (chưa có bóng) từ vị trí của ánh sáng ra màn hình.

Lưu giá trị độ sâu trong Z-buffer vào trong bản đồ độ sâu. Các giá trị độ sâu này thực chất là giá trị z của những điểm có depth test pass và được hiển thị trên màn hình.

Bản đồ độ sâu này sẽ được dùng trong bước 2.

+ Code: Tính toán chiều sâu của điểm ảnh.

```
struct PixelInputType
{
    float4 position : SV_POSITION;
    float4 depthPosition : TEXTURE0;
};
```

```
////////////////////////////////////////////////////////////////////////////////
// Vertex Shader
////////////////////////////////////////////////////////////////////////////////
PixelInputType DepthVertexShader(VertexInputType input)
{
    PixelInputType output;


// Thay đổi vector vị trí là 4 đơn vị để tính toán ma trận thích hợp.


    input.position.w = 1.0f;


// Tính toán vị trí của các đỉnh so với tọa độ thế giới, xem và chiếu.
    output.position = mul(input.position, worldMatrix);

    output.position = mul(output.position, viewMatrix);

    output.position = mul(output.position, projectionMatrix);

// / Lưu trữ các giá trị vị trí trong một giá trị đầu vào thứ hai để tính toán giá trị chiều sâu.

    output.depthPosition = output.position;

    return output;

}
```

```
////////////////////////////////////////////////////////////////////////////////
// Filename: depth.ps
////////////////////////////////////////////////////////////////////////////////
// TYPEDEFS //
struct PixelInputType
{
    float4 position : SV_POSITION;
    float4 depthPosition : TEXTURE0;
};
////////////////////////////////////////////////////////////////////////////////
// Pixel Shader
////////////////////////////////////////////////////////////////////////////////
float4 DepthPixelShader(PixelInputType input) : SV_TARGET
{
        float depthValue;
        float4 color;
// Lấy giá trị chiều sâu của điểm ảnh.
depthValue = input.depthPosition.z / input.depthPosition.w;
```

- Sau đó chúng ta lại vẽ toàn bộ khung cảnh từ vị trí của điểm nhìn(camera).

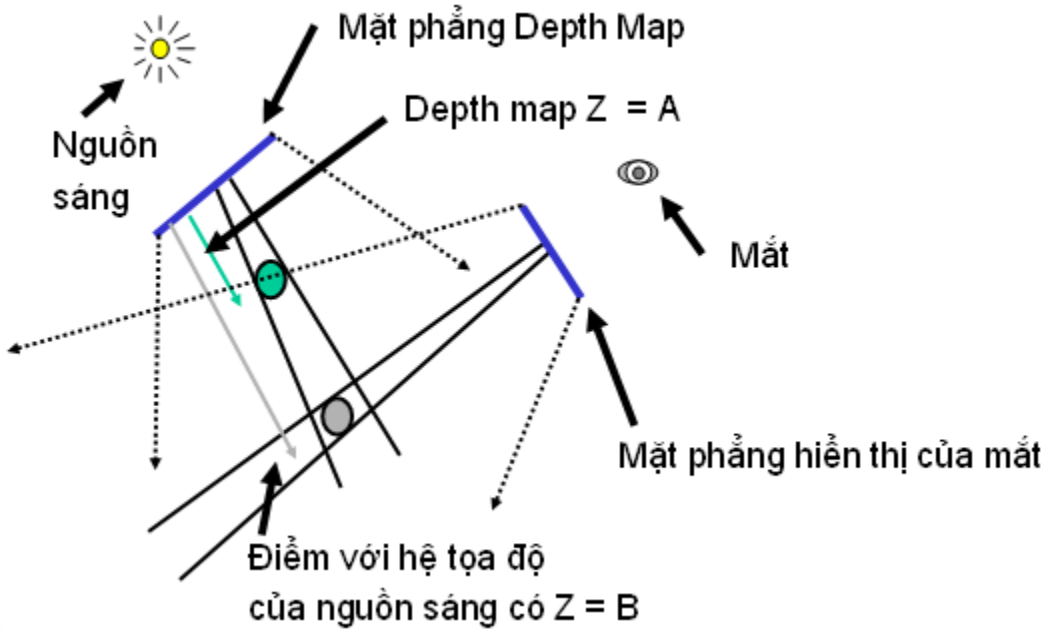
Với mỗi điểm ta sẽ chuyển tọa độ của chúng sang hệ tọa độ của ánh sáng.

Tọa độ này chính là tọa độ để xác định giá trị độ sâu khi thực hiện depth test. Giả sử điểm này khi chuyển sang sẽ có tọa độ là (x,y,z').

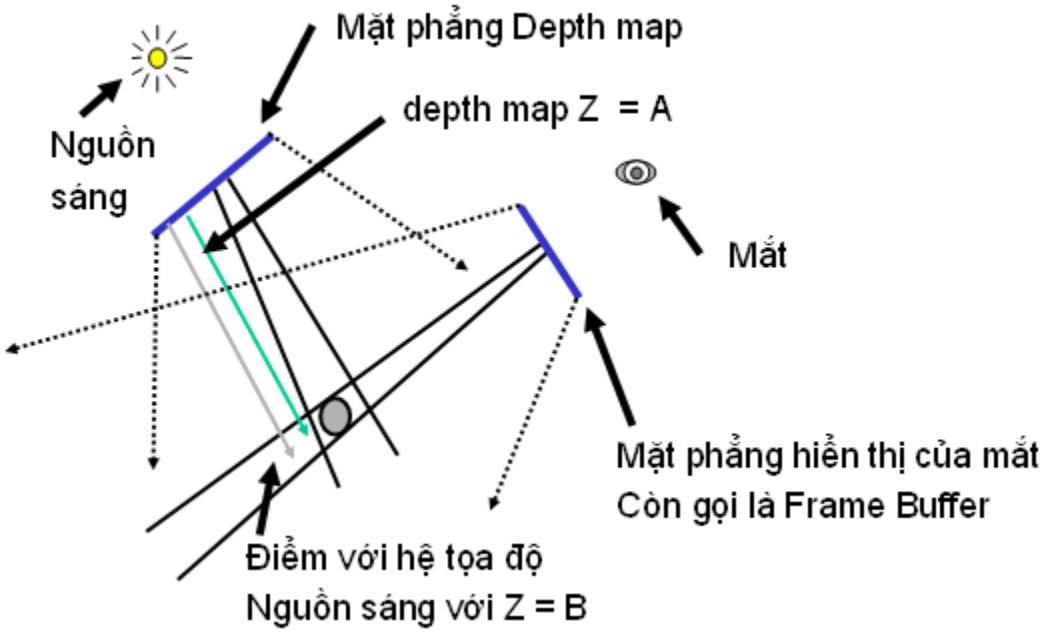
Sau đó công việc còn lại là so sánh giá trị độ sâu z của điểm có tọa độ (x,y) trong bản đồ độ sâu với giá trị z' của điểm (x,y,z') đó.

- Nếu z < z' thì điểm này nằm trong bóng và sẽ không được vẽ ra màn hình khi thực hiện vẽ khung cảnh từ vị trí của điểm nhìn.

- Nếu z >=z' thì điểm đó được chiếu sáng và sẽ được vẽ ra màn hình từ vị trí của điểm nhìn.



Hình 2.4: A < B điểm nằm trong bóng



Hình 2.5: A >= B điểm được chiếu sáng

+Code: chuyển tọa độ sang hệ tọa độ của ánh sáng.

```
// Tính toán vị trí của các đỉnh  xem bởi nguồn sáng.
    output.lightViewPosition = mul(input.position, worldMatrix);

    output.lightViewPosition=mul(output.lightViewPosition,
lightViewMatrix);

    output.lightViewPosition=mul(output.lightViewPosition,
lightProjectionMatrix);

    // Lưu trữ kết cấu tọa độ cho các pixel shader.

    output.tex = input.tex;

      // Tính toán vector bình thường so với tọa độ thế giới.

    output.normal = mul(input.normal, (float3x3)worldMatrix);

    // Chuẩn hóa các vector bình thường.

    output.normal = normalize(output.normal);
```

// Tính toán vị trí của các đỉnh theo tọa độ thế giới.

worldPosition = mul(input.position, worldMatrix);

// Xác định vị trí ánh sáng dựa trên vị trí của ánh sáng và vị trí của các đỉnh theo tọa độ thế giới.

output.lightPos = lightPosition.xyz - worldPosition.xyz;

// Chuẩn hóa các vector tho vị trí ánh sáng.

output.lightPos = normalize(output.lightPos);

return output;

}

+So sánh độ sâu của bóng và chiều sâu của ánh sáng để xác định xem có bóng hoặc ánh sáng ở điểm ảnh.

//Nếu ánh sáng ở phía trước đối tượng thì các điểm ảnh được chiếu sáng.Nếu không điểm ảnh này được đổ bóng bởi một đối tượng đang tạo bóng trên đó.

if (lightDepthValue <depthValue)

{

// Nếu ánh sáng ở phía trước đối tượng và không có đổ bóng thì chúng ta làm chiếu sáng thông thường.

// Tính toán lượng ánh sáng vào điểm ảnh này.

lightIntensity = saturate(dot(input.normal, input.lightPos));

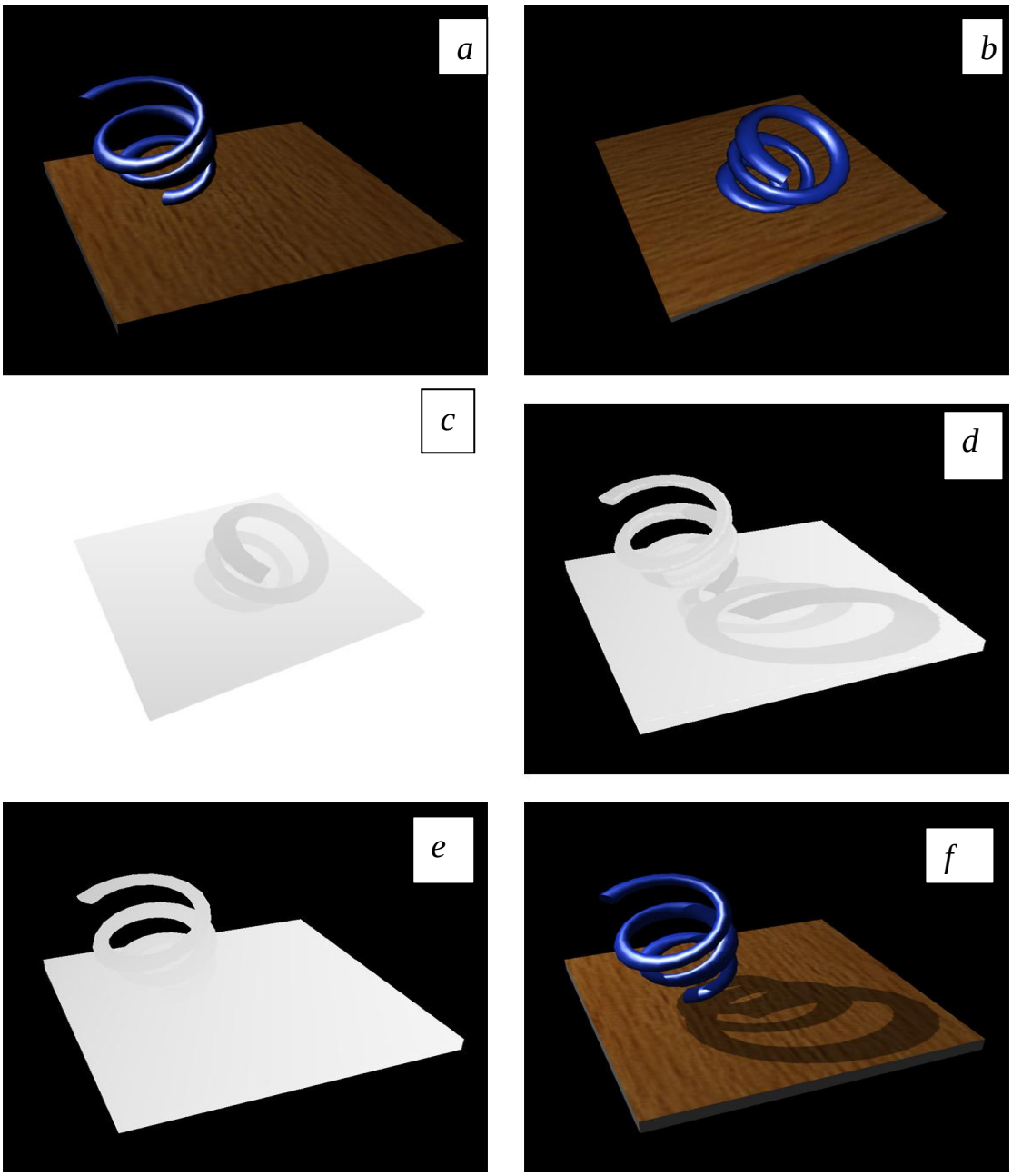
if(lightIntensity > 0.0f)

{

/ / Xác định màu khuếch tán cuối cùng dựa trên các màu khuếch tán và số tiền của cường độ ánh sáng.

*Color + = (diffuseColor * lightIntensity);*

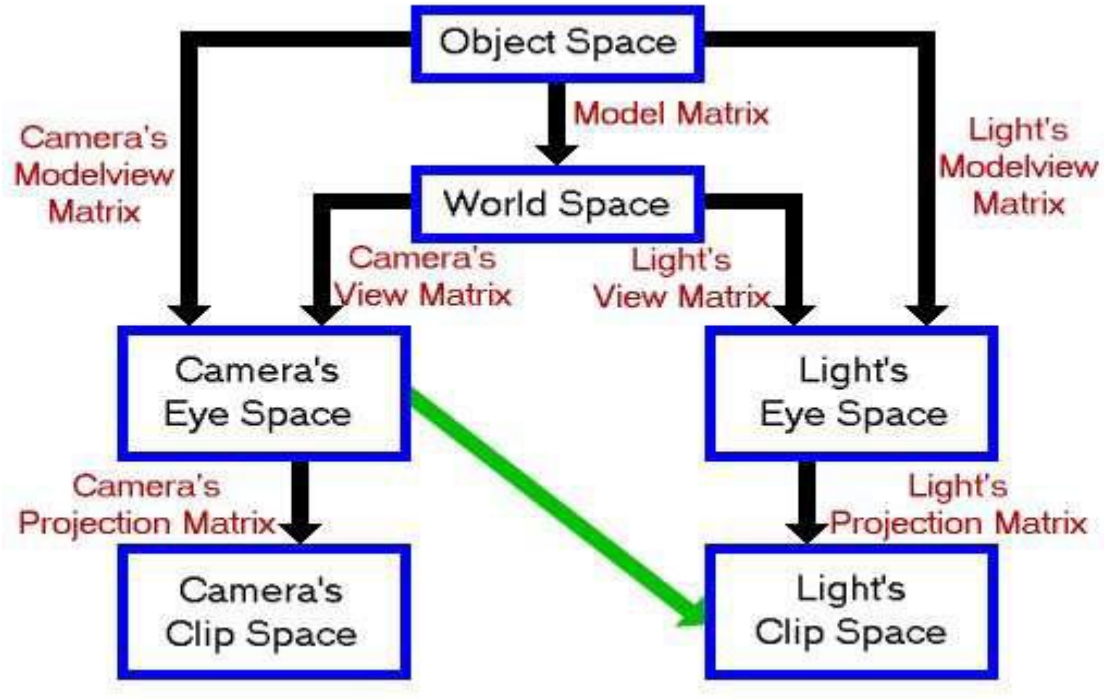
```
                    / / Nhúng màu sắc ánh sáng.
                        color = saturate(color);
                    }
                }
            }
        / / Mẫu màu sắc điểm ảnh từ các kết cấu bằng cách sử dụng mẫu ở kết cấu phối hợp vị trí.
        textureColor = shaderTexture.Sample (SampleTypeWrap, input.tex);
        / / Kết hợp ánh sáng và kết cấu màu sắc.
       color = color * textureColor;
        return color;
    }
```



Hình 2.6: Hình (a): Khung cảnh không có bóng đươc vẽ từ vị trí của điểm nhìn . Hình (b) : Khung cả nh đươc nhìn từ nguồn sáng. Hình (c) : Bản đồ bóng được tạo ra khi nhìn từ vị trí của nguồn sáng. Hình (d): Bản đồ bóng được chiếu lên trên khung cảnh được nhìn từ mắt(giá trịA). Hình (e): Chiếu khoảng cách phẳng của nguồn sáng lên khung cảnh được nhìn từ mắt (Giá trị B). Hình (f) : Bóng được vẽ ra sau phép kiểm tra chiều sâu giữa (d) và (e).

2.3. Chuyển tọa độ

Ta phải xác định chuyển tọa độ của một điểm từ hệ tọa độ của mắt về tọa độ hiển thị ánh sáng.



Hình 2.7: Các hệ tọa độ và các ma trận biến đổi

- Đầu tiên ta phải chuyển từ hệ tọa độ mắt (Camera's Eye Space) về hệ tọa độ thế giới bằng ma trận nghịch đảo của ma trận biến đổi hiển thị. (Camera's View Matrix = Viewing matrix). Ta ký hiệu ma trận này là VEV
- Sau đó ta sẽ phải đưa từ hệ tọa độ thế giới về hệ tọa độ của ánh sáng bằng ma trận hiển thị của ánh sáng (Light's Viewing Matrix). Kí hiệu ma trận này là VLV
- Tiếp theo thực hiện phép chiếu thích hợp (Phép chiếu trực giao) để đưa về hệ tọa độ thiết bị tiêu chuẩn. PS

Cuối cùng thực hiện phép biến đổi cổng nhìn đưa đoạn [-1,1] về đoạn [0,1] với ma trận S là:

$$\begin{bmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Như vậy ma trận T đưa một điểm từ hệ tọa độ của mắt về tọa độ hiển

thị của ánh sáng sẽ có dạng:

$$T = S\ P_S\ V_{LV}\ V_{EV}$$

2.4. Nhận xét.

Đây là thuật toán thường được áp dụng trong các thế giới có các vật dụng thủy tinh và các bề mặt đơn giản, vì mặc dù có thể đổ bóng được trên các vật thể phức tạp tuy nhưng tốn kém chi phí, hơn nữa bóng phụ thuộc vào kích thước shadow map nên nếu muốn bóng chất lượng cao phải tốn bộ nhớ để tăng kích thước texture.

Ưu điểm: Đổ bóng được trên các mặt cong, các mặt hình học phức tạp. Do quá trình tạo shadow map là quá trình render bình thường nên có thể render vật chắn sáng với độ trong suốt hay độ mờ đục.

Nhược điểm: Tốn thêm thời gian cho quá trình render shadow map.

- Thời gian xoá frame buffer
- Thời gian sao chép dữ liệu từ frame buffer sang bộ nhớ chính (cả CPU và GPU phải đồng thời xử lý).

Chất lượng bóng phụ thuộc vào kích thước texture làm shadow map, quá trình render vật chắn sáng tạo shadow map, độ xa gần của bề mặt đổ bóng, kích thước của vật chắn sáng.

Thời gian render bóng phụ thuộc vào vật hứng bóng.

Không có khả năng tự đổ bóng lên chính vật thể chắn sáng.

Có hiện tượng đổ bóng ngược.

CHƯƠNG 3: CHƯƠNG TRÌNH THỬ NGIỆM

3.1. Bài toán.

Thuật toán tạo bóng dựa trên bản đồ bóng là kỹ thuật tạo bóng trên không gian ảnh (image-space algorithm) nên những thông tin về hình dạng của vật thể hay những kiến thức hình học là không cần thiết. Thuật toán này cũng giống như thuật toán bóng khối, thực hiện phép kiểm tra "trong bóng" (in shadow) trên từng pixel. Một pixel được chiếu sáng nếu không có vật nào chắn giữa đường nối nó và nguồn sáng. Chìa khóa để hiểu được thuật toán Shadow Mapping (bản đồ bóng) là những điểm nằm trên bản đồ bóng chính là những điểm sẽ được hiển thị ra màn hình nếu như điểm nhìn đặt ở vị trí của ánh sáng.

3.2. Phân tích và thiết kế.

3.2.1. Giới thiệu về ngôn ngữ lập trình.

Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft. Nó được sử dụng để phát triển giao diện điều khiển và giao diện người dùng đồ họa ứng dụng cùng với Windows Forms, các trang web , các ứng dụng web , và các dịch vụ web trong cả mã nguồn gốc cùng với mã số quản lý cho tất cả các nền tảng được hỗ trợ bởi Microsoft Windows , Windows Mobile ,Windows CE , NET Framework. , . NET Compact Framework và Microsoft Silverlight .

Visual Studio bao gồm một trình biên tập mã hỗ trợ IntelliSense cũng như refactoring code. Là công cụ cho phép bạn viết mã, gỡ rối và biên dịch chương trình trong nhiều ngôn ngữ lập trình .NET khác nhau.

Được xây dựng bằng các ngôn ngữ bao gồm C /C + + (thông qua Visual C + +), VB.NET (thông qua Visual Basic.NET.), C # (thông qua Visual C #), và F #. Hỗ trợ cho các ngôn ngữ khác như M , Python , và của Ruby số những người khác có sẵn thông qua dịch vụ ngôn ngữ cài đặt riêng rẽ. Nó cũng hỗ trợ XML / XSLT , HTML / XHTML , JavaScript và CSS.

Microsoft Visual C++ (còn được gọi là MSVC) là một sản phẩm Môi trường phát triển tích hợp (IDE) cho các ngôn ngữ lập trình C, C++, và C++/CLI củaMicrosoft. Nó có các công cụ cho phát triển và gỡ lỗi mã nguồn C++, đặc biệt là các mã nguồn viết cho Microsoft Windows API, DirectX API, vàMicrosoft.NET Framework.

Các chức năng của Visual C++ như tô sáng cú pháp, IntelliSense (chức năng về tự động hoàn thành việc viết mã) và các chức năng gỡ lỗi tiên tiến.

Đặc trưng biên dịch và xây dựng hệ thống, tính năng tiền biên dịch các tập tin đầu đề (header files) và liên kết tịnh tiến (incremental link) - chỉ liên kết những phần bị thay đổi trong quá trình xây dựng phần mềm mà không làm lại từ đầu: Những đặc trưng về tính năng này thuyên giảm tổng thời gian biên tập, biên dịch và liên kết chương trình phần mềm, đặc biệt đối với những đề án phần mềm lớn.

3.2.2. Chức năng một số hàm trong chương trình.

```
//Một số thư viện liên kết.
    #pragma comment(lib, "opengl32.lib")
    #pragma comment(lib, "glu32.lib")
    #pragma comment(lib, "winmm.lib")
//Thiết lập camera.
    camera.Init(VECTOR3D(-2.5f, 3.5f,-2.5f));
//Thiết lập ánh sáng.
    light.Init(VECTOR3D(2.0f, 3.0f, -2.0f), VECTOR3D(0.0f,-0.5f, 0.0f));
    light.SetClipDistances(2.0f, 8.0f);
    light.UpdateMatrices();
//thiết lập khung nhìn
    int height;
    if (window.height==0)
        height=1;
    else
    height=window.height;
        glViewport(0, 0, window.width, height);
```

```
//thiết lập ma trận chiếu
        glMatrixMode(GL_PROJECTION);  //chọn ma trận chiếu
        glLoadIdentity();

        gluPerspective(45.0f, (GLfloat)window.width/(GLfloat)height, 1.0f, 500.0f);
//tải bản sắc modelview
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
//cập nhật khung
        window.Update();
        currentInteractor->Update();
//chuyển đổi camera hoặc ánh sáng
        if(window.isKeyPressed('C'))
                currentInteractor=&camera;

        if(window.isKeyPressed('L'))
                currentInteractor=&light;
//chuyển đổi hình
        if(window.isKeyPressed('T'))
                objectType=TORI;
        if(window.isKeyPressed('B'))
                objectType=SPHERES;
```

3.3. Thực nghiệm chương trình và đánh giá kết quả.

3.3.1. Thực nghiệm chương trình.

Một số hướng dẫn:

Nhấn C: Chuyển chế độ sang dịch chuyển Camera.

Nhấn L: Chuyển chế độ sang dịch chuyển nguồn sáng.

Click chuột trái và di chuyển để dịch chuyển (nguồn sáng hoặc camera tùy thuộc đang ở chế độ nào)

Ấn Space: Để vẽ vùng quan sát (Frustum)

Nhấn T: Để chuyển vật thể sang hình chiếc vòng .

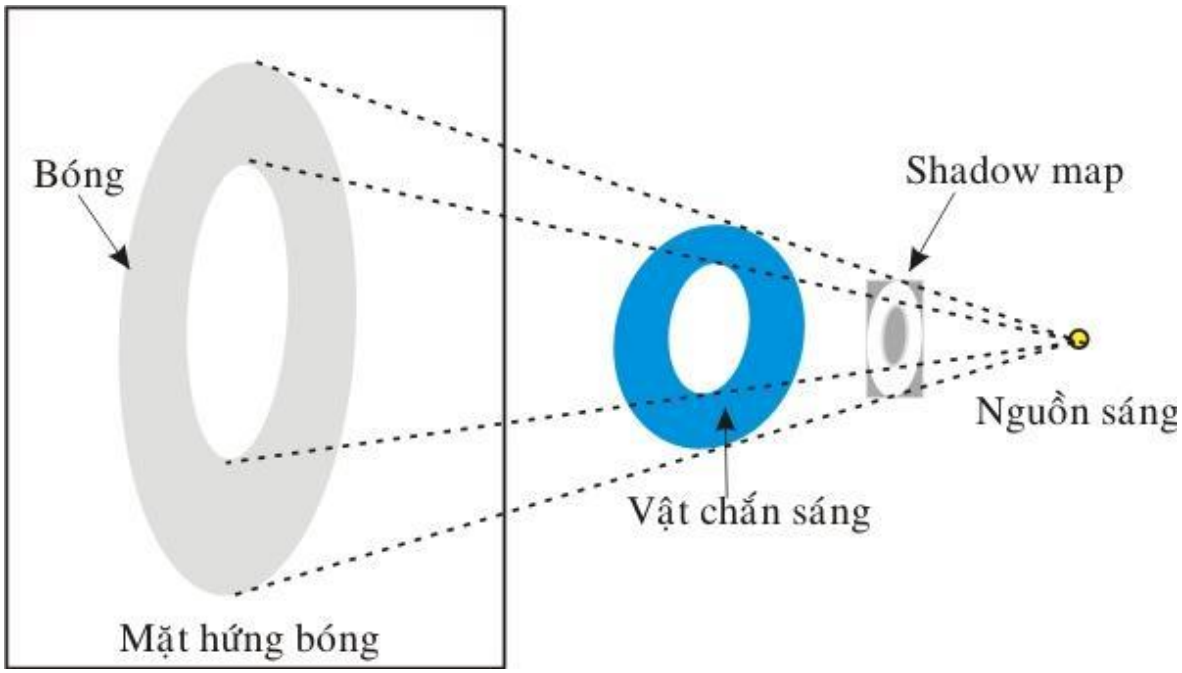
Nhấn B: Để chuyển vậ t thể thành hình cầu .

Nhấn (↑) hoặ c (↓): Để tăng hoặc giảm kích thước của Bản đồ bóng.

- Mô tả:

Đầu tiên ta thiết lập một phép chiếu phối cảnh với tâm là nguồn sáng.

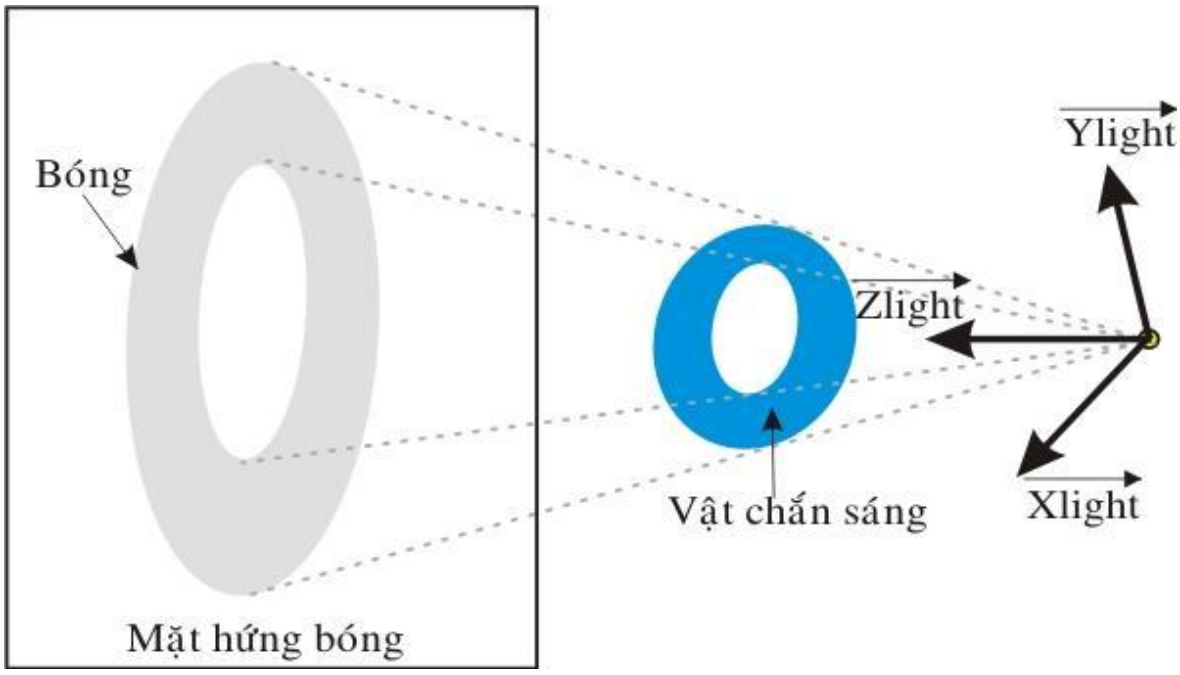
Phép chiếu này chiếu vật chắn sáng lên trên một mặt phẳng ảo ở giữa nguồn sáng và vật thể tạo ra shadow map của vật thể đó.



Hình 3.1: Chiếu Shadow Map

Để thực hiện phép chiếu trên, trước tiên ta định nghĩa một hệ tọa độ có gốc là nguồn sáng và trục Z hướng về phía vật chắn sáng. Trục Z của hệtọa độ này sẽ xác định đường tâm của phép chiếu trong khi đó mặt phẳng tạo bởi 2 trục X-Y xác định các trục của mặt phẳng ảo mà chúng ta chiếu shadow map lên đó. Nếu chuyển vật chắn sáng sang hệ tọa độ nguồn sáng ta có thể chiếu nó lên mặt phẳng ảo một cách dễ dàng.

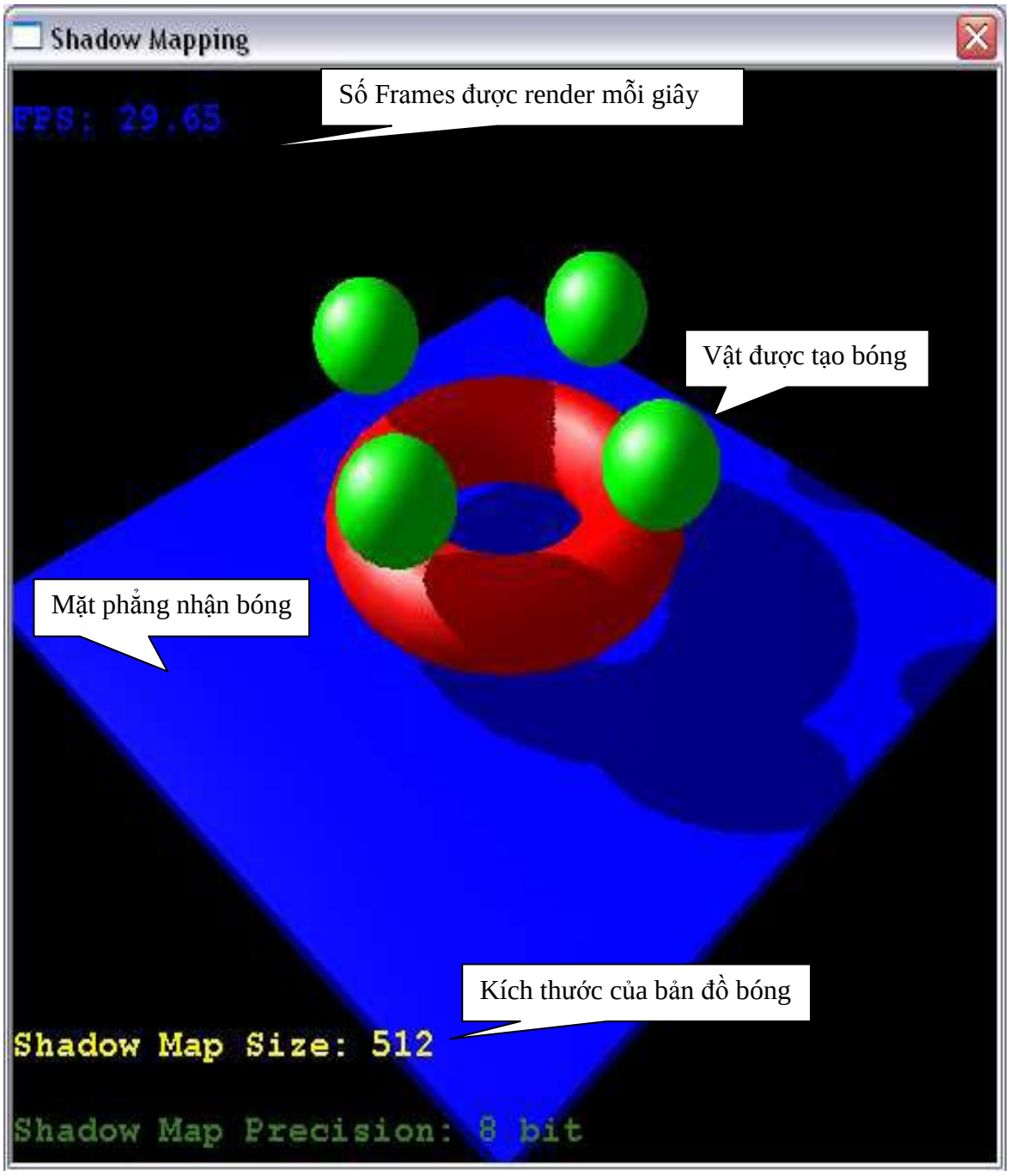
Để render một hệ tọa độ bất kỳ, chúng ta cần phải biết được gốc và các trục của nó. Trong hệ tọa độ nguồn sáng, chúng ta đã biết được gốc là nguồn sáng , các trục còn lại được gọi lần lượt là Xlight, Ylight, Zlight, tất cả các vector còn lại đều thuộc về hệ tọa độ thế giới thực.



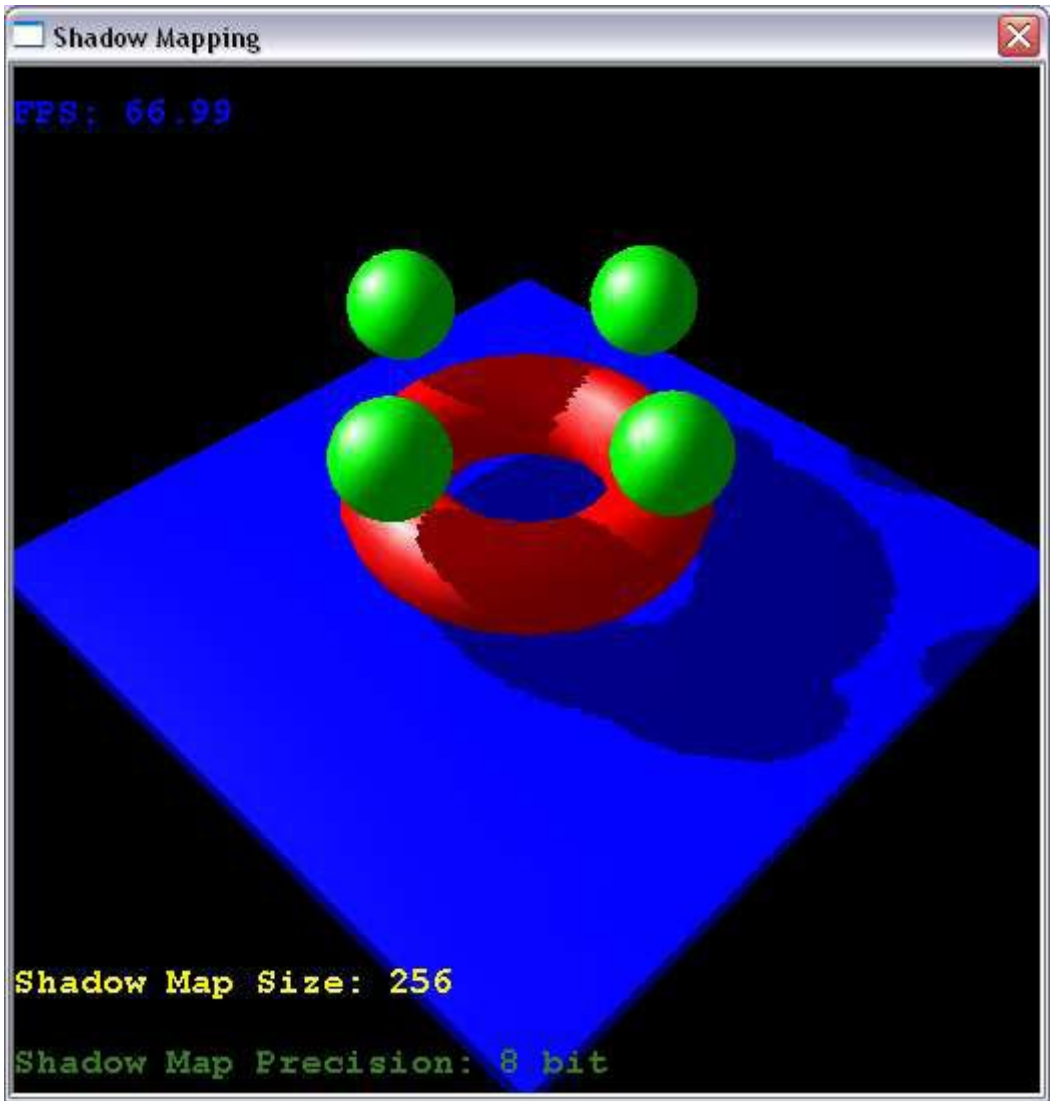
Hình 3.2: Hệ tọa độ nguồn sáng

Để render một hệ tọa độ bất kỳ, chúng ta cần phải biết được gốc và các trục của nó. Trong hệ tọa độ nguồn sáng, chúng ta đã biết được gốc là nguồn sáng , các trục còn lại được gọi lần lượt là Xlight, Ylight, Zlight, tất cả các vector còn lại đều thuộc về hệ tọa độ thế giới thực.

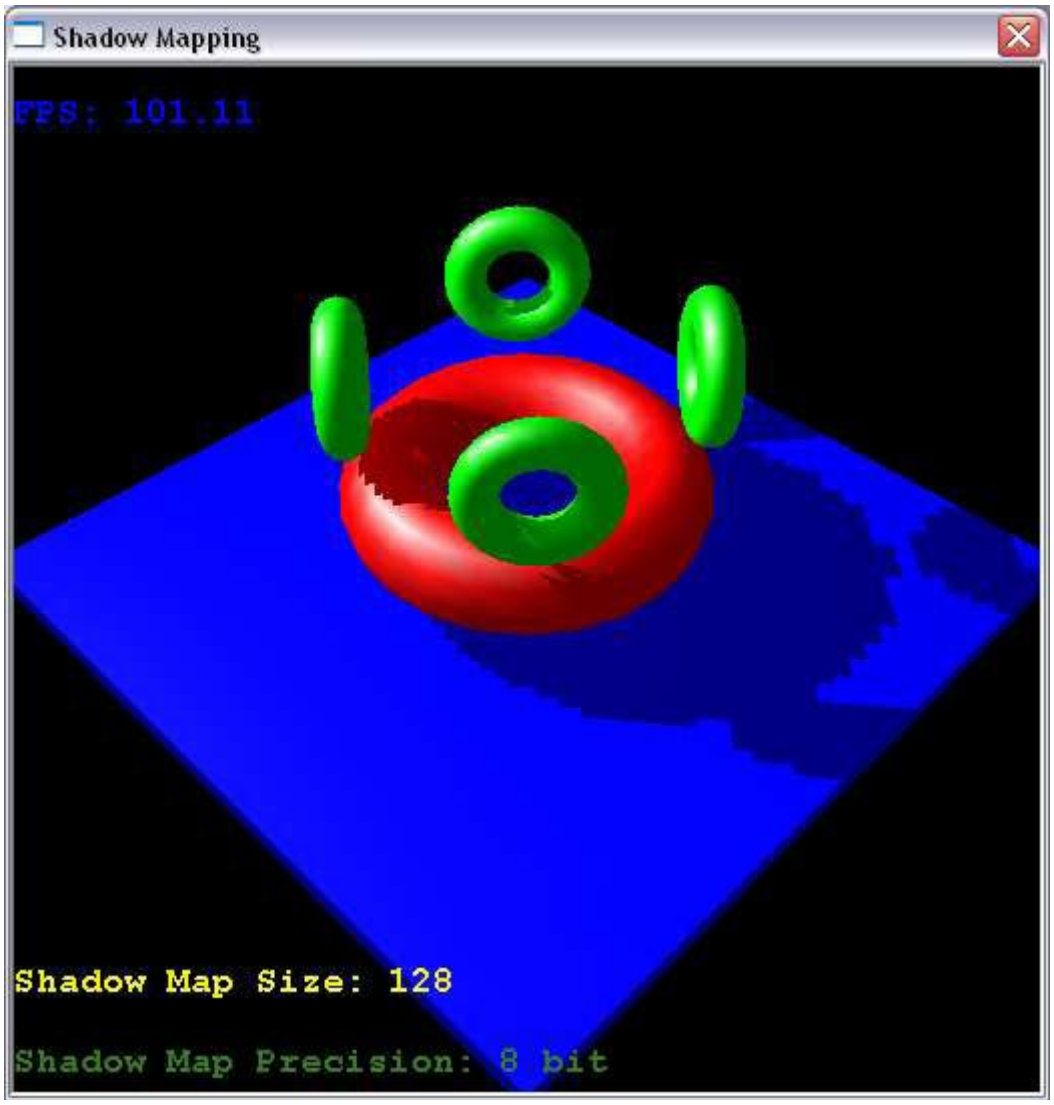
3.3.2. Kết quả thực hiện.



Hình 3.3: Bóng được tạo với kích thước bản đồ bóng là 512. Số frames được render mỗi giây khá thấp do kích thước bản đồ bóng tăng , phải tính toán nhiều.



Hình 3.4: Bóng được tạo với kích thước bản đồ bóng là 256. Số Frames được render mỗi giây tăng lên khá cao do số lượng tính toán giảm.



Hình 3.5: Bóng được tạo với kích thước bản đồ bóng là 128. Số Frames được render mỗi giây lên đến 101.

PHẦN KẾT LUẬN

Mỗi đồ án là một quá trình tìm hiểu, nghiên cứu thực tế, quá trình làm đồ án cũng là quá trình đúc kết kinh nghiệm của mỗi người.

Qua quá trình làm đồ án, sau khi phân tích, tìm hiểu chung về các ứng dụng của đồ hoạ 3D, em đã bổ sung thêm cho mình nhiều kiến thức quý giá. Em đã tìm hiểu sâu hơn, đầy đủ hơn về các kỹ thuật tạo bóng. Em đã có thêm những kiến thức mới về các ứng dụng của đồ hoạ máy tính trong việc tạo ra các loại bóng, ứng dụng của các kỹ thuật tạo bóng trong các lĩnh vực như Điện ảnh, Hoạt hình, kiến trúc và các ứng dụng xây dựng các mô hình thực tại ảo. Em đã tìm hiểu được 4 kỹ thuật tạo bóng cứng và 3 kỹ thuật tạo bóng mềm. Tuy nhiên trong đồ án em có đi sâu giới thiệu về kỹ thuật tạo bóng dùng bản đồ bóng.

Sau một quá trình nghiên cứu làm đồ án với sự chỉ dẫn nhiệt tình của thầy giáo hướng dẫn em đã học được cách tìm hiểu, phân tích và nghiên cứu một vấn đề khoa học mới. Trong thời gian làm đồ án tốt nghiệp, mặc dù bản thân đã rất nỗ lực, cố gắng, đầu tư nhiều thời gian, công sức cho việc tìm hiểu nghiên cứu đề tài và đã nhận được sự chỉ bảo, định hướng tận tình của thầy giáo hướng dẫn cùng các anh, chị đi trước nhưng do hạn chế về mặt thời gian và khó khăn trong việc tìm kiếm tài liệu, hạn chế về mặt kiến thức của bản thân, nên chưa có được kết quả thực sự hoàn hảo. Kính mong các thầy cô giáo cũng như các bạn chỉ bảo và giúp đỡ.

Hướng phát triển:

Đồ án tuy đã đi sâu nghiên cứu về một số kỹ thuật tạo bóng, đưa ra được mô phỏng kỹ thuật tạo bóng cứng là dùng bản đồ bóng. Nhưng hiện nay với tốc độ phát triển nhanh chóng của ngành công nghệ thông tin nói chung và của các kỹ thuật đồ hoạ nói riêng thì đòi hỏi cần phải đi sâu hơn nữa để nghiên cứu thêm các kỹ thuật tạo bóng mới, mô phỏng thêm một số kỹ thuật tạo bóng khác nữa.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Lương Mạnh Bá, Nguyễn Thanh Thủy (1999), Nhập môn Xử lý ảnh số, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.

2. Trần Thanh Hiệp (2004), "Tìm hiểu ngôn ngữ VRML và ứng dụng", Khoá luận tốt nghiệp đại học, Khoa Công nghệ - Đại học Quốc gia, Hà Nội.

3. Phạm Anh Phương , Nguyễn Hữu Tài (2006), Giáo trình Lý thuyết Đồ họa, Nhà xuất bản Khoa học và Kỹ Thuật, Hà Nội.

4. Nguyễn Hải Thanh (2005), " Shadow Technique_Thesis", Khoá luận tốt nghiệp đại học, Khoa Công nghệ - Đại học Quốc Gia, Hà Nội.

5. Đỗ Năng Toàn, Phạm Việt Bình (2008), Giáo trình Xử lý ảnh, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.

Tiếng Anh

6. Andrew V.Nealen (2000), "Shadow Volume and Shadow Mapping, Recent Development in Real-time Shadow Rendering", University of British Columbia.

7. Dietrich, Sim (2003), "Shadow Techniques", Game Developers

Convention 2001, http://developer.nvidia.com/attach/1308, 2003-11-01.

8. Jackie Neider, Tom Davis, Mason Woo (1997), "The Red Book", Addison

Wesley Publisher.

9. Woo, Andrew Poulin, Pierre Fournier, Alain (1990) , "A Survey of shadow

algorithms", IEEE CG&A.

10. Williams, Lance (1978), "Casting curved shadows on curved surfaces",

Computer Graphics 12(8), pp. 270-274.

11. Mark Kilgard, "Shadow Mapping with Today's Hardware", Technical

Presentation: http://developer.nvidia.com/view.asp?IO=cedec_shadowmap.

12. Everitt, CassRege, Ashu Cebenoyan, Cem (2003), Hardware Shadow Mapping, http://developer.nvidia.com/attach/5708, 2003-12-17.

13. Ikrima Elhassan (2007), Shadow Algorithms, 20-02-2007.