MỞ ĐẦU

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

Đồ án được chia làm 4 chương:

- - Chương 1: Tổng quan về kỹ thuật đồ họa máy tính.
- - Chương 2: Tìm hiểu kỹ hơn về một số kỹ thuật ứng dụng trong đồ họa 3D.
- - Chương 3: Tìm hiểu tổng quan về công nghệ Silverlight.
- - Chương 4: Xây dựng Album ảnh 3D bằng Silverlight.

CHƯƠNG 1 : TỔNG QUAN VỀ KỸ THUẬT ĐỒ HỌA MÁY TÍNH

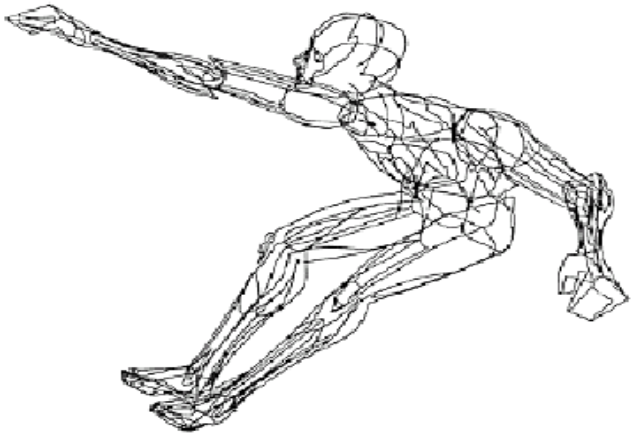
1.1 Các khái niệm tổng quan về kỹ thuật đồ họa máy tính

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

Thuật ngữ đồ họa máy tính (*computer graphics*) do William Fetter đặt ra năm 1960 để mô tả một cách thiết kế mới khi đang làm việc tại hãng Boeing. Với cách này giúp tạo nhiều ảnh có thể sử dụng lại để có thể dễ dàng thiết kế buồng lái của phi công theo ý muốn.

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

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



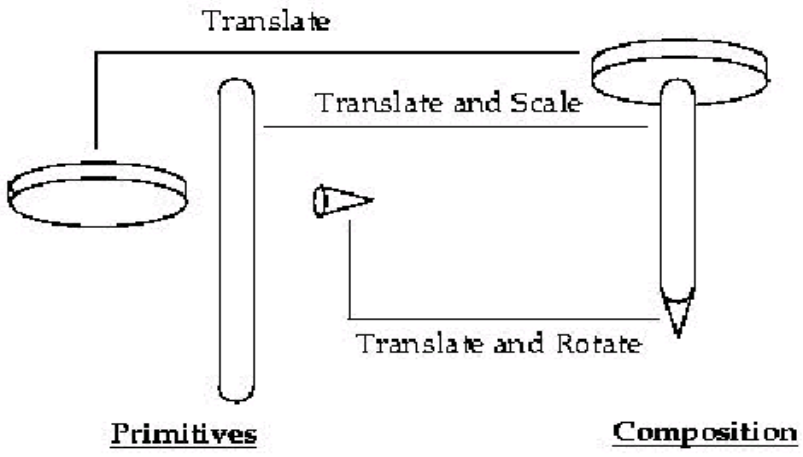
Hình 1.1: Đồ họa máy tính

Đồ họa máy tính (*computer graphics*) có thể được hiểu như là tất cả những gì liên quan đến việc tạo ra ảnh (image) bằng máy tính. Chúng bao gồm: tạo, lưu trữ, thao tác trên các mô hình (model) và các ảnh.

1.2 Tổng quan về một hệ đồ họa

- Các thành phần phần cứng :
    - Thiết bị nhập : chuột, bàn phím, ...
    - Thiết bị hiển thị: màn hình, máy in, ...
- Các công cụ phần mềm:
    - Công cụ ứng dụng (application package) : thiết kế cho người sử dụng để tạo ra các hình ảnh mà không cần quan tâm tới các thao tác bên trong hoạt động mhư thế nào. Ví dụ: AutoCAD, Adobe Photoshop, 3D Studio, ...
    - Công cụ lập trình (programming package) : Cung cấp một tập các hàm đồ họa có thể được dùng trong các ngôn ngữ lập trình cấp cao như C, Pascal, ... Ví dụ : GRAPH.TPU, GRAPHICS.LIB, Open GL, ...
- Các chuẩn phần mềm :
    - Ra đời để đáp ứng tính tương thích : Nếu các phần mềm đuợc thiết kế với các hàm đồ họa chuẩn chúng có thể dùng được cho nhiều hệ phần cứng và môi trường làm việc khác nhau.
    - GKS (Graphics Kernel System) là chuẩn ra đời đầu tiên cho việc phát triển các phần mềm đồ họa. Ban đầu GKS được thiết kế chỉ dùng cho tập các công cụ đồ họa hai chiều, sau đó mới được mở rộng ra cho đồ họa ba chiều.
    - Các hàm GKS thực sực chỉ là các mô tả trừu tượng, độc lập với bất kỳ ngôn ngữ lập trình nào. Để cài đặt một chuẩn đồ hoạ cho ngôn ngữ cụ thể nào, các cú pháp tương ứng sẽ được xác định và cụ thể hóa.

- Các thành phần của công cụ lập trình:
  - Tập các công cụ tạo ra các đối tượng đồ họa cơ sở như điểm, đoạn thẳng, đường cong, vùng tô, kí tự, ...
  - Tập các công cụ thay đổi thuộc tính của các đối tượng cơ sở kể trên như màu sắc, kiểu đường, kiểu chữ, mầu tô, ...
  - Tập các công cụ thực hiện các phép biến đổi hình học dùng để thay đổi kích thước, vị trí, hướng, ...
  - Tập các công cụ biến đổi hệ quan sát dùng để xác định vị trí quan sát của các đối tượng và vị trí trên thiết bị hiển thị đối tượng.
  - Tập các công cụ nhập liệu : các ứng dụng đồ họa có thể sử dụng nhiều loại thiết bị nhập khác nhau như chuột, bàn phím, bút vẽ, bảng, ... để điều khiển và xử lí dòng dữ liệu nhập.
  - Tập các công cụ chứa các thao tác dùng cho quản lí và điều khiển như khởi tạo và đóng chế độ đồ họa, xóa toàn bộ màn hình, ...



Hình 1.2: Hình ảnh minh họa

1.3 Các kỹ thuật đồ họa

1.3.1 Kỹ thuật đồ họa điểm

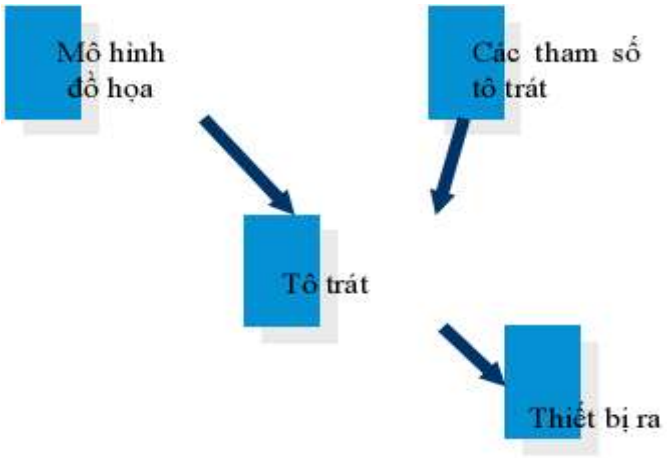
- Các mô hình, hình ảnh của các đối tượng được hiển thị thông qua từng pixel (từng mẫu rời rạc).
- Đặc điểm: có thể thay đổi thuộc tính
  - Xoá đi từng pixel của mô hình và hình ảnh các đối tượng.
  - Các mô hình hình ảnh được hiển thị như một lưới điểm (grid) các pixel rời rạc.
  - Từng pixel đều có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

Hình 1.3: Ảnh đồ họa điểm

Phương pháp để tạo ra các pixel:

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

1.3.2 Kỹ thuật đồ họa vector



Hình 1.4: Mô hình đồ họa vector

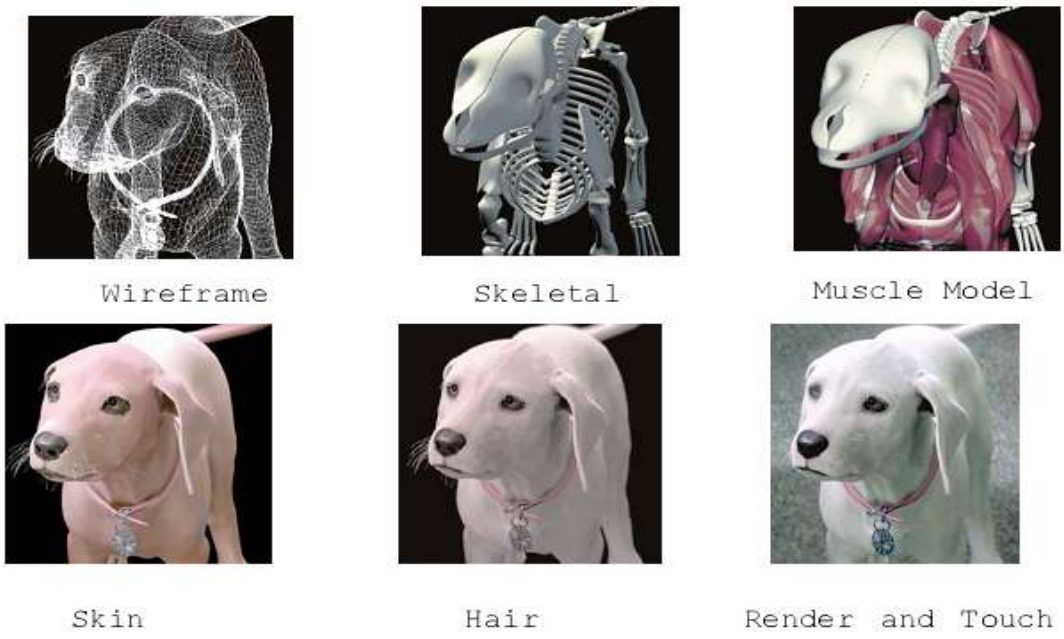
- Mô hình hình học (geometrical model) cho mô hình hoặc hình ảnh của đối tượng.
- Xác định các thuộc tính của mô hình hình học này.
- Quá trình tô trát (rendering) để hiển thị từng điểm của mô hình, hình ảnh thực của đối tượng.

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

So sánh đồ họa điểm và đồ họa vector

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

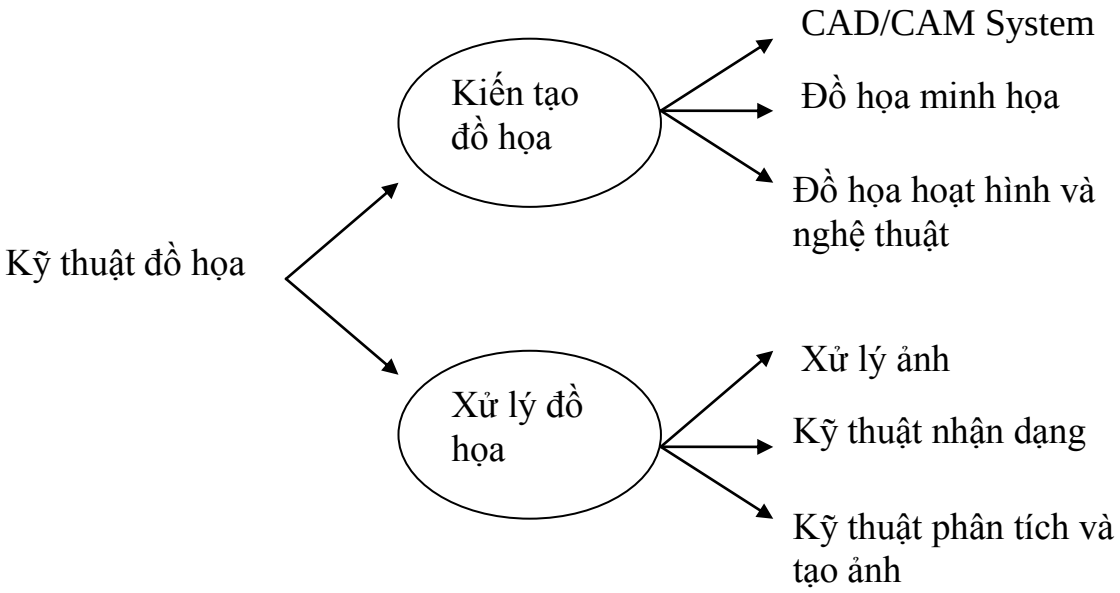
Ví dụ về hình ảnh đồ họa vector:



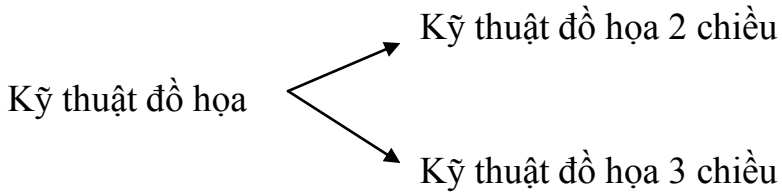
Hình 1.5: Ví dụ về đồ họa vector

1.3.3 Phân loại của đồ họa máy tính

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



Phân loại theo hệ tọa độ:



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

Các lĩnh vực của đồ họa máy tính:

- Kỹ thuật xử lý ảnh (*Computer Imaging*): sau quá trình xử lý ảnh cho ta ảnh số của đối tượng, Trong quá trình xử lý ảnh sử dụng rất nhiều các kỹ thuật phức tạp: kỹ thuật khôi phục ảnh, kỹ thuật làm nổi ảnh, kỹ thuật xác định biên ảnh.
- Kỹ thuật nhận dạng (*Pattern Recognition*): từ những ảnh mẫu có sẵn ta phân loại theo các cấu trúc, hoặc theo các tiêu trí được xác định từ trước và bằng các thuật toán chọn lọc để có thể phân tích hay tổng hợp các ảnh gốc, các ảnh gốc này được lưu trong một thư viện và căn cứ vào thư viện này ta xây dựng được các thuật giải phân tích và tổ hợp ảnh.
- Kỹ thuật tổng hợp ảnh (*Image Synthesis*): là lĩnh vực xây dựng mô hình và hình ảnh của các vật thể dựa trên các đối tượng và mối quan hệ giữa chúng.
- Các hệ CAD/CAM (Computer Aided Design/Computer Aided Manufacture System): kỹ thuật đồ họa tập hợp các công cụ, các kỹ thuật trợ giúp cho thiết kế các chi tiết và các hệ thống khác nhau: hệ thống cơ, hệ thống điện, hệ thống điện tử…

- Đồ họa minh họa (Presentation Graphics): gồm các công cụ giúp hiển thị các số liệu thí nghiệm một cách trực quan, dựa trên các mẫu đồ thị hoặc các thuật toán có sẵn.
- Đồ họa hoạt hình và nghệ thuật: bao gồm các công cụ giúp cho các họa sĩ, các nhà thiết kế phim hoạt hình chuyên nghiệp làm các kỹ xảo hoạt hình, vẽ tranh… ví dụ: phần mềm Studio, 3D Animation, 3D Studio Max.

1.4 Các ứng dụng tiêu biểu của kỹ thuật đồ họa

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

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

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

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

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

- Tự động hóa văn phòng và chế bán điện tử.
- Thiết kế với sự trợ giúp của máy tính (CAD_CAM).
- Lĩnh vực giải trí, nghệ thuật và mô phỏng.
- Điều khiển các quá trình sản xuất (Process Control).
- Lĩnh vực bản đồ (Cartography).
- Giáo dục và đào tạo.

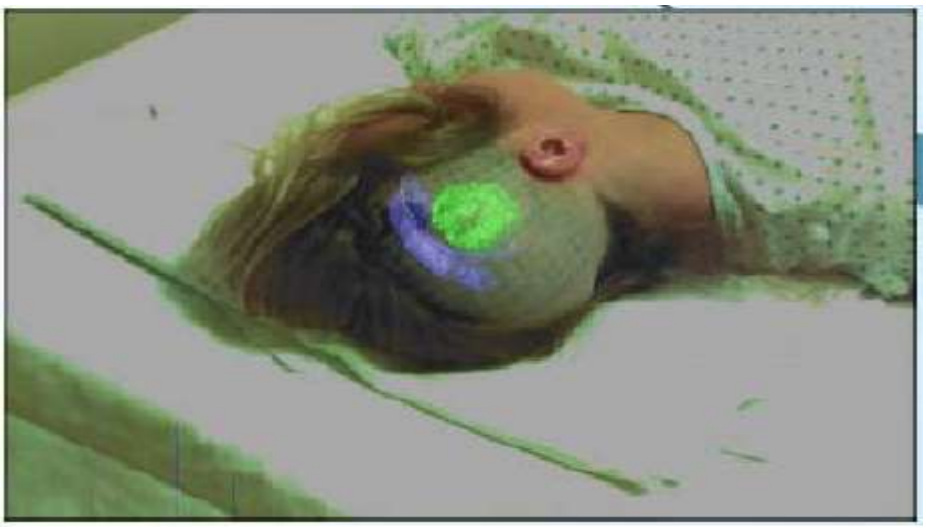
CHƯƠNG 2: TỔNG QUAN VỀ KỸ THUẬT ĐỒ HỌA 3D

2.1 Giới thiệu 3D graphics

Hình ảnh 3D đã xuất hiện từ năm 1833 khi lần đầu tiên hình nổi được giới thiệu bởi ngài Charlets Wheastone. Và ngày nay hiển thị 3 chiều ngày càng trở lên phổ biến, nhất là một số lĩnh vực đặc biệt. Do các thiết bị kĩ thuật số ngày càng rẻ hơn và phổ biến hơn, sự cải thiện trong băng thông mạng và các chuẩn nén, sự tiến bộ trong công nghệ hiển thị, 3D nổi lên như một công nghệ của tương lai. Hiển thị lập thể rất có ích trong nhiều lĩnh vực ứng dụng như: Các hệ thống ứng dụng như: Các hệ thống mô phỏng, các hệ thống y học, hệ thống robot học, thiết kế hỗ trợ máy tính, viễn thông và giải trí.

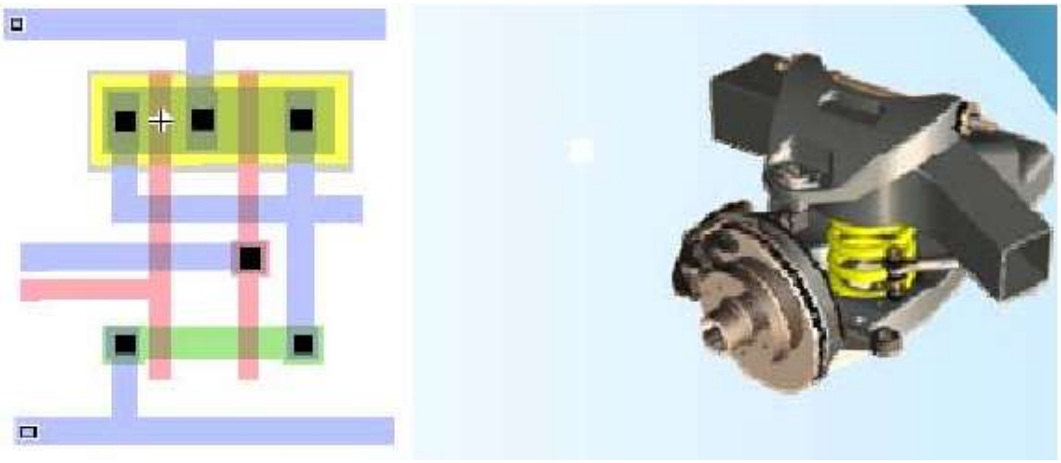
Dưới đây là một số hình ảnh minh hoạ cho ứng dụng của 3D:

Hình 2.1 : 3D trong giải trí (Games)

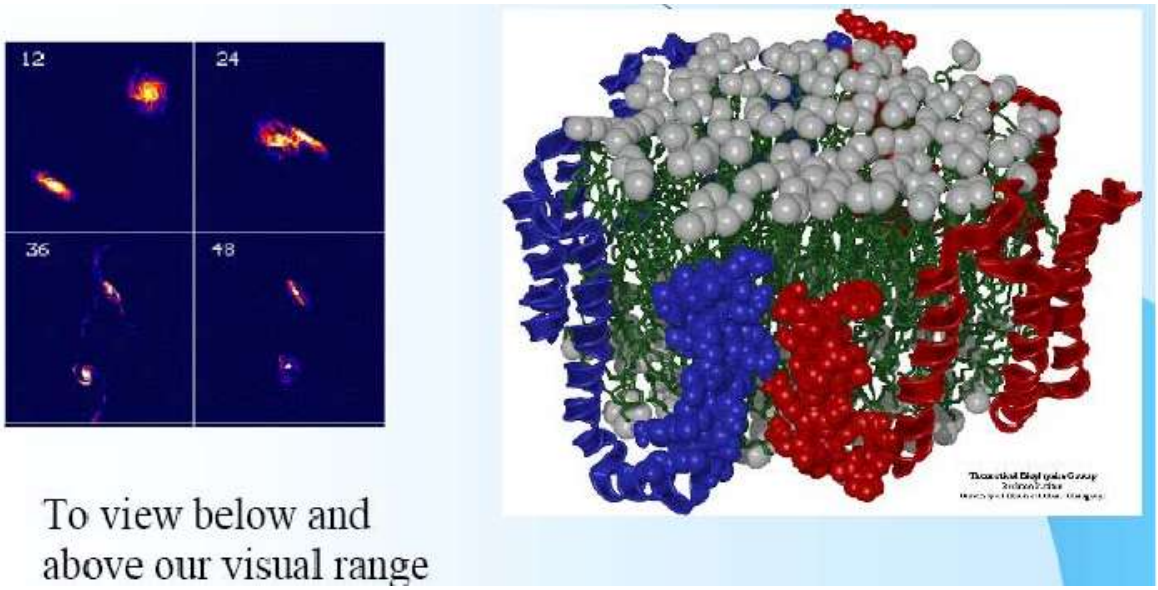


Hình 2.2: 3D trong Y học (Medical Imaging)

Hình 2.3: 3D trong hoạt hình (Animation)



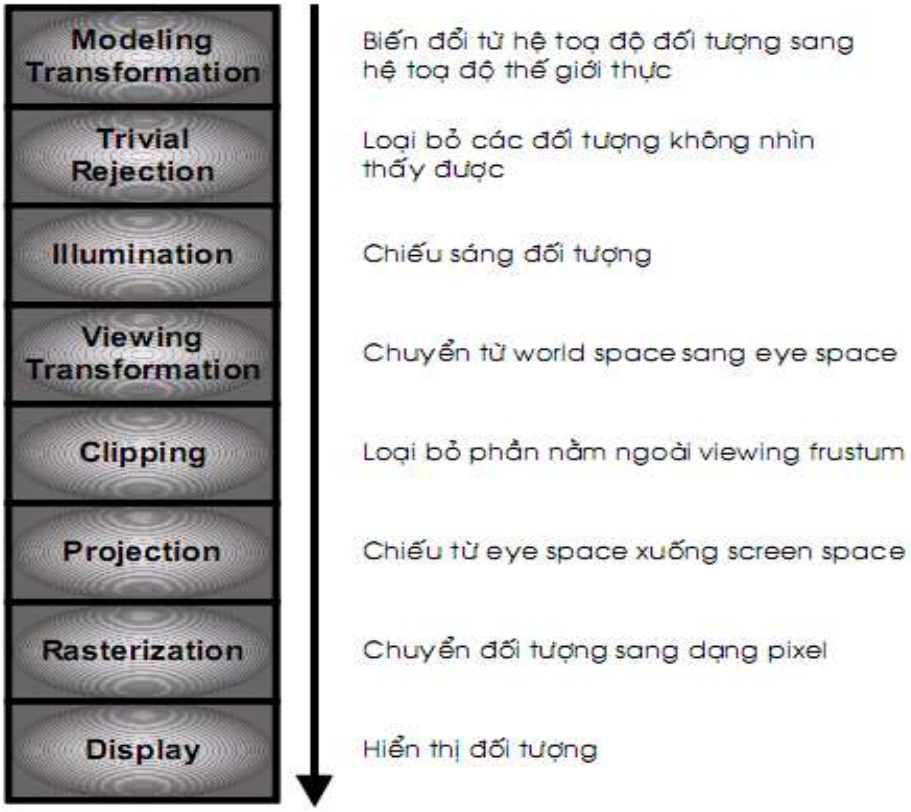
Hình 2.4: 3D trong thiết kế (Computer Aided)



Hình 2.5: 3D trong mô phỏng khoa học

Trước khi có thể tạo được các đồ hoạ 3D và stream nó qua mạng, chúng ta sẽ dành một chút thời gian để nắm được một số từ vựng về 3D, khái niệm cơ bản đồ hoạ 3D và các hệ thống toạ độ của nó. Khi đó chúng ta sẽ hiểu được rằng tại sao các đồ hoạ 3D lại được hiển thị trên một màn hình máy tính phẳng.

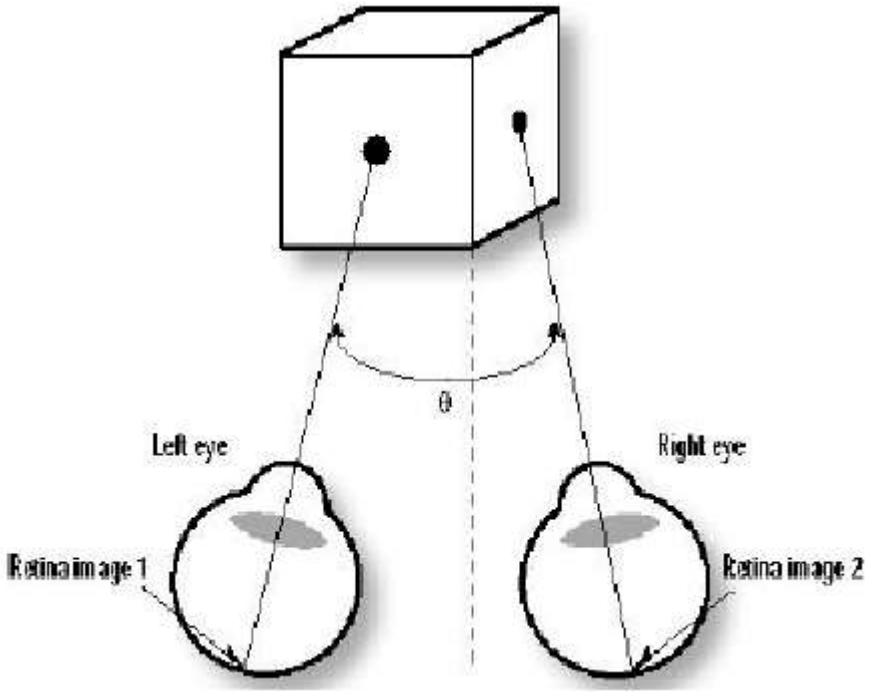
Quy trình hiển thị :



Hình 2.6: Quy trình hiển thị

2.2. Nhận thức về 3D

Thực sự "3D computer graphics" là các hình ảnh hai chiều trên một màn hình máy tính phẳng mà nó được cung cấp thêm ảo ảnh về độ sâu hay gọi là kích thước thứ ba. Để thực sự nhìn hình 3D, bạn cần phải xem đối tượng bằng cả hai mắt hay cung cấp cho mỗi mắt với các ảnh riêng biệt và duy nhất của đối tượng. Như hình 2.7 mỗi mắt nhận một ảnh hai chiều trên võng mạc. Hai ảnh này hơi khác nhau vì chúng được nhận tại hai góc độ khác nhau. Sau đó bộ não sẽ tập hợp hai ảnh này để được một bức tranh 3D phức hợp trong đầu bạn.



Hình 2.7: Cách đôi mắt nhìn thấy 3 kích thước

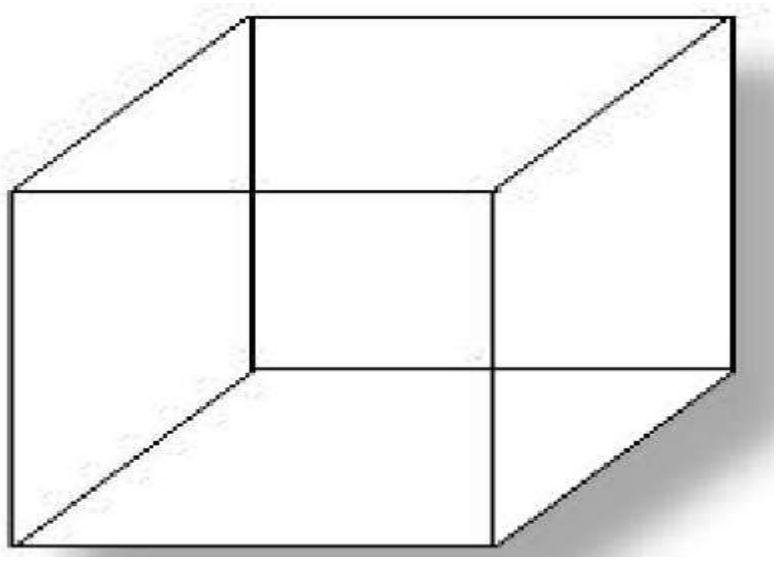
Hiệu quả 3D sẽ được phóng đại bằng cách làm tăng góc giữa hai ảnh.

Nếu bạn che một mắt thì chuyện gì xảy ra? Bạn có thể nghĩ bạn vẫn nhìn thấy ba chiều, nhưng hãy thử một thí nghiệm này. Đặt một cái cốc hay một cái gì đó ngoài tầm với của bạn và ở phía bên tay trái. Che mắt phải với tay phải của bạn và với lấy cái cốc. Chú ý rằng bạn sẽ mất một khoảng thời gian khó khăn để tính toán khoảng cách bao xa để sờ được cái cốc. Bây giờ bạn bỏ tay ra và lấy cái cốc thì bạn thấy rất dễ dàng. Điều này giải thích tại sao những người mất một mắt sẽ rất khó khăn trong nhận biết khoảng cách.

2.3. 2D + Phối cảnh = 3D

Lý do tại sao thế giới không đột ngột trở thành 2D khi bạn che một mắt đó là nhiều hiệu quả của thế giới 3D cũng hiện diện trong thế giới 2D. Điều này chỉ đủ để kích hoạt khả năng trong não của bạn nhận thức về độ sâu. Một gợi ý rõ ràng nhất đó là các đối tượng ở gần sẽ thấy lớn hơn các đối tượng ở xa. Hiệu quả này gọi là Phối cảnh hay luận xa gần (perspective). Và phối cảnh cùng với mầu(color), bề mặt(textures), ánh sáng(lighting), đánh bóng (shading) và những thay đổi về cường độ màu thêm vào nhận thức của bạn về

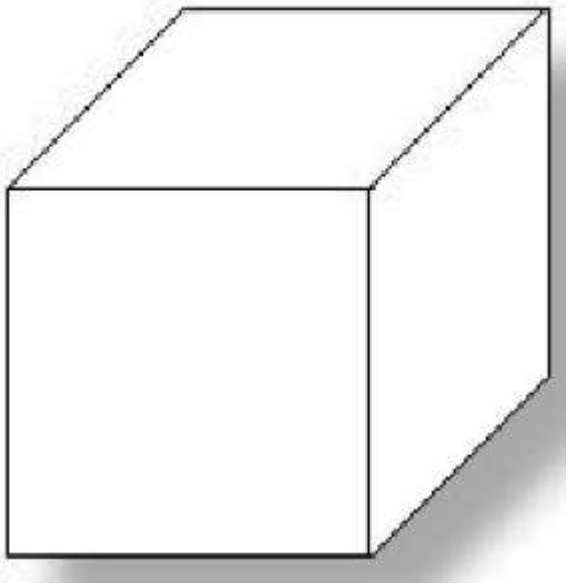
một ảnh kích thước ba chiều. Chỉ cần phối cảnh cũng đủ để xuất hiện hình ảnh ba chiều. Hình 2.8 cho thấy không cần màu hay bóng đổ … chỉ cần phối cảnh, hình lập phương vẫn xuất hiện như một đối tượng ba chiều. Tuy nhiên hãy nhìn thật kĩ hình lập phương trong một khoảng thời gian bạn sẽ thấy mặt trước và mặt sau của hình hoán đổi vị trí cho nhau. Điều này là do bộ não không rõ ràng vì thiếu bất kì bề mặt trong khi vẽ.



Hình 2.8: Phối cảnh hình lập phương

2.4. Loại bỏ các đường ẩn (Hidden Line Removal)

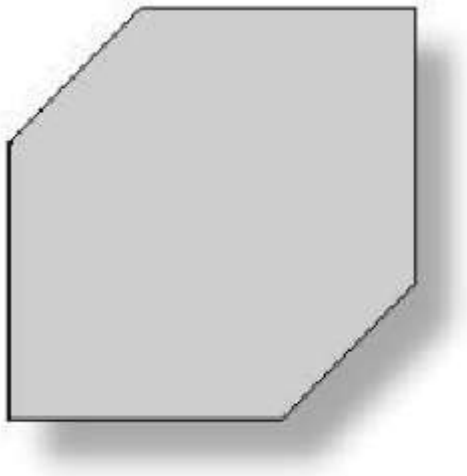
Hình 2.8 chỉ đủ để biểu diễn thông tin về ba chiều nhưng không đủ để phân biệt mặt trước và mặt sau của hình lập phương. Khi đang nhìn một đối tượng ba chiều, cách để bạn nói mặt trước và mặt sau là như thế nào? Đơn giản là mặt sau bị mờ hơn mặt trước. Nếu hình lập phương ở hình 2.8 là ở thể rắn thì bạn không thể nhìn thấy các góc ở đằng sau hình lập phương và như thế bạn sẽ không xáo trộn chúng với các góc đằng trước của hình lập phương. Để mô phỏng điều này trong khi vẽ hai chiều thì các đường bị tối đi do mặt trước của hình sẽ bị loại bỏ. Cái này gọi là hidden line removal, và minh hoạ như hình 2.9.



Hình 2.9: Hình lập phương sau khi loại bỏ các đường ẩn

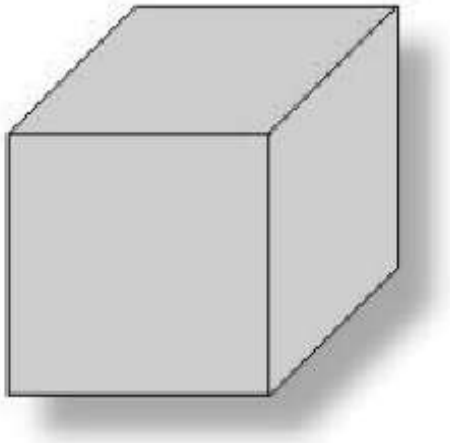
2.5. Color và shading

Hình 2.9 trông vẫn không giống như đối tượng hình thật. Các mặt của hình lập phương có mầu như màu nền, và những gì bạn thấy là các cạnh trước của đối tượng. Một hình lập phương thực sẽ có một vài mầu và bề mặt của nó. Ví dụ một hình lập phương bằng gỗ thì ta sẽ thấy màu và độ rám của gỗ. Trên máy tính nếu chúng ta tô tất cả trong một mầu và biểu diễn bằng hai chiều thì ta sẽ thấy như hình 2.10



Hình 2.10: Hình lập phương có mầu và không có shading

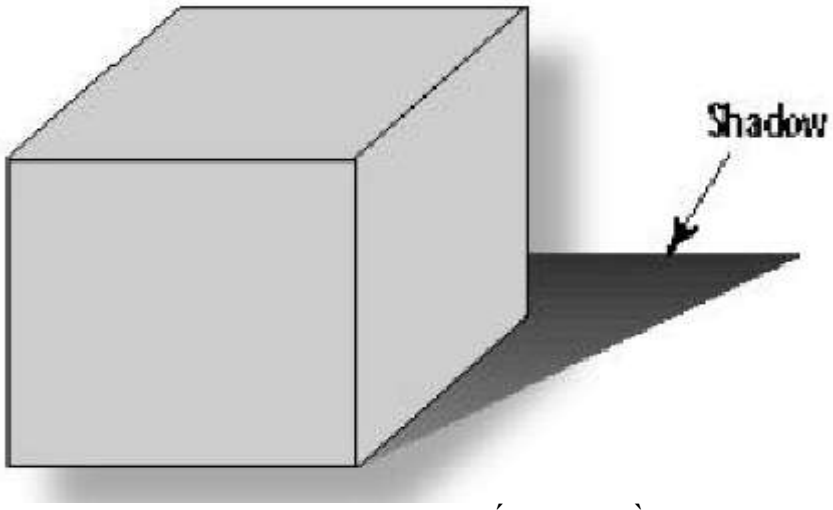
Bây giờ chúng ta sẽ quay lại với đối tượng hai chiều nhưng chúng ta sẽ vẽ rõ các cạnh bằng một màu khác. Để có thể thu được một phối cảnh đối tượng rắn của chúng ta thì chúng ta hoặc tạo ra ba mặt nhìn thấy với màu khác nhau hoặc cho chúng cùng màu nhưng đánh bóng để có ảo giác về ánh sáng. Như hình 2.11.



Hình 2.11: Hình lập phương với đánh bóng khác nhau trên 3 mặt

2.6. Light và shadows

Một phần tử cuối cùng không thể không nói đến là ánh sáng (Lighting). Bố trí ánh sáng có hai hiệu quả quan trọng trên đối tượng hiển thị ba chiều. Thứ nhất tạo bóng của bề mặt có mầu không thay đổi khi nhìn hay khi được chiếu sáng từ một góc. Thứ hai là là tạo bóng đổ cho đối tượng khi mà tia sáng bị chặn bởi đối tượng. Xem hình 2.12



Hình 2.12: Hình lập phương đặc được chiếu sáng bằng ánh sáng đơn

Hai nguồn sáng có thể ảnh hưởng đến một đối tượng ba chiều. Một là ánh sáng xung quanh, đơn giản nó là sự chiếu sáng thống nhất gây ra hiệu quả bóng trên các đối tượng màu đặc. Một nguồn sáng khác đó là đèn (lamp), được dùng để thay đổi bóng của các đối tượng đặc và cho hiệu quả bóng đổ.

2.7. Hệ thống toạ độ (Coordinate Systems)

Đến lúc này bạn đã hiểu được tại sao mắt có thể quan sát được ba chiều trên bề mặt hai chiều (như màn hình máy tính), bây giờ chúng ta sẽ xem xét cách vẽ các đối tượng này trên màn hình. Khi bạn vẽ các điểm, đường hay hình nào khác trên màn hình máy tính, bạn thường đưa ra một vị trí theo hàng và cột.

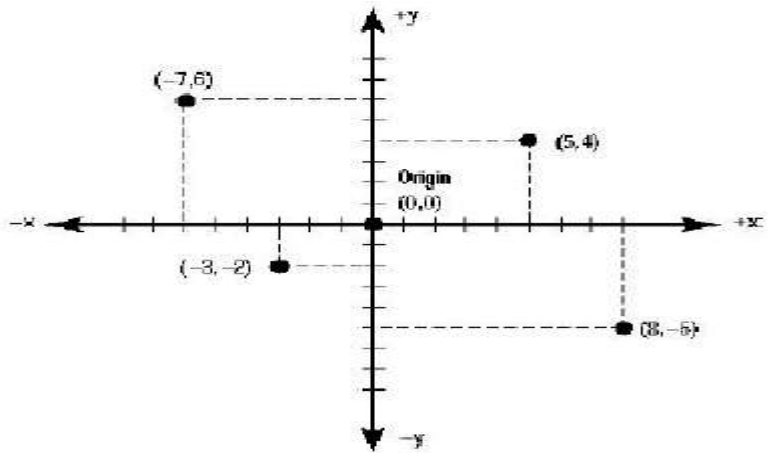
Ví dụ theo màn hình chuẩn VGA có 640 pixels từ trái sang phải và có 480 pixels từ trên xuống dưới. Để chỉ ra một điểm ở giữa màn hình bạn cho điểm đó trên vùng (320,240), đó là 320 pixels từ trái màn hình và 240 pixel xuống từ đỉnh màn hình.

Trong OpenGL, khi bạn tạo một cửa sổ để vẽ , bạn cũng phải chỉ ra hệ thống toạ độ mà bạn muốn sử dụng, và cách ánh xạ các toạ độ đưa ra lên các pixels màn hình vật lý. Trước tiên chúng ta sẽ thấy cách áp dụng điều này để vẽ hai chiều , sau đó mở rộng nguyên lý cho ba chiều.

2.7.1. Hệ toạ độ đề các 2D

Hệ thống toạ độ phổ biến nhất để vẽ đồ thị hai chiều là hệ toạ độ đề các. Hệ toạ độ đề các chỉ ra toạ độ x và toạ độ y, toạ độ x là phép đo theo chiều ngang và toạ toạ độ y là phép đo vị trí theo chiều đứng.

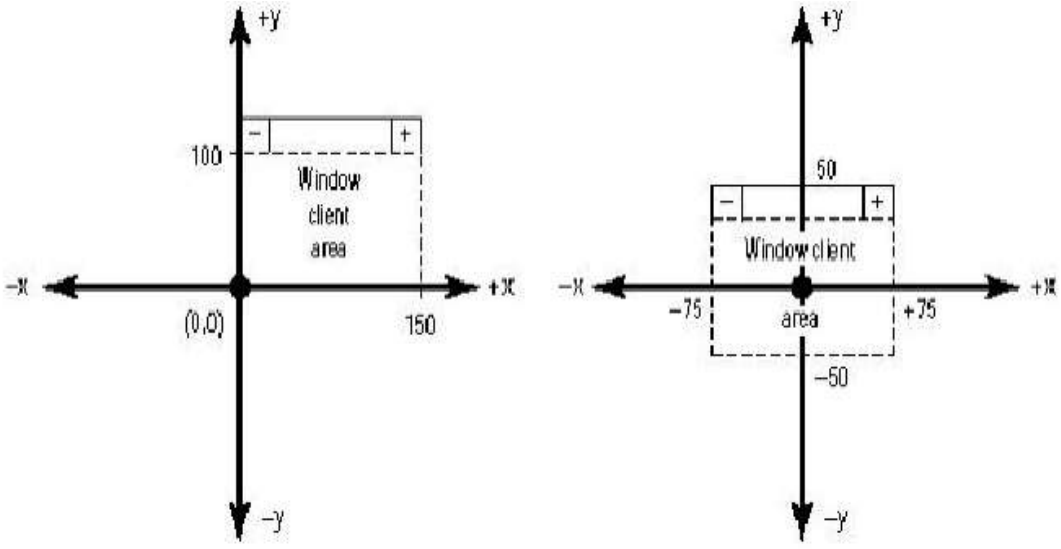
Gốc của hệ toạ độ là x=0,y=0. Một toạ độ đề các được viết trong ngoặc đơn với x trước y sau và cách nhau bởi dấu phẩy. Ví dụ gốc là (0,0). Hình 2.13 là một hệ toạ độ đề các hai chiều được sử dụng phổ biến trong phổ thông. Hiện nay các chế độ Window khác nhau, do đó các toạ độ khi bạn vẽ sẽ được dịch khác nhau, tức là có một ánh xạ từ không gian toạ độ thực vào toạ độ window.



Hình 2.13: Mặt phẳng đề các

2.7.2. Coordinate clipping

Một cửa sổ được đo một cách vật lý theo pixels. Trước khi bạn vẽ các điểm, đường, và các hình bạn phải thông báo cho thư viện đồ hoạ (ví dụ OpenGL) biết cách chuyển đổi các phần toạ độ đã đưa ra thành toạ độ màn hình. Cái này được hoàn thành bằng cách chỉ ra vùng không gian đề các mà nó chiếm cửa sổ, vùng này được gọi là clipping area. Trong không gian hai chiều, clipping area là các giá trị x,y max và min bên trong cửa sổ. Một cách khác để chỉ ra điều này là đưa ra toạ độ vị trí gốc và mối quan hệ trong cửa sổ. Hình 2.14 đưa ra hai clipping area phổ biến.

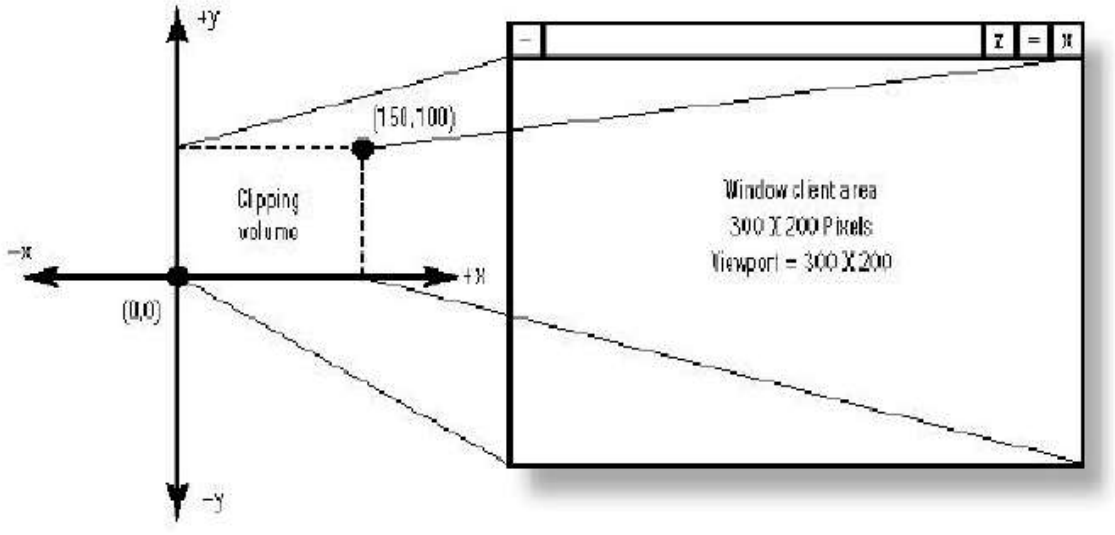


Hình 2.14 : Hai clipping areas

2.7.3. Cổng nhìn, cửa sổ của bạn đến 3D

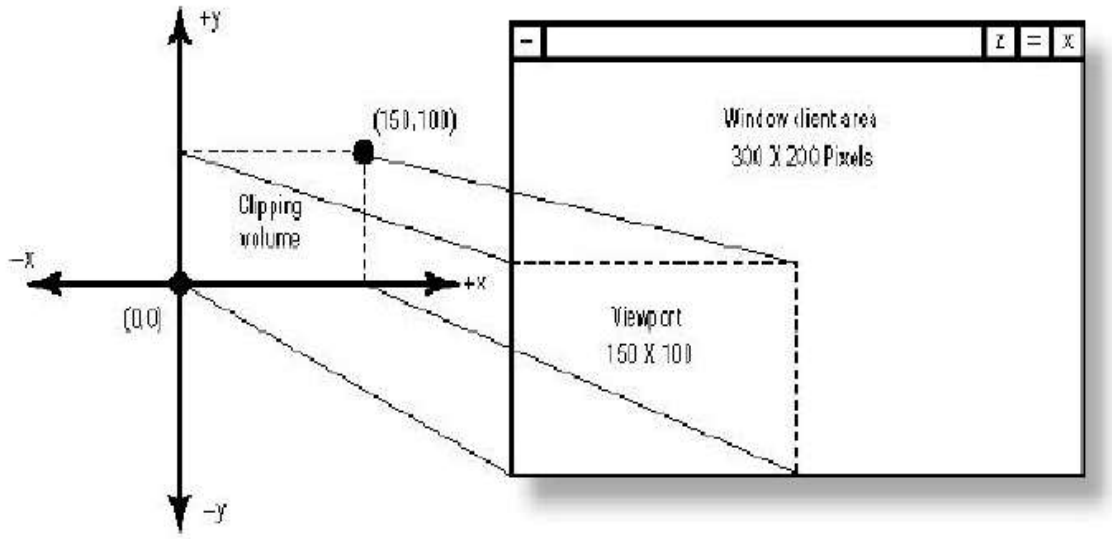
Rất ít khi chiều rộng và chiều cao của clipping area một cách chính xác phù hợp với chiều rộng và chiều cao của cửa sổ theo pixel. Do đó hệ thống toạ độ phải được ánh xạ từ toạ độ đề các logic vào toạ độ pixel màn hình vật lý. Phép ánh xạ này được chỉ ra bằng cách đặt cổng nhìn (Viewport). Cổng nhìn (Viewport) là vùng bên trong khu vực khách của cửa sổ (Window's client area) mà sẽ được sử dụng để vẽ clipping area. Đơn giản cổng nhìn ánh xạ clipping area vào một vùng của cửa sổ. Thường thì cổng nhìn được xác định là toàn bộ cửa sổ, nhưng trong một số trường hợp điều này thực sự không cần thiết, ví dụ bạn chỉ muốn vẽ một nửa dưới của cửa sổ.

Hình 2.15 hiển thị một một cửa sổ lớn kích thước 300x200 pixels với một cổng nhìn là toàn bộ khu vực client (client area). Nếu clipping area cho cửa sổ này đã được đặt là 0 đến 150 theo truc x và 0 đến 100 theo trục y, thì toạ độ logic sẽ được ánh xạ đến toạ độ màn hình lớn hơn trong cửa sổ nhìn. Mỗi lần tăng trong hệ thống toạ độ logic thì sẽ khớp với hai lần tăng trong hệ thống toạ độ vật lý của cửa sổ (theo pixel).



Hình 2.15: Cổng nhìn được xác định bằng 2 lần clipping area

Hình 2.16 cho một cổng nhìn bằng với clipping area, trong khi đó cửa sổ nhìn vẫn là 300x200 pixels. Điều này làm cho vùng nhìn (viewing area) sẽ nằm trong phần dưới bên trái của cửa sổ.



Hình 2.16: Cổng nhìn có cùng kích thước với clipping area

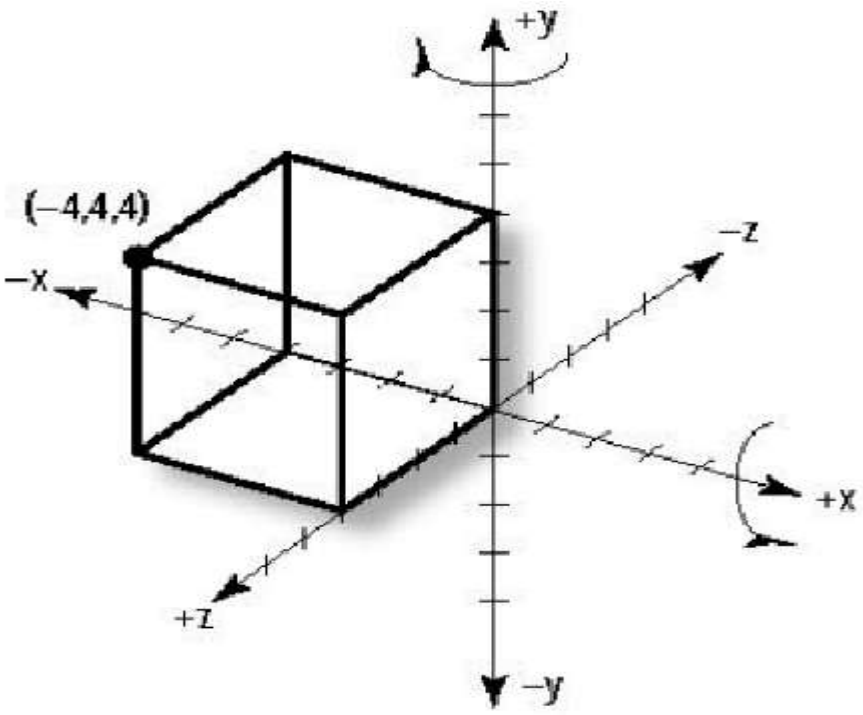
Bạn có thể sử dụng cổng nhìn để phóng to hoặc thu nhỏ ảnh trong cửa sổ window, và chỉ để hiển thi một phần của clipping area bằng cách đặt cổng nhìn lớn hơn client area của Window.

2.7.4. Vẽ hình cơ bản (Primitives)

Trong cả 2D và 3D khi bạn vẽ một đối tượng, thực chất là bạn đang dựng nó với các hình nhỏ hơn gọi là primitives. Primitives có bề mặt hai chiều như các điểm, đường, hình đa giác, nó được tập hợp lại trong không gian ba chiều để vẽ các đối tượng ba chiều. Ví dụ như hình lập phương ở hình 2.10 được tạo bởi 6 hình vuông hai chiều, mỗi cái được đặt ở một mặt riêng biệt. Mỗi góc của hình vuông hay bất kì một hình cơ bản (primitive) được gọi là đỉnh (vertex). Sau đó các đỉnh này được đặt để chiếm một không gian toạ độ cụ thể trong 2D hoặc 3D. Trong tất cả thư viện đồ hoạ đều có các hình cơ bản (OpenGL, DirectX…).

2.7.5. Toạ độ đề các 3D

Bây giờ chúng ta sẽ mở rộng hệ thống toạ độ 2D sang hệ thống toạ độ 3D và thêm một thành phần độ sâu. Hình 2.17 hiển thị một hệ thống đề các có thêm một trục mới đó là trục z. Trục z vuông góc với cả hai trục x và y. Bây giờ chúng ta sẽ chỉ ra một điểm trong toạ độ ba chiều với ba kích thước x,y,z. Trên hình là điểm (-4,4,4).



Hình 2.17: Toạ độ đề các ba chiều

2.8 Các phép biến đổi hình học 3 chiều

2.8.1 Hệ tọa độ thuần nhất

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

+ Ảnh của các đường thẳng song song là đường thẳng song song.
+ Bảo toàn tỷ lệ khoảng cách.
+ Tổ hợp các phép biến đổi có tính phân phối.

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

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

2.8.2 Phép tịnh tiến

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

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

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}.T(dx, dy, dz) = \begin{bmatrix} x+dx & y+dy & z+dz & 1 \end{bmatrix}$$

2.8.3 Phép tỷ lệ

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

$$X' = X.Ts$$

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}.Ts$$

$$= \begin{bmatrix} x.Sx & y.Sy & z.Sz & 1 \end{bmatrix}$$

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

2.8.4 Phép biến dạng

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

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

$$\bar{X}' = \bar{X}.\bar{Tsh}$$

$$\bar{x}' \quad \bar{y}' \quad \bar{z}' \quad 1 = \bar{x} \quad \bar{y} \quad \bar{z} \quad 1.\bar{Tsh}$$

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

2.8.5 Phép quay 3 chiều

- Quay quanh trục Oz
- $$\bar{Tz} = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Quay quanh trục Ox

$$\bar{Tx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Quay quanh trục Oy

$$\bar{Ty} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.8.6 Phép đối xứng

- Qua mặt phẳng tọa độ

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

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

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

- Qua các trục

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

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

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

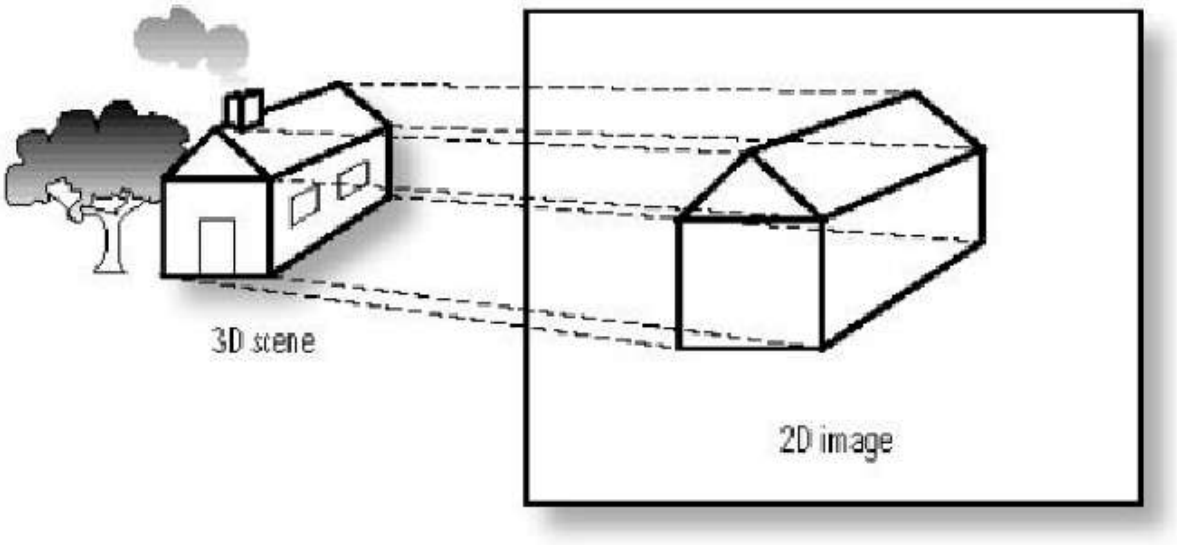
- Qua gốc tọa độ

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

2.9 Phép chiếu và bản chất của ba chiều

Bạn đã thấy cách chỉ ra một vị trí trong không gian ba chiều sử dụng toạ độ đề các. Không có vấn đề để thuyết phục mắt bạn, tuy nhiên các pixel trên màn hình chỉ có hai chiều. Cách OpenGL chuyển toạ độ đề các này thành toạ độ hai chiều vẽ được trên màn hình? Câu trả lời ngắn gọn là "Vận dụng lượng giác học và thao tác ma trận đơn". Thật may là bạn không phải giỏi toán học để có thể sử dụng OpenGL để vẽ đồ hoạ. Để có thể hiểu sâu hơn về vấn đề này các bạn có thể đọc ở chương 7 của "OpenGL Supper Bible".

Thực sự tất cả những gì bạn cần hiểu là khái niệm phép chiếu (projection). Các toạ độ 3D sẽ được chiếu lên bề mặt 2D (nền cửa sổ). Nó giống như ta chèn phác thảo một số đối tượng đằng sau tấm kính với một dấu đen. Khi lối tượng đi qua hoặc bạn di chuyển cái kính, bạn vẫn có thể nhìn thấy hình dáng bên ngoài của đối tượng với các cạnh góc của nó. Trong hình 2.18 một ngôi nhà được vạch ra trên một miếng kính phẳng. Bằng cách chỉ ra phép chiếu(projection) bạn đưa ra một clipping volume(nhớ lại clipping area) mà bạn muốn được hiển thị trên cửa sổ và cách nó sẽ được dịch.



Hình 2.18: Một ảnh được chiếu lên bề mặt 2D

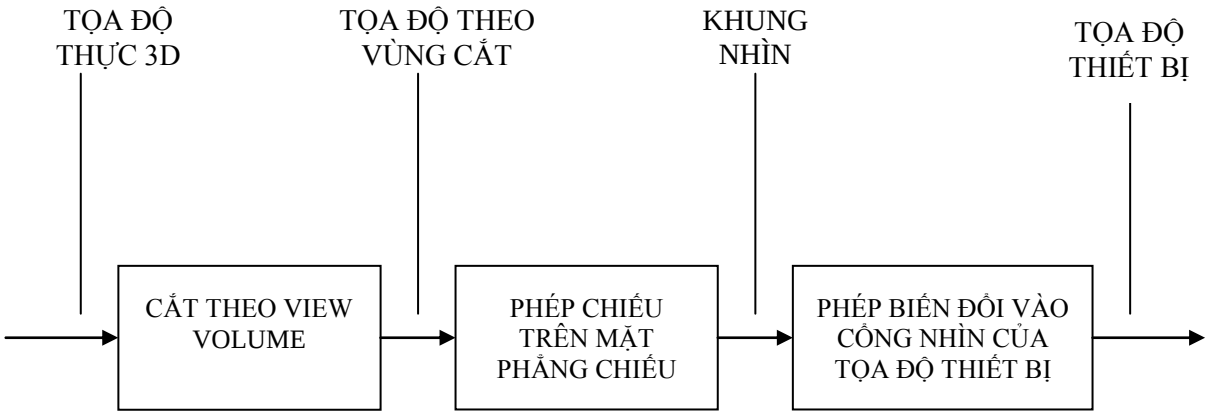
2.9.1 Phép chiếu

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

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

Các bước xây dựng hình chiếu

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



Hình 2.19: Mô hình nguyên lý của tiến trình biểu diễn đối tượng 3D

2.9.2 Phép chiếu song song (Parallel Projections)

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

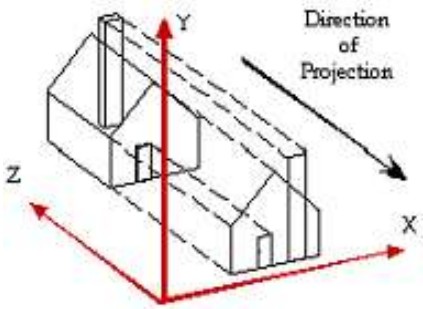
Phân loại phép chiếu song song dựa trên hướng của tia chiếu (Direction Of Projection) và mặt phẳng chiếu (projection plane).

2.9.2.1 Phép chiếu trực giao (Orthographic projection)

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

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

$$Ty = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Tx = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Tz = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Hình 2.20: Phép chiếu trực giao

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

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

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

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

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

2.9.2.2 Phép chiếu trục lượng (Axonometric)

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

Có 3 phép chiếu

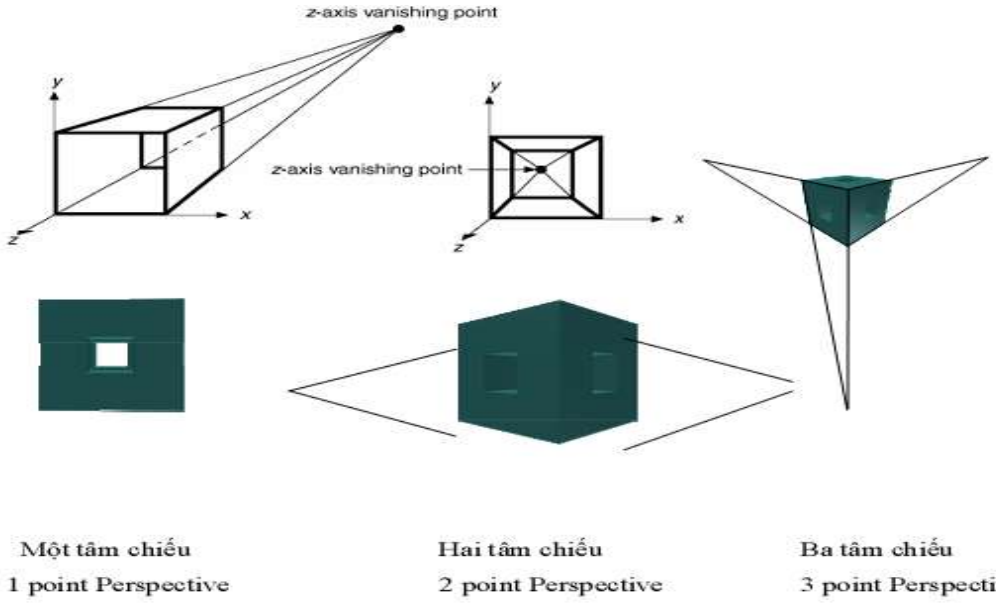
- Phép chiếu Trimetric
- Phép chiếu Dimetic
- Phép chiếu Isometri

2.9.3 Phép chiếu phối cảnh (Perspective Projection)

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

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

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



Hình 2.21: Phép biến đổi phối cảnh

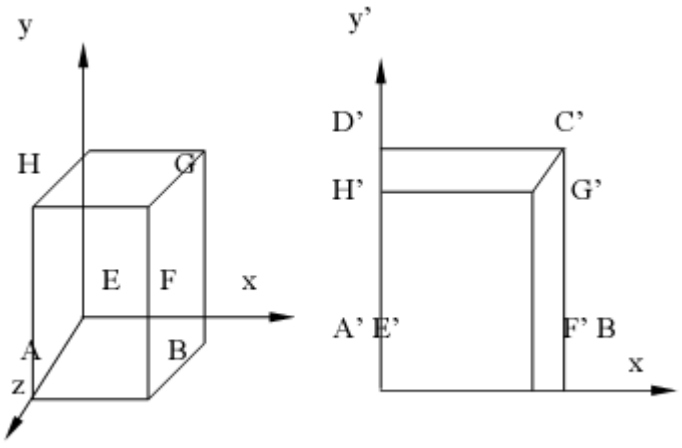
2.9.3.1 Phép chiếu phối cảnh một tâm chiếu

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

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

Phương trình biến đổi:

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



Hình 2.22 : Phép chiếu phối cảnh một tâm chiếu

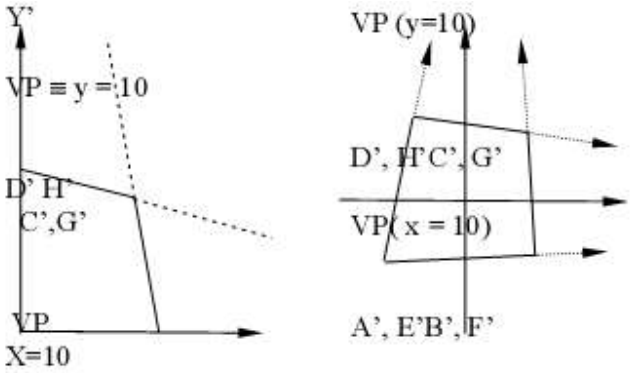
- Ma trận biến đổi một điểm phối cảnh [Tr] có dạng:

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

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

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

2.9.3.2 Phép chiếu phối cảnh hai tâm chiếu



Hình 2.23: Phép chiếu phối cảnh hai tâm chiếu

$$T_c = T_{pq} \cdot T_z$$

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

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

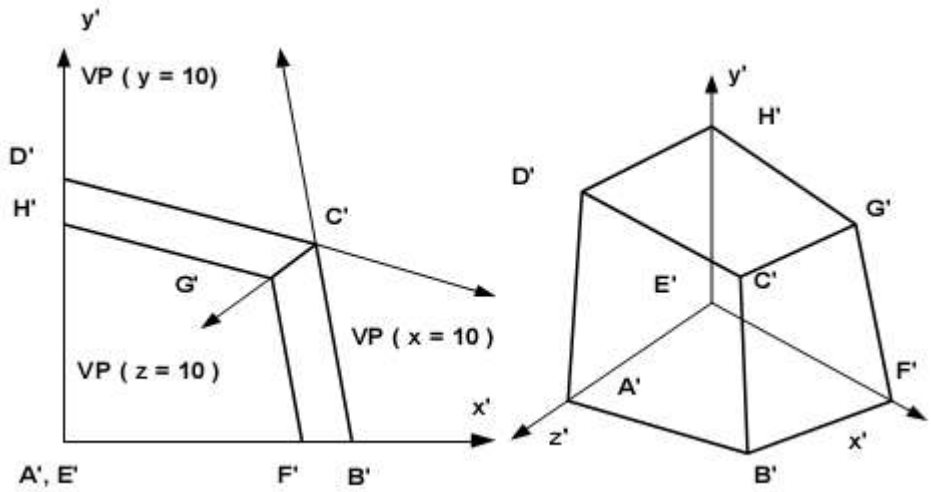
$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & px+qy+1 \end{bmatrix}$$

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

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

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

2.9.3.3 Phép chiếu phối cảnh ba tâm chiếu



Hình 2.24 : Phép chiếu phối cảnh ba tâm chiếu

$$T_{pqr} = T_p \cdot T_q \cdot T_r$$

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

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

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

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

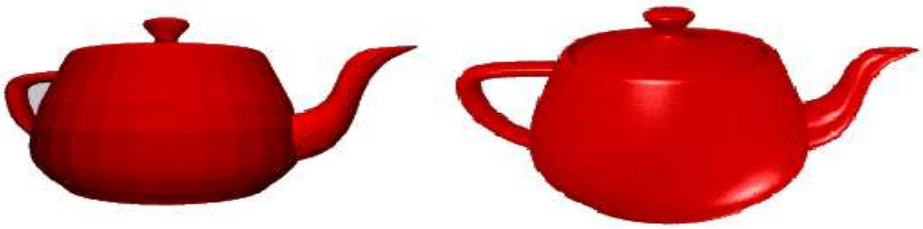
Ba tâm chiếu: trên trục x tại điểm [-1/p 0 0 1], y tại điểm [0 -1/q 0 1] và z tại điểm [0 0 -1/r 1]

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

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

2.10 Chiếu sáng và tô bóng

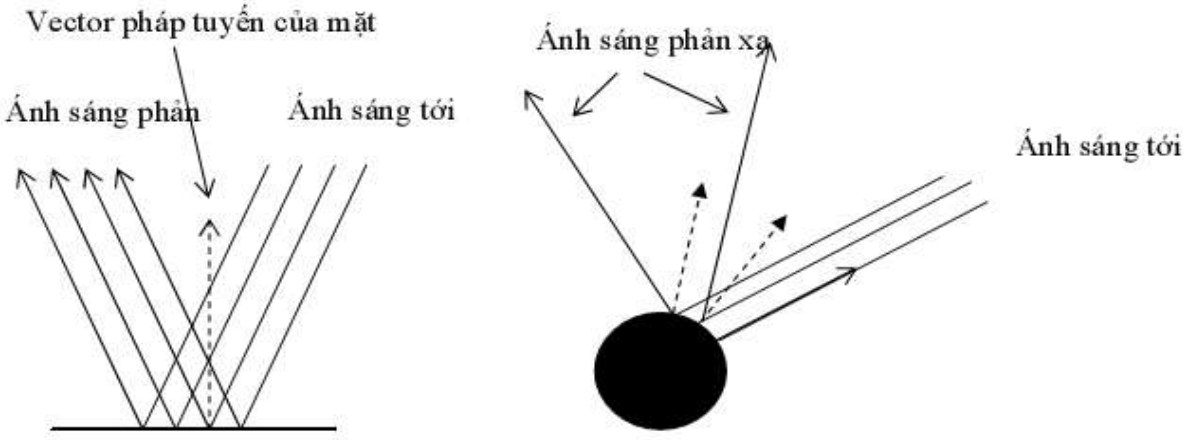
Khi biểu diễn các đối tượng 3 chiều, một yếu tố không thể bỏ qua để tăng tính thực của đối tượng đó là tạo bóng sáng cho vật thể. Để thực hiện được điều này, chúng ta cần phải lần lượt tìm hiểu các dạng nguồn sáng có trong tự nhiên, cũng như các tính chất đặc trưng khác nhau của mỗi loại nguồn sáng.



Hình2.25: Hình được chiếu sáng

2.10.1 Nguồn sáng xung quanh

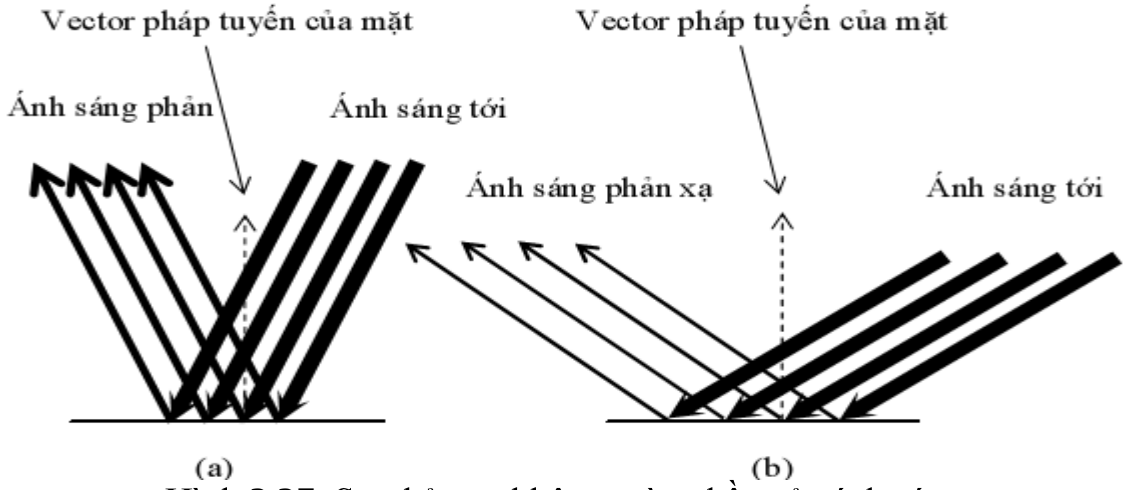
Ánh sáng xung quanh là mức ánh sáng trung bình, tồn tại trong một vùng khôn gian. Một không gian lý tưởng là không gian mà tại đó mọi vật đều được cung cấp một lượng ánh sáng lên bề mặt là như nhau, từ mọi phía ở mọi nơi. Thông thường ánh sáng xung quanh được xác định với một mức cụ thể gọi là mức sáng 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 2.26: Sự phản xạ của ánh sáng

2.10.2 Nguồn sáng định hướng

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



Hình 2.27: Sự phản xạ không toàn phần của ánh sáng

Trong hình 2.27 thể hiện sự phản xạ ánh sáng không toàn phần. Độ đậm nét của các tia ánh sáng tới thể thiện cường độ sáng cao, độ mảnh của các tia phản xạ thể hiện cường độ sáng thấp. Nói chung, khi bề mặt là không phản xạ toàn phần thì cường độ của ánh sáng phản xạ luôn bé hơn so với cường độ của ánh sáng tới, và cường độ của tia phản xạ còn tỷ lệ với góc giữa tia tới với vector pháp tuyến của bề mặt, nếu góc này càng nhỏ thì cường độ phản xạ càng cao. Ở đây ta chỉ quan tâm đến thành phần ánh sáng khuếch tán và tạm bỏ qua hiện tượng phản xạ toàn phần.

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

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

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

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

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

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

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

2.10.3 Nguồn sáng điểm

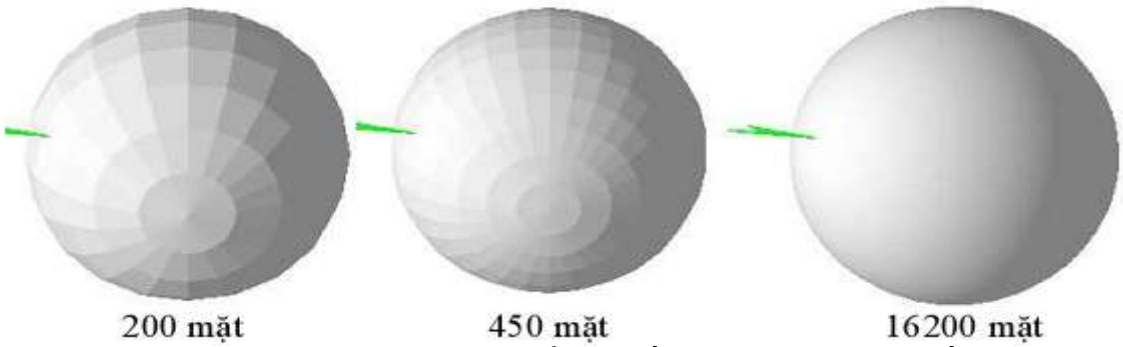
Nguồn sáng định hướng là tương đương với nguồn sáng điểm đặt ở vô tận. Nhưng khi nguồn sáng điểm được mang đến gần đối tượng thì các tia sáng từ nó phát ra không còn song song nữa mà được tỏa ra theo mọi hướng theo dạnh hình cầu. Vì thế, các tia sáng sẽ rơi xuống các điểm trên bề mặt dưới các góc khác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

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

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

2.10.4 Mô hình bóng Gouraud

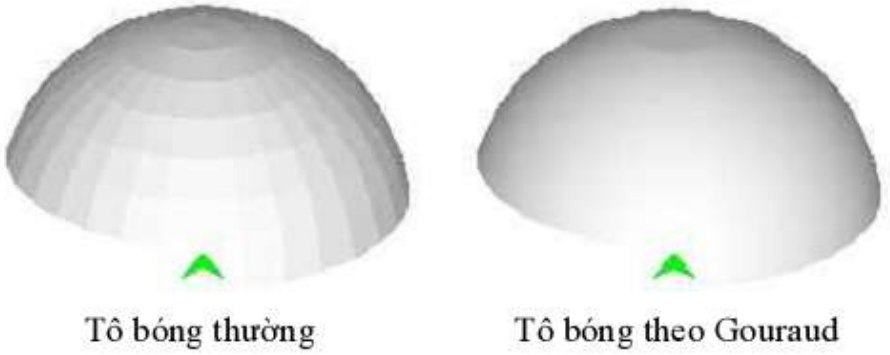
Mô hình bóng Gouraud là một phương pháp vẽ bóng, tạo cho đối tượng 3D có hình dáng cong có một cái nhìn có tính thực hơn. Phương pháp này đặt cơ sở trên thực tế sau: đối với các đối tượng 3D có bề mặt cong thì người ta thường xấp sỉ bề mặt cong của đối tượng bằng nhiều mặt đa giác phẳng, ví dụ như một mặt cầu có thể xấp sỉ bởi một tập các mặt đa giác phẳng có kích thước nhỏ sắp xếp lại, khi số đa giác xấp xỉ tăng lên thì tính thực của mặt cầu sẽ tăng, sẽ cho ta cảm giác mặt cầu trông tròn trịa hơn, mịn và cong hơn. Tuy nhiên, khi số đa giác xấp xỉ một mặt cong tăng thì khối lượng tính toán và lưu trữ cũng tăng theo tỷ lệ thuận theo số mặt, điều đó dẫn đến tốc độ thực hiện sẽ trở nên chậm chạp hơn. Vấn đề thứ 2 nảy sinh là khi ta phóng lớn hay thu nhỏ vật thể. Nếu ta phóng lớn thì rõ ràng các đa giác cũng được phóng lớn theo cùng tỷ lệ, dẫn đến hình ảnh về các mặt đa giác lại hiện rõ và gây ra cảm giác không được trơn mịn. Ngược lại, khi ta thu nhỏ thì nếu số đa giác xấp xỉ lớn thì sẽ dẫn đến tình trạng các đa giác nhỏ, chồng chất lên nhau không cần thiết.



Hình 2.28: So sánh vật thể với số mặt đa giác tăng dần

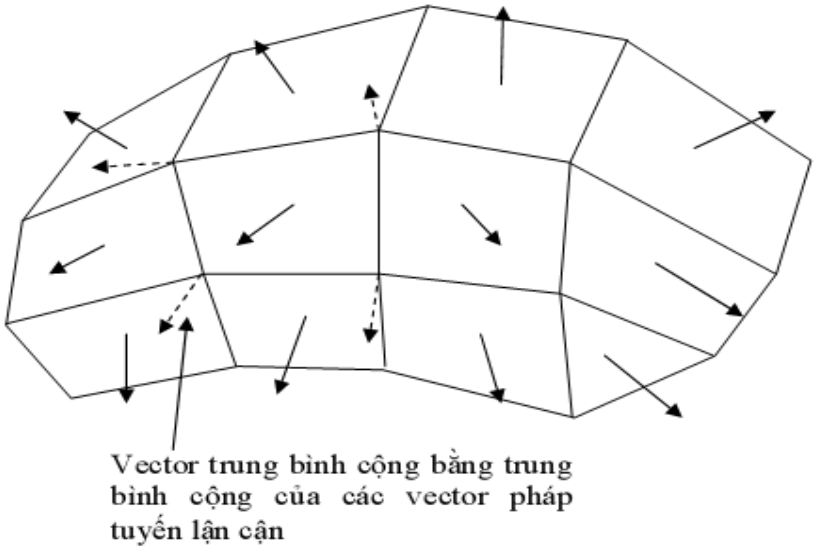
Để giải quyết vấn đề trên, chúng ta có thể tiến hành theo phương pháp tô bóng Gouraud. Mô hình bóng Gouraud tạo cho đối tượng một cái nhìn giống như là nó có nhiều mặt đa giác bằng cách vẽ mỗi mặt không chỉ với một cường độ sáng mà vẽ với nhiều cường độ sáng khác nhau trên các vùng

khác nhau, làm cho mặt phẳng nơm như bị cong. Bởi thực chất ta cảm nhận được độ cong của các mặt cong do hiệu ứng ánh sáng khi chiếu lên mặt, tại các điểm trên mặt cong sẽ có vector pháp tuyến khác nhau nên sẽ đón nhận và phản xạ ánh sáng khác nhau, từ đó ta sẽ cảm nhận được các độ sáng khác nhau trên cùng một mặt cong.



Hình 2.29: So sáng tô bóng thường và tô bóng Gouraud

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



Hình 2.30. Mô tả vector trung bình cộng của các mặt

CHƯƠNG 3: TỔNG QUAN VỀ SILVERLIGHT

3.1 Sự ra đời của Silverlight.

Ngày nay khi phát triển các ứng dụng trên Web các doanh nghiệp phần mềm thường đau đầu với những khó khăn về sự hỗ trợ trình duyệt và hệ điều hành. Điều họ muốn là với những ngôn ngữ và công cụ phát triển vốn đã quen thuộc từ trước tới giờ đều có thể làm cho họ những ứng dụng chạy tốt trên mọi trình duyệt. Silverlỉgt ra đời như một công nghệ phù hợp cho phép họ làm được những việc như thế. Nếu bạn đã quen thuộc với công nghệ .Net Framework thì khi tiếp cận với Silverlight bán sẽ tiết kiệm được rất nhiều thời gian và chi phí cho công nghệ Web mới.

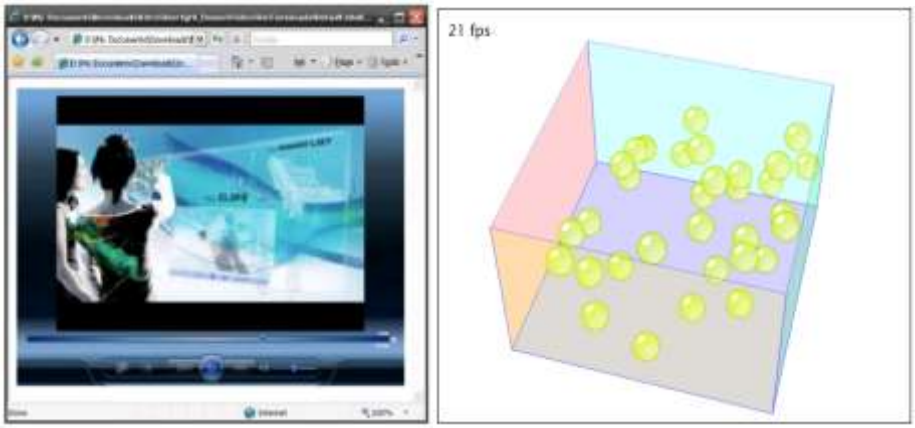
Các công nghệ plug-in trên Web trước đây không cho phép bạn truyền tải những dữ liệu hình ảnh chất lượng cao như 720p HDTV. Trong khi chất lượng đường truyền mạng ngày nay đang ngày càng tốt hơn và nhu cầu được xem những video chất lượng cao cũng tăng lên thì sự ra đời của Silverlight đã đem đến cho người đam mê thế giới đa phương tiện một sự thỏa mãn thật sự.

Chính vì vậy Silverlight ra đời, trở thành một đối thủ nặng ký của Adobe Flash. Ra đời sau, Silverlight thừa hưởng tính ưu việt của các công nghệ hiện có, nhỏ gọn, đa nền tảng, bộ công cụ phát triển mạnh mẽ và hoàn chỉnh, hơn hết là được phát triển bởi Microsoft – ông trùm số một trong thế giới phần mềm.

3.2 Định nghĩa Silverlight

Silverlight là một dạng plug-in dựa trên công nghệ của Microsoft .Net, nó độc lập với đa nền tảng và đa trình duyệt, nó cho phép phát triển các ứng dụng trên Web. Silverlight cung cấp một mô hình lập trình mềm dẻo và đồng nhất, nó hỗ trợ Ajax, Python, Ruby và các ngôn ngữ lập trình .Net như Visual Basic, C#.

Khả năng đa phương tiện của Silverlight thể hiện ở mức độ truyền tải âm thanh và hình ảnh chất lượng cao một cách nhanh chóng và hiệu quả trên tất cả các trình duyệt chính như Internet Explorer, Firefox, Safari.



Hình 3.1 : Ứng dụng của Silverlight

Với việc sử dụng Expression Studio và Visual Studio, các nhà phát triển và thiết kế co thể hợp tác một cách hiệu quả hơn bằng cách sử dụng chính kỹ năng của họ có hiện nay để làm phát triển các sản phẩm Web tương lai "Light up the Web".

3.3 Đặc tính của Silverlight

Silverlight kết hợp nhiều công nghệ vào một nền tảng phát triển, nó cho phép bạn được lựa chọn nhiều công cụ và ngôn ngữ lập trình thích hợp để giải quyết bài toán của bạn.

3.3.1 Sự kết hợp của WPF và XAML

Silverlight là một gói nhỏ của công nghệ Windows Presentation Foundation (WPF). Nó được mở rộng nhiều hơn các các Element trong trình duyệt để tạo ra giao diện người dùng. PWF cho phép bạn tạo ra đồ họa 3 chiều, hình ảnh động, đa phương tiện và nhiều tính năng phong phú khác trên máy khách. XAML (Extensible Application Markup Language) cung cấp các cú pháp đánh dấu đặc trưng cho công việc tạo các Element.

3.3.2 Mở rộng cho ngôn ngữ kịch bản

Silverlight cung cấp việc mở rộng cho các ngôn ngữ kịch bản (Javascript) ở một số các trình duyệt phổ thông để thực hiện việc trình bày giao diện và thao tác người dùng một cách phong phú hơn.

3.3.3 Sự tích hợp với các ứng dụng đã có

Silverlight tích hợp liền mạch với ngôn ngữ Javascript và mã Ajax của ASP.Net để bổ sung các chức năng bạn đã xây dựng được. Bạn có thể tạo những tài nguyên trên nền máy chủ có trong ASP.Net và sử dụng các khả năng của Ajax trong ASP.Net để tương tác với tài nguyên trên nền máy chủ đó mà không làm gián đoạn người dùng

3.3.4 Sử dụng mô hình ngôn ngữ lập trình trên nền tảng .Net Framework và các công cụ để kết hợp.

Bạn có thể tạo các ứng dụng trên nền tảng Silverlight và sử dụng các ngôn ngữ động như InronPython cũng như các ngôn ngữ lập trình C# và Visual Bassic. Bạn có thể sử dụng các công cụ phát triển như Visual Studio để tạo các ứng dụng trên nền tảng Silverlight.

3.3.5 Hỗ trợ mạng

Silverlight bao gồm các hỗ trợ cho HTTP qua TCP. Bạn có thể kết nối tới các dịch vụ của WCF, SOAP, hoặc ASP.NET AJAX và nhận về các định dạng theo cấu trúc XML, JSON hay dữ liệu RSS.

3.3.6 Hỗ trợ ngôn ngữ tích hợp truy vấn

Điều này cho phép bạn truy cập dữ liệu bằng cách sử dụng cú pháp trực quan tự nhiên và mạnh mẽ, được gõ bởi các đối tượng có trong các ngôn ngữ .Net Framework.

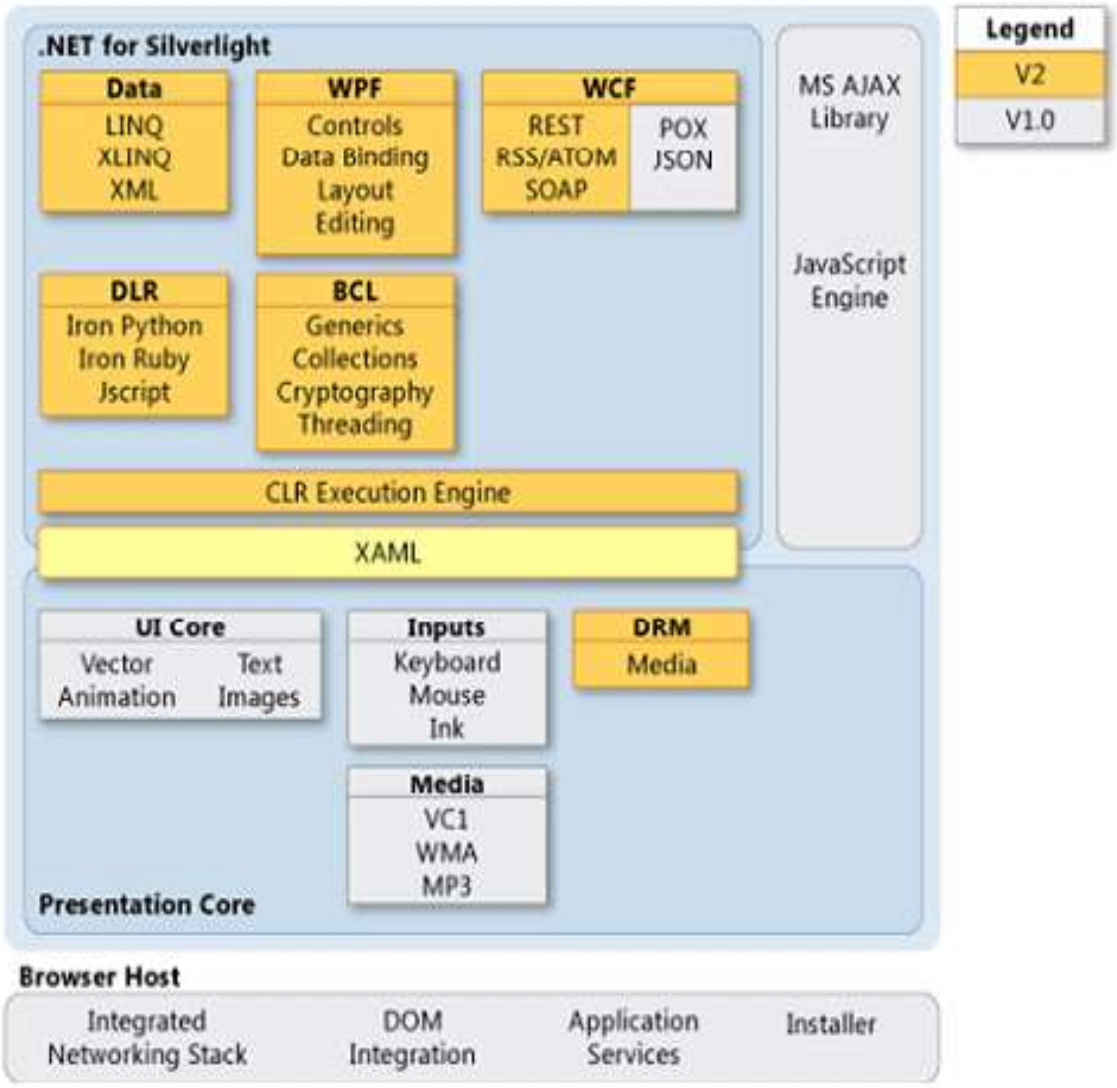
3.4 Kiến trúc tổng thể và các mô hình lập trình của Silverlight

3.4.1 Kiến trúc và các thành phần

Về cơ bản Silverlight là một nền tảng thống nhất của nhiều thành phần khác nhau. Tuy nhiên tôi nhóm lại các thành phần chính của Silverlight vào bảng dưới đây.

Thành phần	Diễn tả
Nền tảng trình bày cơ sở	Các thành phần dịch vụ hướng tới giao diện người dùng và tương tác người dùng, bao gồm các coltrol cho dữ liệu người dùng nhập, thiết bị đa phương tiện, quản lý phân quyền số, trình bày dữ liệu, đồ hoạ vector, chữ, hình ảnh động, cũng bao gồm XAML để đặc tả việc bố trí giao diện.
.Net Framework	Là một gói nhỏ trong .Net Framework, bao gồm các thành phần và cả thư viện, kể cả việc tương tác dữ liệu, khả năng mở rộng các control, mạng, garbage collection, và CLR.
Cài đặt và cập nhập	Là thành phần để xử lý các tiến trình cài đặt làm sao để đơn giản hóa cho lần cài đặt đầu tiên, tiếp sau đó chỉ cung cấp cơ chế tự động cập nhật và tương tác ở mức thấp.

Dưới đây là hình ảnh mô tả những thành phần trong kiến trúc của Silverlight cùng với các thành phần và dịch vụ liên quan khác.



Hình 3.2: Thành phần kiến trúc của Slverlight

Tính năng	Mô tả
Dữ liệu vào (input)	Xử lý dữ liệu đầu vào từ các thiết bị phần cứng như bàn phím, chuột, bảng vẽ hoặc các thiết bị đầu vào khác.
Trình bày giao diện người dùng (UI Rendering)	Trình bày vector và các đồ họa ảnh bitmap, ảnh động, và văn bản.
Thiết bị nghe nhìn (Media)	Các tính năng phát và quản lý một vài thể loại flie âm thanh và hình ảnh như .WMP và MP3
Controls	Hỗ trợ mở rộng cho các control để có khả năng tùy chỉnh về kiểu dáng và khuôn mẫu
Xếp đặt Layout	Cho phép khả năng xếp đặt vị trí động các thành phần giao diện người dùng
Trình bày dữ liệu (DATA Binding)	Cho phép việc kết nối dữ liệu của các đối tượng và các thành phần giao diện người dùng
DRM	Khả năng Quản lý phân quyền số
XAML	Cung cấp trình phân tách cho XAML

Các lập trình viên có thể thao tác với thành phần nền tảng trình bày cơ sở trên đây bằng cách sử dụng XAML để đặc tả. XAML là một yếu tố quan trọng trong việc tương tác giữa .NetFramwork và các kiểu trình bày Layout, ngoài ra các lập trình viên cũng có thể sử dụng cơ chế quản lý code bên trong để thao tác với trình bày.

.NetFrame work for silverlight:

Tính năng	Mô tả
Data	Hỗ trợ ngôn ngữ truy vấn tích hợp(LINQ) và LINQ với đặc tả XML, dễ dàng xử lý việc tích hợp và làm việc với dữ liệu từ nhiều nguồn khác nhau. Hỗ trợ việc sử dụng XML và các lớp biến đổi hóa(serialization) để xử lý dữ liệu
Base class library	Thuộc thư viện của .NetFramework, nó cung cấp các chức năng lập trình chủ yếu cho việc xử lý chuỗi, biểu thức chính quy, đầu vào và đầu ra, ánh xạ, tập hợp và toàn cục hóa.
Window Communication Foundation (WCF)	Cung cấp các tính năng để đơn giản hóa việc truy cập dữ liệu từ xa. Cơ chế này bao gồm một đối tượng trình duyệt, HTTP request và Response, RSS, JSON, POX, và các SOAP.
Common language Runtime (CLR)	Cung cấp việc quản lý bộ nhớ, dọn dẹp bộ nhớ thừa, xử lý ngoại lệ….
Windows Presentation Foundation controls (WPF)	Cung cấp các control giàu tính năng như Button, Calendar, checkbox, DataGrid, DatePicker, HyperlinkButton, RadioButton, và ScrollViewer.
Dynamic language Runtime (DLR)	Hỗ trợ việc thi hành các tính năng động của các ngôn ngữ kịch bản như Javascript và IronPython cho các chương trình trên nền tảng Silverlight.

3.4.2 Các mô hình lập trình của silverlight

Ở phiên bản Silverlight 1.0 cung cấp cho bạn duy nhất một mô hình lập trình và Javascript API, cho đến phiên bản Silverlight 2.0 đã cung cấp cả hai mô hình lập trình là Managed API và Javascript API chỉ cho phép bạn gõ mã lệnh Javascript để tương tác với trình duyệt thì Managed API đã sử dụng được cơ chế làm việc của Common Language Runtime(CLR) và kể cả Dynamic Language Runtime(DLR) để biên dịch và thực thi chương trình code(C#, VB…) của bạn.

3.4.2.1 Javascript API

Trong một chương trình silverlight nhúng theo kiểu Javascript API, nó tải chỉ một trang XAML đơn lẻ thay vì tải một gói ứng dụng. Trang XAML này có thể bao gồm các tham chiếu URI từ những nguồn bên máy chủ khác như các đoạn video và hình ảnh. Silverlight nhúng sử dụng XAML để tạo một cây đối tượng cái mà bạn có thể thao tác lập trình với javascript lưu trữ bên trong một trang HTML

Javascript API không cung cấp một mô hình ứng dụng có khả năng hỗ trợ các ứng dụng tổ hợp với sự điều hướng bên trong. Tuy nhiên nó cho phép làm nhữnh kịch bản theo kiểu Splash screan. Bạn cũng có thể làm các sự điều hướng trong javascript API bằng cách tải lại trang XAML mới hoặc tải lại cả trang web đó trong trình duyệt.

3.4.2.2 Managed API

Trong lập trình silverlight theo kiểu Managed API, bạn có thể thao tác lập trình với cả file XAML và file code bên trong.

Khi một silverlight nhúng tải file XAML, nó sẽ tạo một cây mô hình cái mà bạn cũng có thể gõ bằng các mã lệnh bên trong (thường là C#, VB,…..).

3.5 Khả năng hỗ trợ trình duyệt, hệ điều hành và các công nghệ liên quan

3.5.1 Hỗ trợ của hệ điều hành và trình duyệt được mô tả ở bảng dưới đây

Operating system	Internet Explorer 7	Internet Explorer 6	Firefox 1.5, 2.x, and 3.x	Safari 2.x and 3.x
Windows Vista	có	—	có	—
Windows XP SP2	có	có	có	—
Windows XP SP3	có	có	có	—
Windows 2000	—	có	—	—
Windows Server 2003 (excluding IA-64)	có	có	có	—
Mac OS 10.4.8+ (PowerPC)	—	—	—	—
Mac OS 10.4.8+ (Intel-based)	—	—	có	Có

Hình 3.3: Bảng mô tả hỗ trợ trình duyệt và hệ diều hành

3.5.2 Các công nghệ và công cụ liên quan của Silverlight.

Microsoft Expression Blend: Sử dụng công cụ này bạn có thể tạo và thay đổi cách xắp xếp trình bày Layer của ứng dụng bằng cách thao tác đến canvas và control trong XAML, làm việc với các chức năng đồ họa, lập trình với các ngôn ngữ Javascript.

Visual Studio 2008: Visual Studio cung cấp các công cụ hiệu quả cho việc phát triển các ứng dụng có hỗ trợ thao tác code bên trong. Tất cả các phiên bản đã có của Visual Studio đều có khả năng hỗ trợ Silverlight. Tuy nhiên ở phiên bản mới này nó còn hỗ trợ các tính năng đặc biệt hơn như bao gồm khả năng Intelliense, debugging và các template cho việc tạo mới một ứng dụng Silverlight.

ASP.NET AJAX: Bao gồm tập các Control, service, và các thư viện cần thiết cho việc tạo và tương tác với nền ứng dụng web.

Microsoft ASP.NET 3.5 Extensions Preview: Công nghệ này cung cấp chức năng thêm để việc tăng cường các ứng dụng ASP.NET AJAX.

Nó bao gồm 2 control sử dụng hữu ích cho việc xây dựng nền tảng silverlight cũng như là một phần ứng dụng ASP.NET:

- ASP.NET MediaPlayer Server Control.
- ASP.NET Silverlight Server Control.

Internet server: Bao gồm IIS (Microsoft Internet Information Services), và Apache Web server.

Microsoft Windows Communication Foundation (WCF) services.

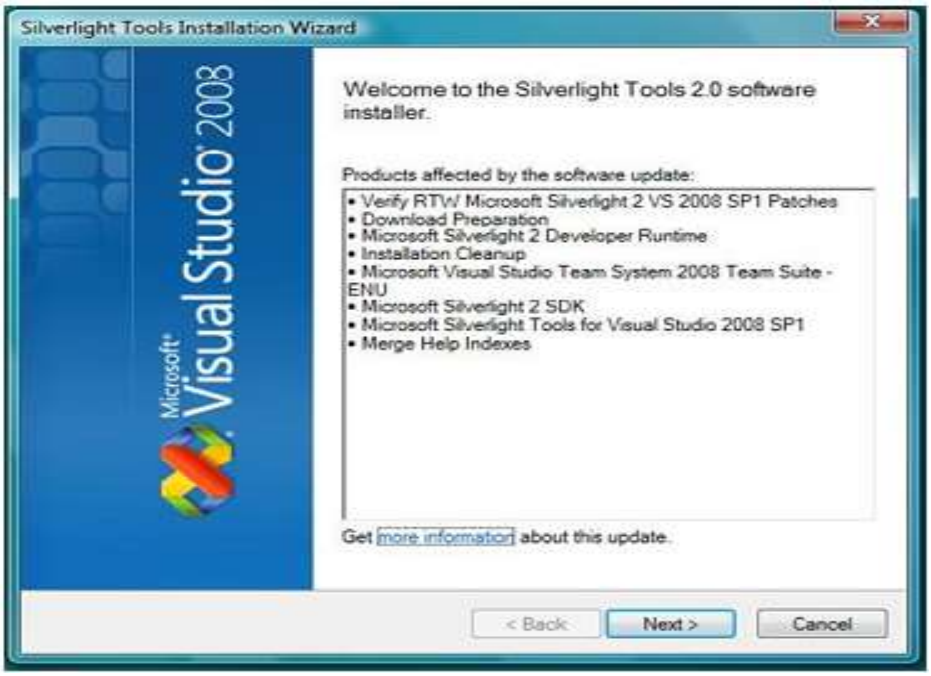
3.6 Hướng dẫn cài đặt và sử dụng công cụ Silverlight trên visual studio 2008

- Tải file Silverlight_tools.exe có trên trang:

http://www.microsoft.com/downloads/details.aspx?familyid=C22d6A7B-546F-4407-8FE6-D60C8EE221ED&displaylang=en

- Bạn phải chắc chắn rằng máy tính của bạn đã cài đặt Visual Studio 2008 SP1

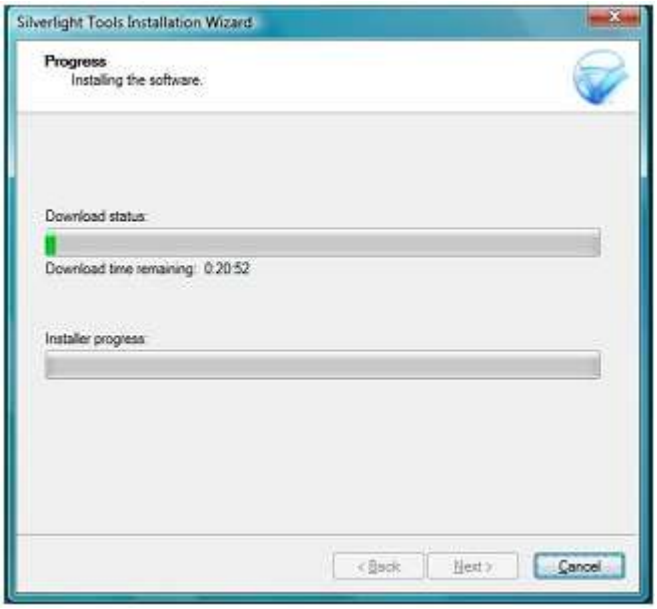
- Chạy file Silverlight_tools.exe, chờ khoảng 1 phút để hiện thị Silverlight tools installation Winzard



- Bấm next để đến bước 2, tích chọn " I have read an accept the license terms".

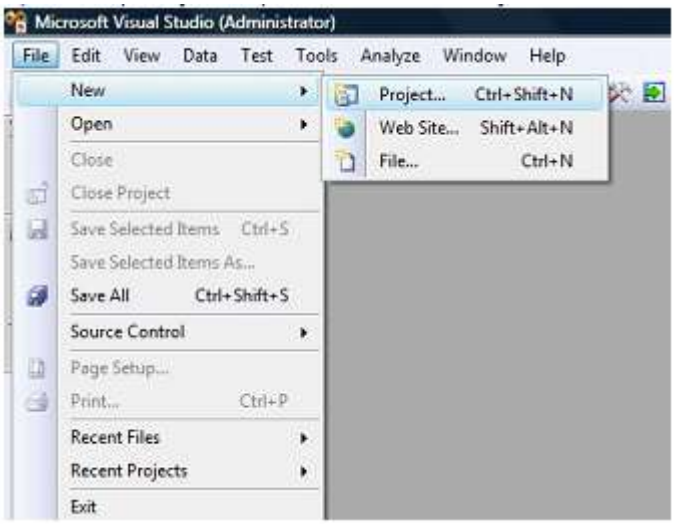
- Bấm next để hệ thống tự động kiểm tra tương thích (lưu ý:phải đóng hết các trình duyệt web)

- Để hệ thống cài đặt và hoàn thành

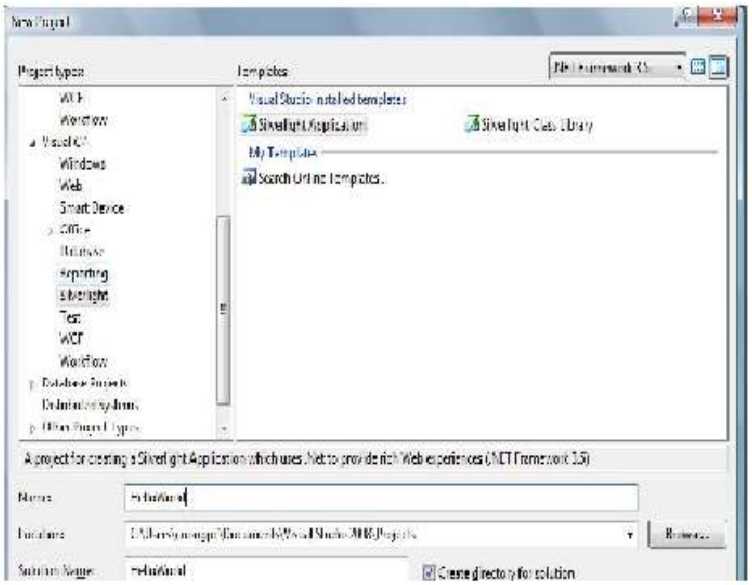


3.7 Ví dụ thực hành :Chương trình đầu tiên "Hello World"

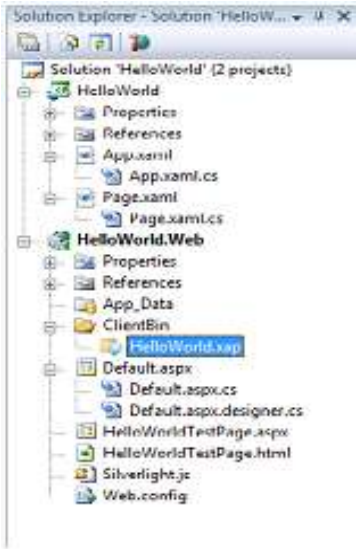
- Tạo mới một Project: chọn File → New → Project



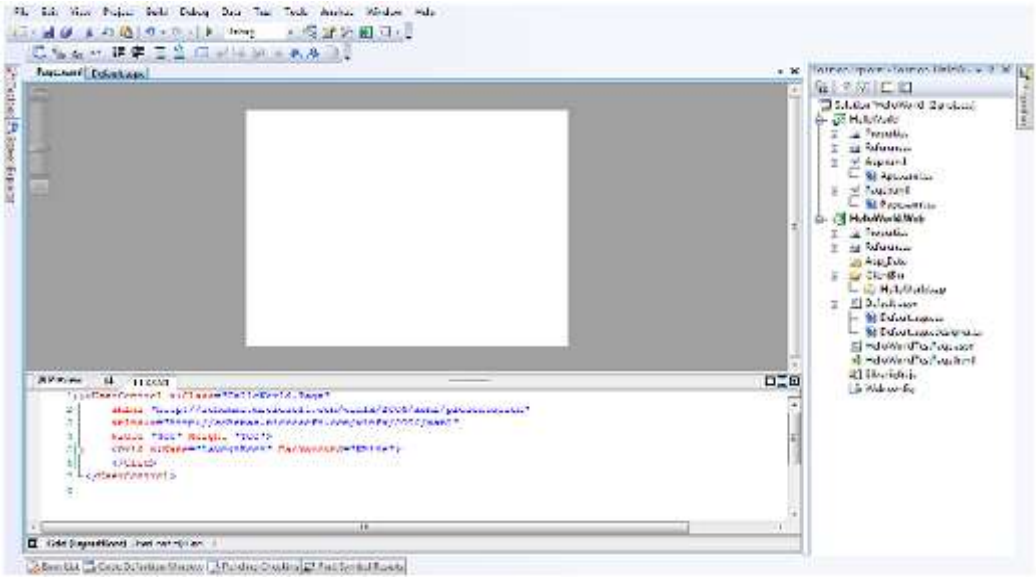
- Một cửa sổ mới "New Project" hiện ra. Chọn Visual C# (hoặc Visual basic) trong project types, chọn Silverlight. Phía bên phải cửa sổ cho phép bạn chọn các Template



- Chúng ta chọn SilverlightApplication trong Templates
- Đặt tên chương trình đầu tiên là "Hellworld", tùy chọn Location, bấm OK
- Bạn có thể chọn ProjectType theo mặc định trong hội thoại Add Silverlight application, bấm OK.
- Solution mới tạo ra với 2 project: Silverlightproject và web project (dung để nhúng silverlight tạo bởi silverlight project).



Trong thư mục ClientBin của web project (HelloWorld.Web) chứa ứng dụng silverlight được đóng gói dưới dạng file HelloWorld.xap của project silverlight(HelloWorld)



Toàn bộ màn hình ứng dụng đầu tiên của bạn được nhìn thấy như sau:

- Chúng ta làm 2 phương pháp một là viết code C# trong code ứng dụng, hai là viết trực tiếp trong XAML.

3.7.1 Viết chương trình bằng Code C#

- Trong file Page.xaml.cs chúng ta bắt đầu với việc tạo một nút theo những dòng lệnh dưới đây

```
// Khai bao button
    Button myButton;
    public Page()
    {
        InitializeComponent();
        // Khởi tạo button
        myButton = new Button();
        //Xác định các thuộc tính cho myButton
        myButton.Content = "Click Me";
        myButton.Height = 25;
        myButton.Width = 100;
        myButton.Margin = new Thickness(10, 10, 0, 0);
        //Đưa myButton vào LayoutRoot
        LayoutRoot.Children.Add(myButton);
```

- Để tạo sự kiện cho một nút chúng ta cần thêm những dòng lệnh sau vào

```
//thêm phương thức xử lý sự kiện cho myButton
 myButton.Click += new RoutedEventHandler(myButton_Click);
```

```
void myButton_Click(object sender, RoutedEventArgs e)
{
//Hiển thị thông điệp trên trình duyệt
    System.Windows.Browser.HtmlPage.Window.Alert("Hello Silverlight World!");
}
```

- Bấm F5 để chạy chương trình.

3.7.2 Viết chương trình bằng XAML

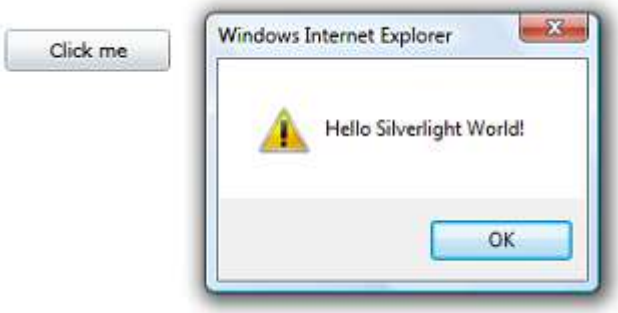
Lưu ý, với cùng project trên, muốn viết đặc tả bằng XAML tương đương ta cần xóa bỏ phần mã trình C# cũ đi, vì C# và XAML không thể cùng sinh một đối tượng.

Trong file Page.xaml ta thêm đoạn mã sau

```
<UserControl x:Class="HelloWorld.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button Name="myButton" Content="Click me" Width="100"Height="25"></Button>
    </Grid>
</UserControl>
```

- Gán sự kiện xử lý khi click

- Bấm F5 để chạy ứng dụng. Cả hai phương thức viết trên đều cho ra một kết quả như hình vẽ sau:



CHƯƠNG 4: XÂY DỰNG ALBUM ẢNH 3D BẰNG SILVERLIGHT

4.1 Giới thiệu ứng dụng

Silverlight 3 đã tung ra một loạt các tính năng mới có sẵn cho người dùng phát triển, và một trong đó sẽ cung cấp một cấp độ mới của thiết kế giao diện người dùng là việc giới thiệu quan điểm 3D. Tính năng này cho phép người dùng áp dụng cho bất kỳ UIElement để cung cấp cho sự hiển thị 3 chiều.

Trong chương này, sẽ xây dựng một FlipBook đơn giản cho phép người dùng lướt qua các hình ảnh như là đã được chuyển sang các trang trên một cuốn sách. Khi người dùng nhấp vào một hình ảnh trong trang sẽ "biến" để lộ những hình ảnh tiếp theo.

4.2 Đoạn mã xử lý chính

```
<UserControl x:Class="Projection3D.MainPage"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             mc:Ignorable="d"
             d:DesignWidth="640"
             d:DesignHeight="480">
  <UserControl.Resources>
    <Style x:Key="_imageStyle"
          TargetType="Image">
      <Setter Property="Margin"
              Value="275,0,0,0" />
      <Setter Property="Width"
              Value="275" />
      <Setter Property="Stretch"
              Value="Uniform" />
    </Style>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot">
```

```
    <Image Style="{StaticResource _imageStyle}"
           Source="Images/Enterprise-A.jpg"
           MouseLeftButtonUp="FlipImage" />
    <Image Style="{StaticResource _imageStyle}"
           Source="Images/Enterprise-B.jpg"
           MouseLeftButtonUp="FlipImage" />
    <Image Style="{StaticResource _imageStyle}"
           Source="Images/Enterprise-C.jpg"
           MouseLeftButtonUp="FlipImage" />
    <Image Style="{StaticResource _imageStyle}"
           Source="Images/Enterprise-D.jpg"
           MouseLeftButtonUp="FlipImage" />
    <Image Style="{StaticResource _imageStyle}"
           Source="Images/Enterprise-E.jpg"
           MouseLeftButtonUp="FlipImage" />
    <TextBlock Text="Click an image to turn the page."
               VerticalAlignment="Bottom"
               HorizontalAlignment="Center" />
  </Grid>
</UserControl>
```

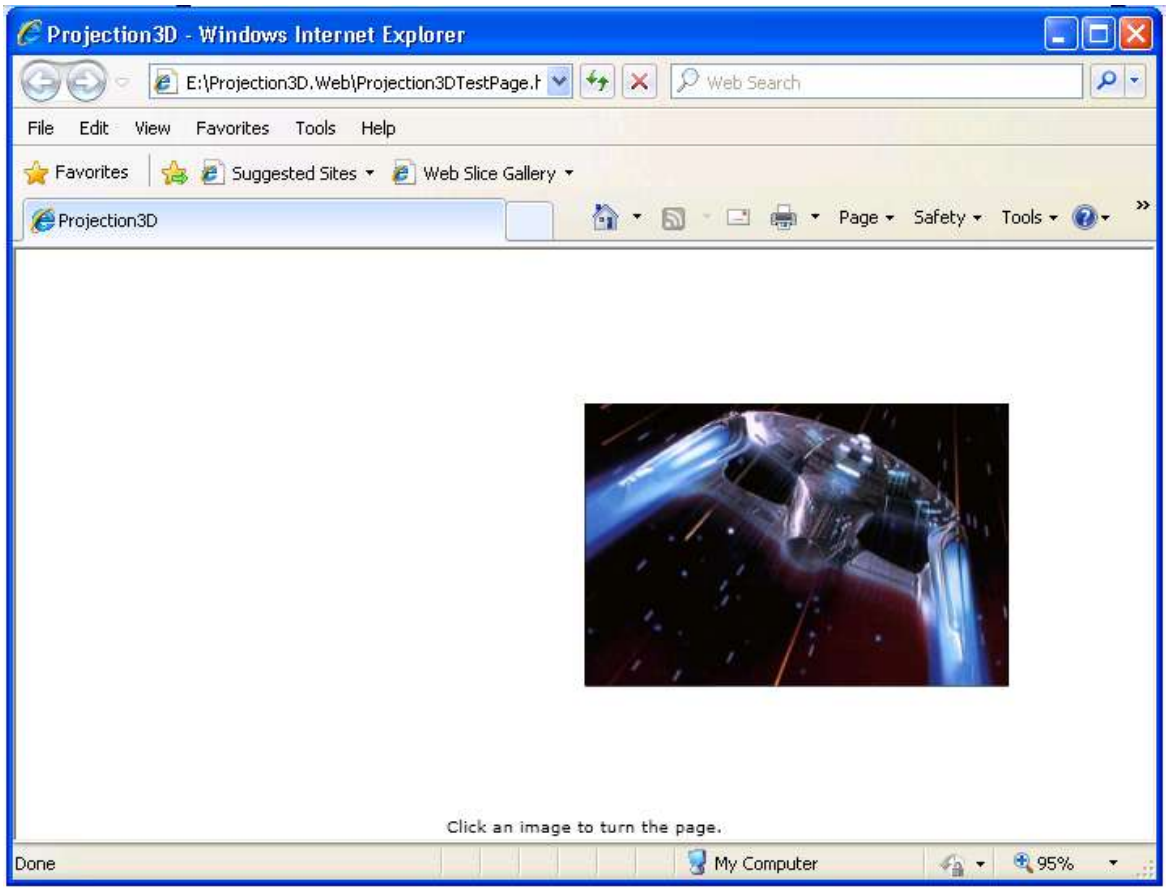
```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;

namespace Projection3D
{
  public partial class MainPage : UserControl
  {
    // Holds the current zIndex.  Used to make sure images
    // that are being flipped are on top of all other images.
    private int _zIndex = 10;

    public MainPage()
    {
      InitializeComponent();
    }

    private void FlipImage(object sender, MouseButtonEventArgs e)
```

```
    {
      Image image = sender as Image;

      // Make sure the image is on top of all other images.
      image.SetValue(Canvas.ZIndexProperty, _zIndex++);

      // Create the storyboard.
      Storyboard flip = new Storyboard();

      // Create animation and set the duration to 1 second.
      DoubleAnimation animation = new DoubleAnimation()
      {
        Duration = new TimeSpan(0, 0, 1)
      };

      // Add the animation to the storyboard.
      flip.Children.Add(animation);

      // Create a projection for the image if it doesn't have one.
      if (image.Projection == null)
      {
        // Set the center of rotation to -0.01, which will put a little space
        // between the images when they're flipped.
        image.Projection = new PlaneProjection()
        {
          CenterOfRotationX = -0.01
        };
      }

      PlaneProjection projection = image.Projection as PlaneProjection;

      // Set the from and to properties based on the current flip direction
of
      // the image.
      if (projection.RotationY == 0)
      {
        animation.To = 180;
      }
      else
      {
        animation.From = 180;
        animation.To = 0;
```

```
        }

        // Tell the animation to animation the image's PlaneProjection object.
        Storyboard.SetTarget(animation, projection);

        // Tell the animation to animation the RotationYProperty.
        Storyboard.SetTargetProperty(animation,
          new PropertyPath(PlaneProjection.RotationYProperty));

        flip.Begin();
      }
    }
}
```

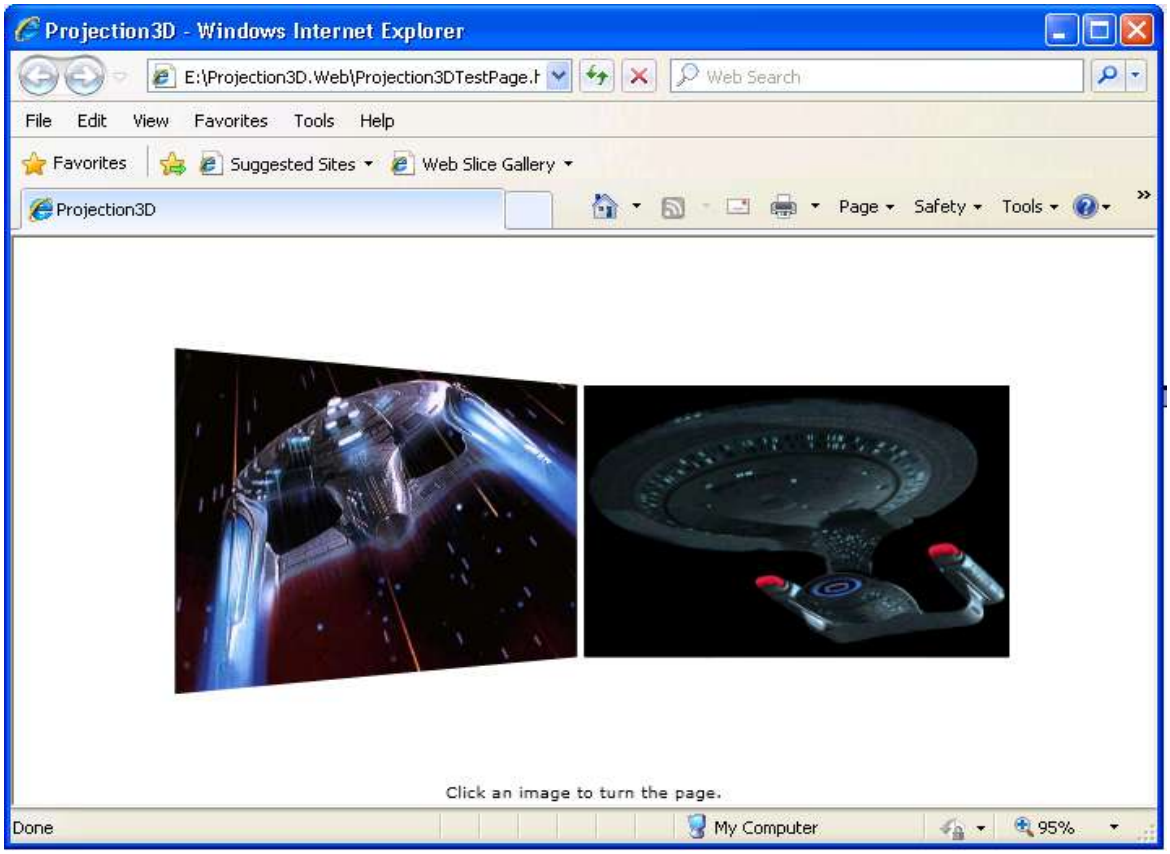
4.3 Giao diện ứng dụng

Ban đầu ta cảm nhận như một quyển Album đặt trước mặt:



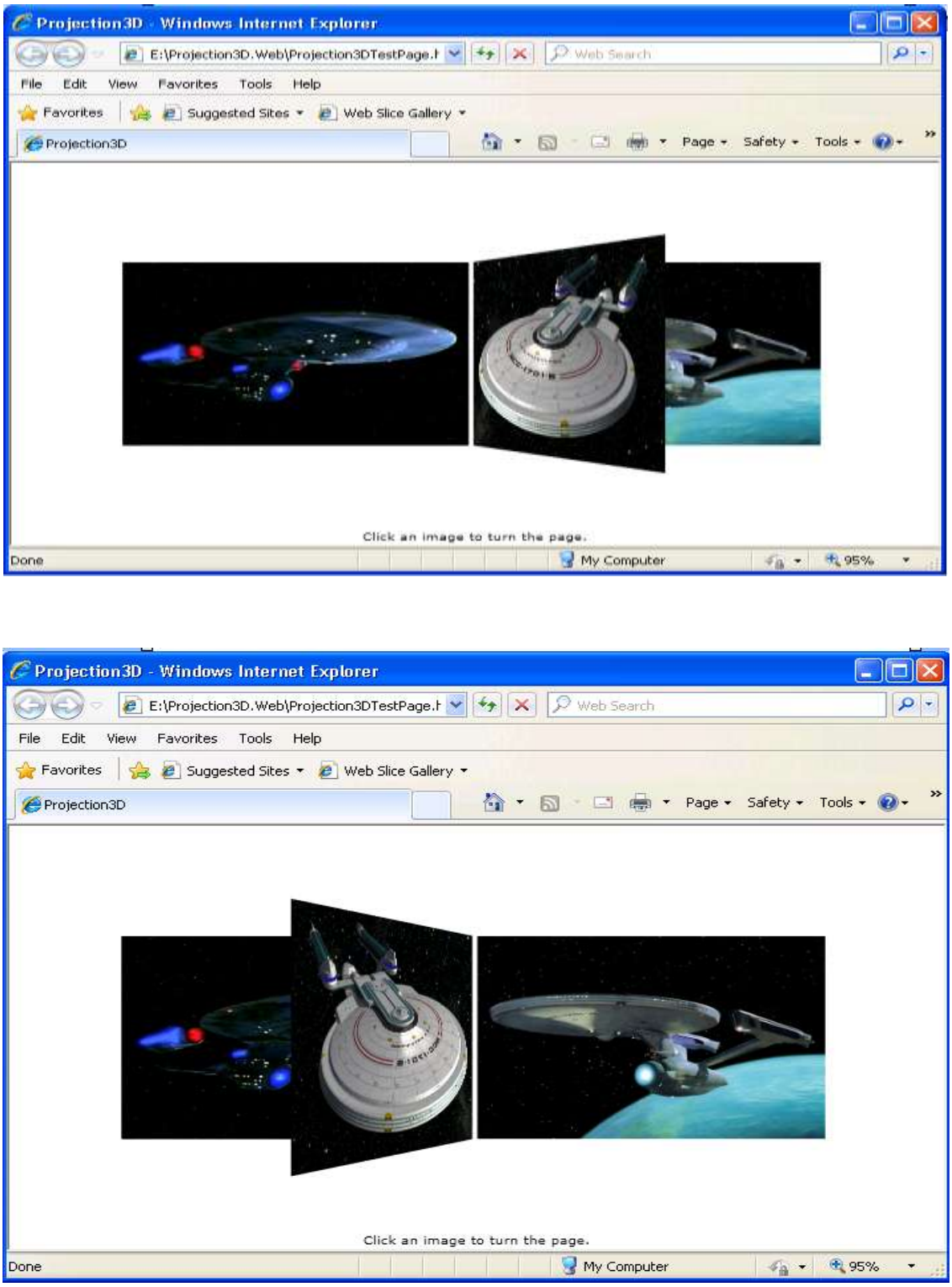
Hình 4.1 Giao diện chương trình ban đầu

Khi lật sang trang tiếp theo hoặc trước đó để xem ảnh, ta có cảm giác không khác gì đang mở một cuốn Album bình thường (sử dụng phép quay quanh trục 0y một góc 180^o) :



Hình 4.2: Giao diện chương trình khi lật trang

Mỗi trang được thiết kế đặc biệt gồm hai mặt (mặt bên này là phép soi gương của mặt bên kia):



Hình 4.3 : Giao diện một trang

TÀI LIỆU THAM KHẢO

[1] Christian Wenz, Essential Silverlight, O'Reilly.

[2] Adam Nathan, Silverlight Unleashed, SAMS.

[3] Silverlight Vietnamese.rar, Silverlight-step by step.rar.

[4] http://www.silverlight.net.

[5] Khóa luận tốt nghiệp Đại học "3D Streaming", Nguyễn Khắc Thắng – Trường Đại học Công Nghệ - Đại học Quốc gia Hà Nội.

[6] Đồ án tốt nghiệp "Tìm hiểu một số kỹ thuật trong đồ họa 3D và ứng dụng", Nguyễn Phi Hùng – Trường Đại Học Dân Lập Hải Phòng.

[7] Silverlight 2 Visual Essentials, Matthew MacDonald.

[8] Introducing Silverlight 2, Laurence Moroney.

[9] beginning Web Development Silverlight and ASP.NET AJAX, Laurence Moroney.

LỜI CẢM ƠN

Trước hết em xin chân thành cảm ơn thầy Trần Ngọc Thái là giáo viên hướng dẫn em trong quá tình thực tập. Thầy đã giúp em rất nhiều và đã cung cấp cho em nhiều tài liệu quan trọng phục vụ cho quá trình tìm hiểu về đề tài "Đồ họa 3D với Silverlight".

Thứ hai, em xin chân thành cảm ơn các thầy cô trong bộ môn công nghệ thông tin đã chỉ bảo em trong quá trình học và rèn luyện trong 4 năm học vừa qua. Đồng thời em cảm ơn các bạn sinh viên lớp CT1001 đã gắn bó với em trong quá trình rèn luyện tại trường.

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

Em xin chân thành cảm ơn!

Hải Phòng, ngày 10 tháng 07 năm 2010

Sinh viên thực hiện

Vũ Hoài Nam

MỤC LỤC

DANH MỤC HÌNH VẼ