

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



KỸ THUẬT ĐỒ HỌA

cuu duong than cong . com

(Dùng cho sinh viên hệ đào tạo đại học từ xa)

Lưu hành nội bộ

cuu duong than cong . com

HÀ NỘI - 2006

KỸ THUẬT ĐỒ HỌA

cuu duong than cong . com

Biên soạn : THS. TRỊNH THỊ VÂN ANH

cuu duong than cong . com

LỜI NÓI ĐẦU

Hiện nay đồ hoạ máy tính (Computer Graphics) là một trong những chương trình thông dụng nhất, nó đã góp phần quan trọng làm cho giao tiếp giữa con người và máy tính trở nên thân thiện hơn. Thật vậy, giao diện kiểu văn bản (text) đã được thay thế hoàn toàn bằng giao diện đồ hoạ, cùng với công nghệ đa phương tiện (multimedia) đã đưa ngành Công Nghệ Thông Tin sang một phiên bản mới.

Cuốn tài liệu giảng dạy này, tôi muốn mang lại cho bạn đọc các cơ sở lý thuyết về đồ hoạ máy tính từ đơn giản nhất như các thuật toán vẽ đường thẳng, đường tròn, đa giác, ký tự..... Tiếp đến các kỹ thuật xén tia, các phép biến đổi đồ hoạ trong không gian 2D và 3D.... Chúng ta lần lượt làm quen với thế giới màu sắc thông qua các hệ màu: RGB, CMYK, HSV.... Phức tạp hơn nữa là các phép chiếu, các phương pháp xây dựng đường cong và mặt cong cho đối tượng.

Tài liệu gồm bảy chương, trong đó chương một giúp bạn có cái nhìn tổng quan về kỹ thuật đồ hoạ từ trước đến giờ cùng định hướng tương lai cho lĩnh vực này. Các chương tiếp theo, mỗi chương sẽ là một vấn đề từ đơn giản đến phức tạp. Cuối mỗi chương đều có phần bài tập cho chúng ta kiểm tra lại kiến thức vừa đọc được. Bài tập gồm hai dạng: dạng tính toán và dạng lập trình, đối với dạng lập trình bạn có thể viết bằng C/C++ hay BC thậm chí bằng VB đều được. Cuối cùng là phần phụ lục gồm các hướng dẫn để chúng ta làm bài tập lập trình, ngôn ngữ hay dùng ở đây là C/C++ hay BC.

Bố cục rõ ràng, hình ảnh phong phú, đa dạng. Dù cho bạn chưa từng biết về đồ hoạ máy tính hay bạn đã nhiều năm làm việc trong lĩnh vực này, bạn đều có thể nhận thấy rằng cuốn sách này là một bộ tham khảo đầy đủ các thông tin hữu ích và có tính chất thực tiễn cao.

Trong quá trình biên soạn mặc dù đã cố gắng hết sức nhưng vẫn không tránh khỏi những sai sót, rất mong nhận được sự đóng góp chân thành từ quý bạn đọc.

Xin chân thành cảm ơn.

Tác giả

cuu duong than cong . com

CHƯƠNG 1: TỔNG QUAN VỀ KỸ THUẬT ĐỒ HOẠ

1. CÁC KHÁI NIỆM TỔNG QUAN CỦA KỸ THUẬT ĐỒ HOẠ MÁY TÍNH (COMPUTER GRAPHICS)

1.1. Lịch sử phát triển

- Graphics những năm 1950-1960

1959 Thiết bị đồ họa đầu tiên là màn hình xuất hiện tại Đức.

1960 - SAGE (Semi-Automatic Ground Environment System) xuất hiện bút sáng thao tác với màn hình.

1960 William Fetter nhà khoa học người Mỹ, ông đang nghiên cứu xây dựng mô hình buồng lái máy bay cho hãng Boeing của Mỹ. Ông đã dựa trên hình ảnh 3 chiều của mô hình người phi công trong buồng lái của máy bay để xây dựng nên một mô hình tối ưu cho buồng lái máy bay. Phương pháp này cho phép các nhà thiết kế quan sát một cách trực quan vị trí của người lái trong khoang. Ông đặt tên cho phương pháp này là đồ họa máy tính (Computer Graphics).

Màn hình là thiết bị thông dụng nhất trong hệ đồ họa, các thao tác của hầu hết các màn hình đều dựa trên thiết kế ống tia âm cực CRT (Cathode ray tube).

Khi đó giá để làm tươi màn hình là rất cao, máy tính xử lý chậm, đắt và không chắc chắn (không đáng tin cậy).

- Graphics: 1960-1970

1963 Ivan Sutherland (hội nghị Fall Joint Computer - lần đầu tiên có khả năng tạo mới, hiển thị và thay đổi được thực hiện trong thời gian thực trên màn CRT).

Hệ thống này được dùng để thiết kế mạch điện: CRT, LightPen (bút sáng), computer (chứa chương trình xử lý thông tin). Người sử dụng có thể vẽ mạch điện trực tiếp lên màn hình thông qua bút sáng.

- Graphics: 1970-1980

Raster Graphics (đồ họa điểm). Bắt đầu chuẩn đồ họa ví dụ như: GKS(Graphics Kernel System): European effort (kết quả của châu âu), Becomes ISO 2D standard.

- Graphics: 1980-1990

Mục đích đặc biệt về phần cứng, thiết bị hình học đồ họa Silicon. Xuất hiện các chuẩn công nghiệp: PHIGS (Programmers Hierarchical Interactive Graphics Standard) xác định các phương pháp chuẩn cho các mô hình thời gian thực và lập trình hướng đối tượng.

Giao diện người máy Human-Computer Interface (HCI)

- Computer Graphics: 1990-2000

OpenGL API (Application Program Interface – giao diện chương trình ứng dụng).

Completely computer-sinh ra ngành điện ảnh phim truyện (Toy Story) rất thành công.

Các tiềm tàng phần cứng mới: Texture mapping (dán các ảnh của cảnh thật lên bề mặt của đối tượng),blending (trộn màu)....

- Computer Graphics: 2000- nay

Ảnh hiện thực các card đồ họa cho máy tính (Graphics cards for PCs), game boxes and game players

Công nghiệp phim ảnh nhờ vào đồ họa máy tính (Computer graphics becoming routine in movie industry): Maya (thế giới vật chất tri giác được)....

1.2. Kỹ thuật đồ họa vi tính.

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

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

Computer Graphics (Kỹ thuật đồ họa 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 để: kiến tạo, xây dựng, lưu trữ, xử lý Các mô hình (model) và hình ảnh (image) của đối tượng. Các mô hình (model) và hình ảnh này có thể là kết quả thu được từ những lĩnh vực khác nhau của rất nhiều ngành khoa học (vật lý, toán học, thiên văn học...)

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

2. CÁC KỸ THUẬT ĐỒ HOẠ

2.1. Kỹ thuật đồ họa điểm (Sample based-Graphics)

- 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,

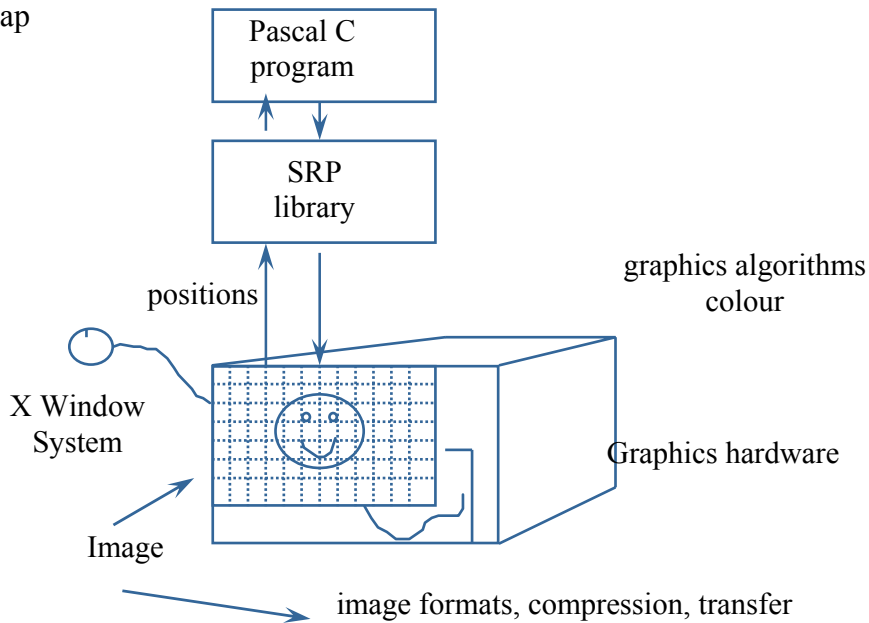
+ Từng pixel đều có vị trí xác định, được hiển thị với một giá trị rời rạc (số nguyên) các thông số hiển thị (màu sắc hoặc độ sáng)

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



Hình 1.1 Ảnh đồ họa điểm

Bitmap

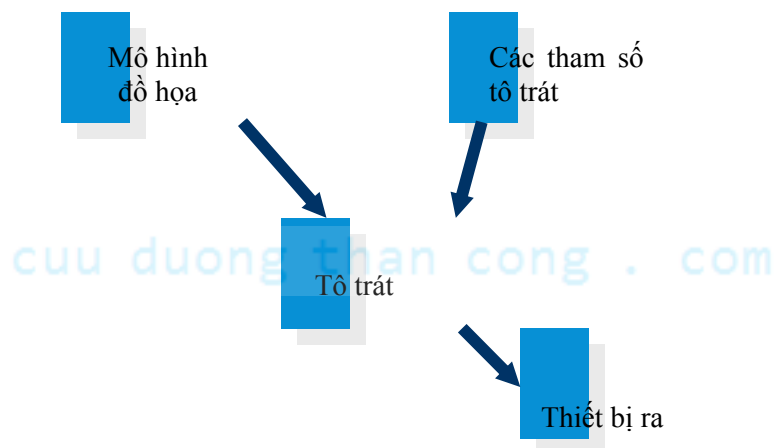


Hình 1.2 Kỹ thuật đồ họa điểm

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 của sự vật.
- Phương pháp rời rạc hoá (số hoá) 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.

2.2. Kỹ thuật đồ họa vector



Hình 1.3 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,

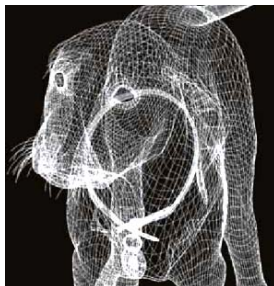
- Quá trình tô trát (rendering) để hiển thị từng điểm của mô hình, hình ảnh thực của đối tượng

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

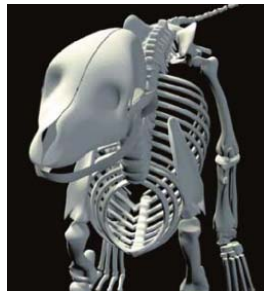
So sánh giữa Raster và Vector Graphics

Đồ họa điểm(Raster Graphics)	Đồ họa vector(Vector Graphics)
<ul style="list-style-type: none">- 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.	<ul style="list-style-type: none">- 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 cách thay đổi điểm nhìn và góc nhìn.

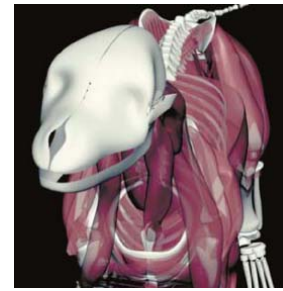
Ví dụ về hình ảnh đồ họa Vector



Wireframe



Skeletal



Muscle Model



Skin



Hair

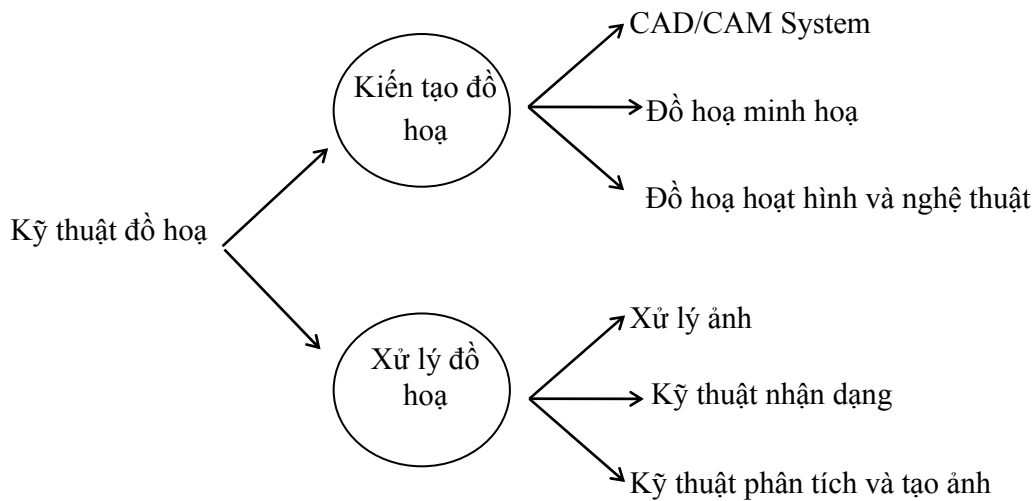


Render and Touch

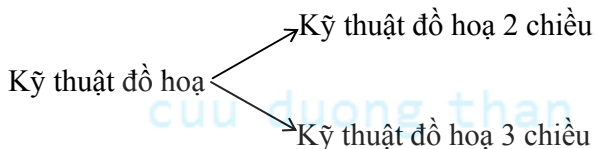
Hình 1.4 Ví dụ về đồ họa vector

2.3. Phân loại của đồ họa máy tính

Phân loại theo các lĩnh vực của đồ họa máy tính



Phân loại theo hệ tọa độ



- *Kỹ thuật đồ họa hai chiều*: là kỹ thuật đồ họa máy tính sử dụng hệ tọa độ hai chiều (hệ tọa độ phẳng), sử dụng rất nhiều trong kỹ thuật xử lý bản đồ, đồ thị.

- *Kỹ thuật đồ họa ba 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 cấu trúc, hoặc theo các tiêu chí được xác định từ trước và bằng các thuật toán chọn lọc để có thể phân tích hay tổng hợp ảnh đã cho thành một tập 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 3D Studio, 3D Animation, 3D Studio Max.

2.4. 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 họa 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 ưu thích nhờ 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 hoá văn phòng và chế bản điện tử*

- *Thiết kế với sự trợ giúp của máy tính (CAD_CAM)*

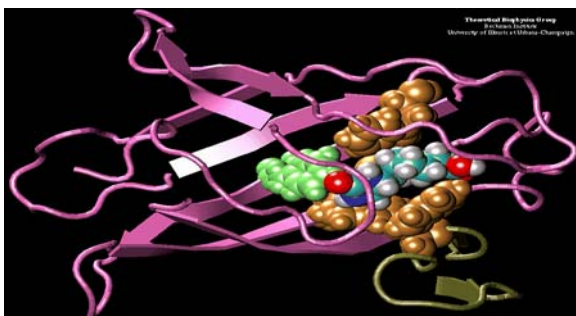
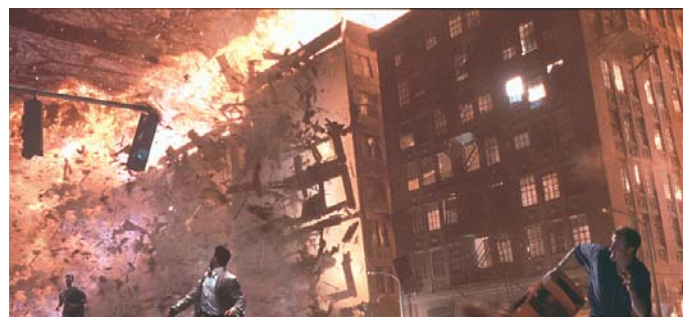
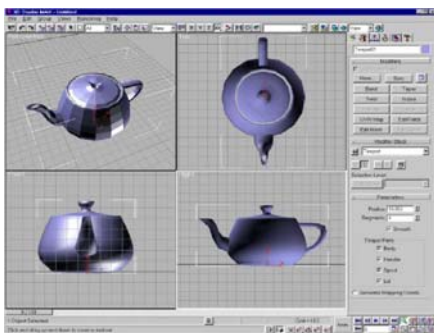
- *Lĩnh vực giải trí, nghệ thuật và mô phỏng*

- *Điều khiển các quá trình sản xuất (Process Control)*

- *Lĩnh vực bản đồ (Cartography)*

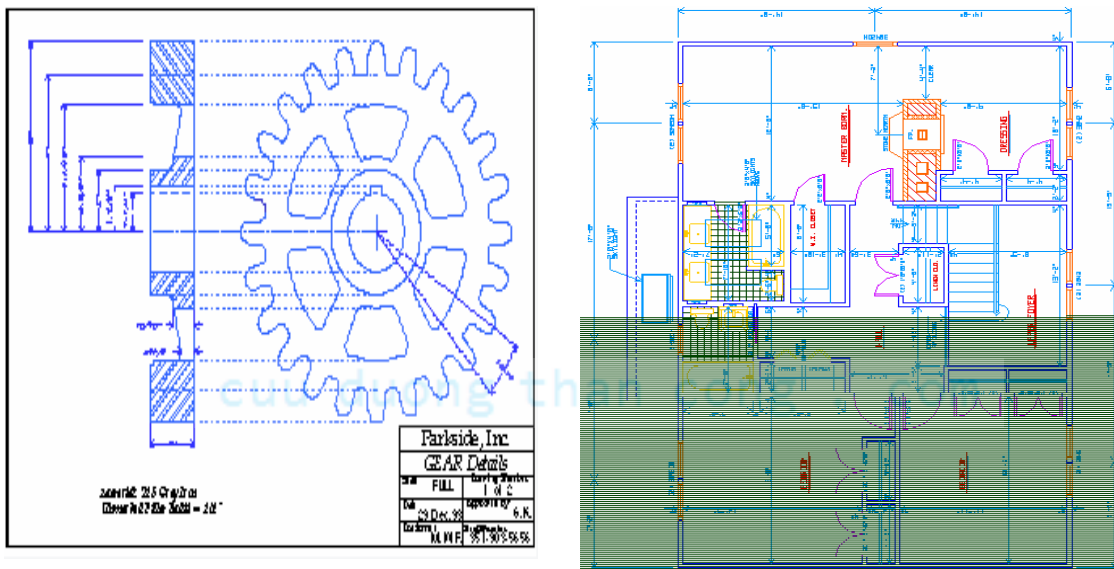
- *Giáo dục và đào tạo*

Một số ví dụ của ứng dụng kỹ thuật đồ họa:





Hình 1.5 Các ứng dụng của kỹ thuật đồ họa



Hình 1.6 Hệ ứng dụng CAD - CAM

2.5. Các chuẩn giao diện của hệ đồ họa

Mục tiêu căn bản của phần mềm đồ họa được chuẩn là tính tương thích. Khi các công cụ được thiết kế với hàm đồ họa chuẩn, phần mềm có thể được di chuyển một cách dễ dàng từ hệ phần cứng này sang hệ phần cứng khác và được dùng trong nhiều cài đặt và ứng dụng khác nhau.

GKS (Graphics Kernel System): chuẩn xác định các hàm đồ họa chuẩn, được thiết kế như một tập hợp các công cụ đồ họa hai chiều và ba chiều.

GKS Functional Description, ANSI X3.124 - 1985. GKS - 3D Functional Description, ISO Doc #8805:1988.

CGI (Computer Graphics Interface System): hệ chuẩn cho các phương pháp giao tiếp với các thiết bị ngoại vi.

CGM (Computer Graphics Metafile): xác định các chuẩn cho việc lưu trữ và chuyển đổi hình ảnh.

VRML (Virtual Reality Modeling Language): ngôn ngữ thực tại ảo, một hướng phát triển trong công nghệ hiển thị được đề xuất bởi hãng Silicon Graphics, sau đó đã được chuẩn hóa như một chuẩn công nghiệp.

PHIGS (Programmers Hierarchical Interactive Graphics Standard): xác định các phương pháp chuẩn cho các mô hình thời gian thực và lập trình hướng đối tượng.

PHIGS Functional Description, ANSI X3.144 - 1985. + Functional Description, 1988, 1992.

OPENGL thư viện đồ họa của hãng Silicon Graphics, được xây dựng theo đúng chuẩn của một hệ đồ họa năm 1993.

DIRECTX thư viện đồ họa của hãng Microsoft, Direct X/Direct3D 1997

3. PHẦN CỨNG ĐỒ HOẠ (GRAPHICS HARDWARE)

3.1. Các thành phần phần cứng của hệ đồ họa tương tác

CPU: thực hiện các chương trình ứng dụng.

Bộ xử lý hiển thị (Display Processor): thực hiện công việc hiển thị dữ liệu đồ họa.

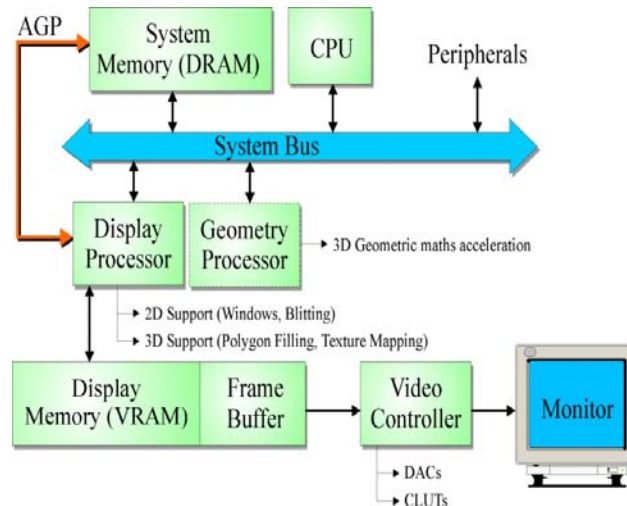
Bộ nhớ hệ thống (System Memory): chứa các chương trình và dữ liệu đang thực hiện.

Gói phần mềm đồ họa (Graphics Package): cung cấp các hàm đồ họa cho chương trình ứng dụng

Phần mềm ứng dụng (Application Program): phần mềm đồ họa ứng dụng.

Bộ đệm (Frame buffer): có nhiệm vụ chứa các hình ảnh hiển thị.

Bộ điều khiển màn hình (Video Controller): điều khiển màn hình, chuyển dữ liệu dạng số ở frame buffer thành các điểm sáng trên màn hình.



Hình 1.7 Các thành phần cứng của hệ đồ họa tương tác

3.2. Máy in

Dot size: đường kính của một điểm in bé nhất mà máy in có thể in được

Addressability: khả năng địa chỉ hoá các điểm in có thể có trên một đơn vị độ dài (dot per inch)

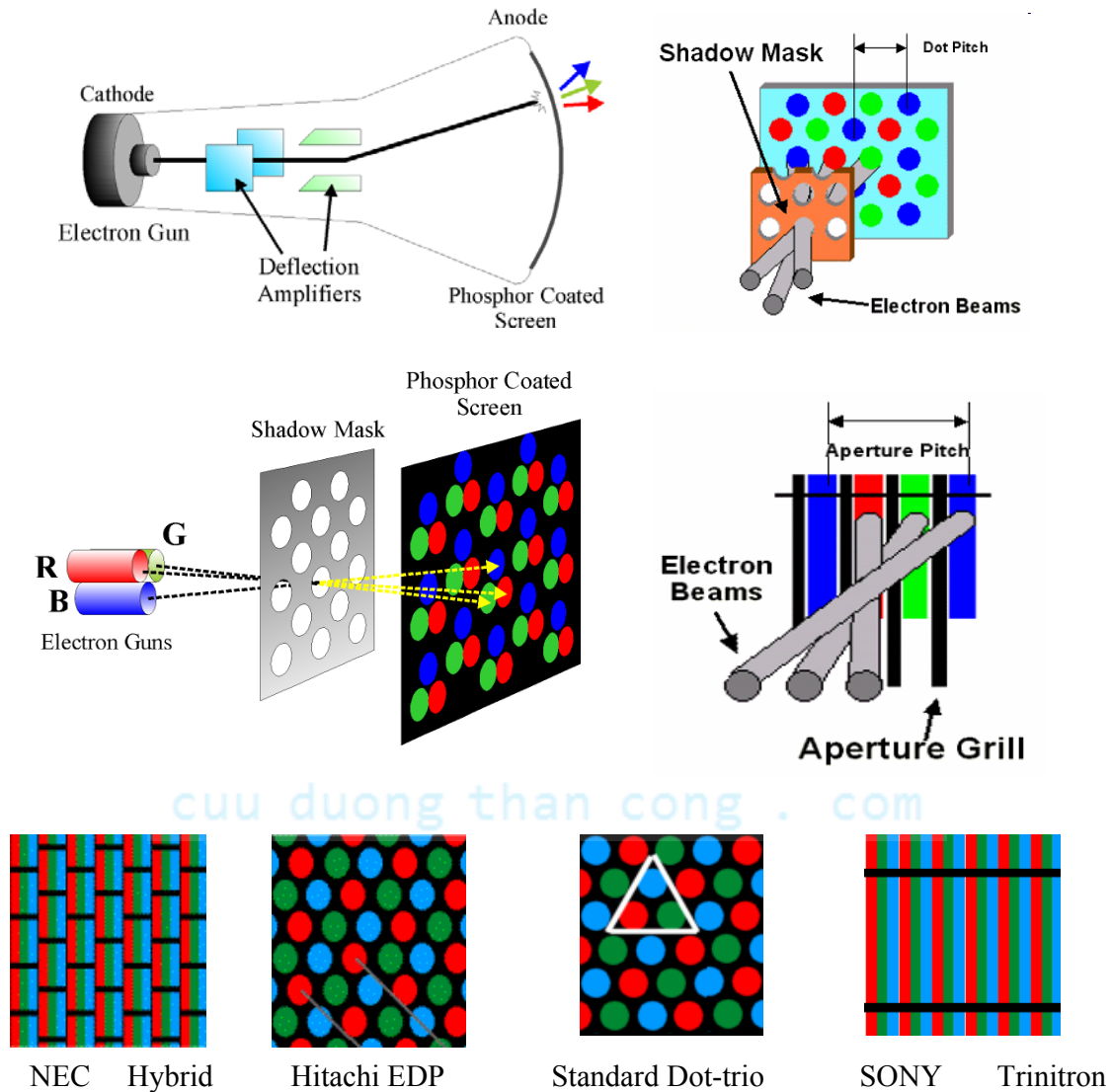
Số lượng màu có thể vẽ trên một điểm:

	Dot size	Point per inch
	8 - 20/ 100inch	200, 600
	5/1000inch	1500
Máy vẽ	6,15/1000 inch	1000, 2000

3.3. Màn hình CRT

Một chùm các tia điện tử (tia âm cực) phát ra từ một súng điện tử, vượt qua cuộn lái tia dẫn đến vị trí xác định trên màn hình được phủ một lớp phosphor. Tại mỗi vị trí tương tác với tia điện tử hạt phosphor sẽ phát lên một chấm sáng nhỏ. Nhưng chấm sáng sẽ mờ dần rất nhanh nên cần có cách nào nó duy trì ảnh trên màn hình. Một trong các cách là: lặp đi lặp lại nhiều lần việc vẽ lại ảnh thật nhanh bằng cách hướng các tia điện tử trở lại vị trí cũ. Gọi là làm tươi (refresh CRT).

Số lượng tối đa các điểm có thể hiển thị trên một CRT được gọi là độ phân giải (Resolution). Hay độ phân giải là số lượng các điểm trên một cm mà có thể được vẽ theo chiều ngang và chiều dọc (được xem như tổng số điểm theo mỗi hướng).



Hình 1.8 Công nghệ màn hình CRT

Kích thước vật lý của màn hình đồ họa được tính từ độ dài của đường chéo màn hình. Thường dao động từ 12-27 inch, hoặc lớn hơn.

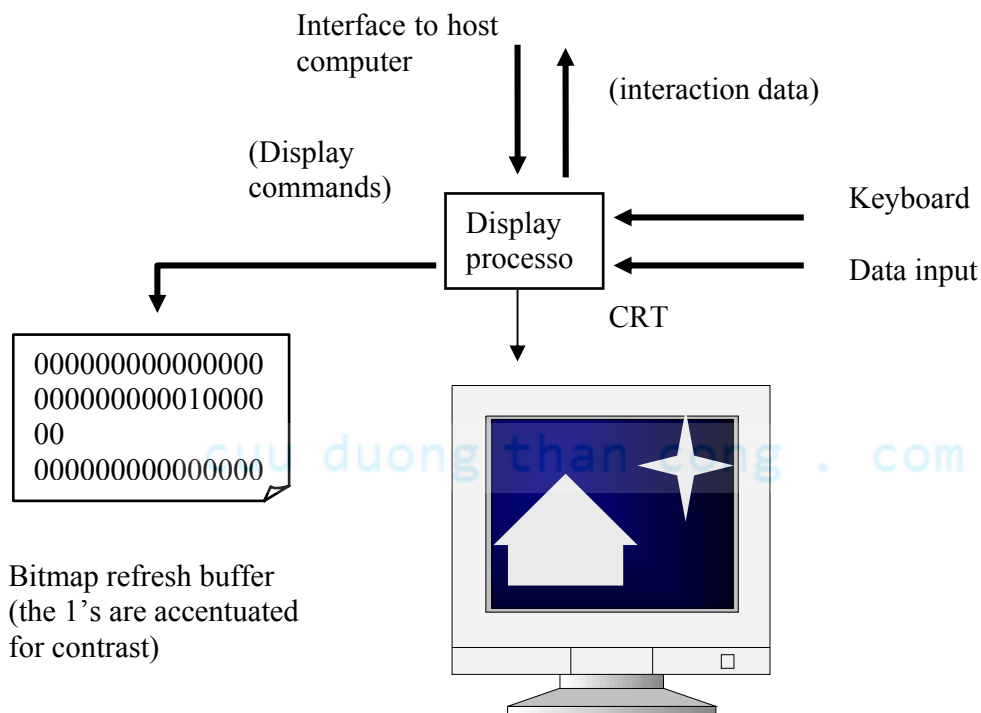
Thuộc tính khác của màn hình là tỷ số phương (aspect ratio). Nó là tỷ lệ của các điểm dọc và các điểm ngang cần để phát sinh các đoạn thẳng có độ dài đơn vị theo cả hai hướng trên màn hình. Màn hình có tỷ số phương khác một, thì hình vuông hiển thị trên đó thành hình chữ nhật còn hình tròn thành hình ellipse.

Màn hình dạng điểm (Raster Display): thường gặp nhất trong số các dạng màn hình sử dụng CRT trên công nghệ truyền hình. Mỗi điểm trên màn hình được gọi là pixel. Các thông tin về ảnh hiển thị trên màn hình được lưu trữ trong một vùng bộ nhớ gọi là vùng đệm làm tươi (Refresh buffer) hay là vùng đệm khung (Frame Buffer). Vùng lưu trữ tập các giá trị cường độ sáng của toàn bộ các điểm trên màn hình và luôn tồn tại một cách song ánh giữa mỗi điểm trên màn hình và mỗi phần tử trong vùng này.

Để tạo ra hình ảnh đen trắng, đơn giản chỉ cần lưu thông tin của mỗi Pixel là một bit (0,1) (xem hình 1.9). Trong trường hợp ảnh nhiều màu thì cần nhiều bit hơn, nếu thông tin mỗi pixel được lưu bằng b bit thì ta có thể có 2^b giá trị màu phân biệt cho pixel đó.

Trong các màn hình màu, người ta định nghĩa tập các màu làm việc trong một bảng tra (LookUp Table - LUT). Mỗi phần tử của LUT được định nghĩa một bộ ba giá trị (RGB) mô tả một màu nào đó. Khi cần sử dụng một màu, ta chỉ cần chỉ định số thứ tự (index) tương ứng của màu đó trong LUT, số phần tử trong bảng LUT chính là số màu có thể được hiển thị cùng một lúc trên màn hình.

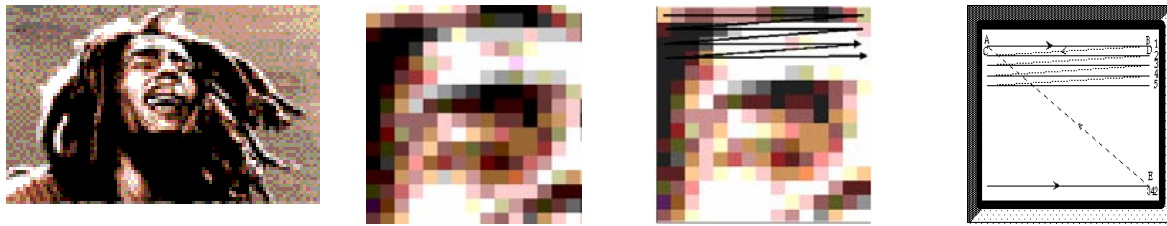
Ví dụ mô hình đồ họa điểm ngôi nhà và ngôi sao.



Hình 1.9 Song ánh giữa vùng đệm khung và màn hình

X: 0 , Xmax2 màu/ 1 bit	640 x 480 x 16 → Video RAM = 2MB
Y: 0 , Ymax16 màu/ 4 bit ;256 màu/ 8bit	1024 x 1024 x 24 → Video RAM = 24MB
2^{16} màu/ 16 bit ; 2^{24} màu/ 24 bit	

Việc làm tươi trên màn hình dạng này được thực hiện ở tốc độ 60 - 80 frame/giây. Đôi khi tốc độ làm tươi còn được biểu diễn bằng đơn vị Hertz (Hz - số chu kỳ trên/giây), trong đó một chu kỳ tương ứng với một frame. Vậy tốc độ làm tươi 60 frame/giây đơn giản là 60 Hz. Khi đạt đến cuối mỗi dòng quét, tia điện tử quay trở lại bên trái của màn hình để bắt đầu dòng quét kế tiếp. Việc quay trở về bên trái màn hình sau khi làm tươi mỗi dòng quét được gọi là tia hồi ngang (Horizontal retrace). Và tới cuối mỗi frame, tia điện tử (tia hồi dọc - Vertical retrace) quay trở lại góc bên trái của màn hình để chuẩn bị bắt đầu frame kế tiếp.



Hình 1.10 Quét màn và quét dòng của màn hình CRT

Ví dụ về việc tia quét trên màn hình CRT:

```
MOVE 10,15
```

```
LINE 400,300
```

```
LINE 600,800
```

```
Refresh Buffer
```

```
DrawLine(A, B):
```

```
Turn beam off,
```

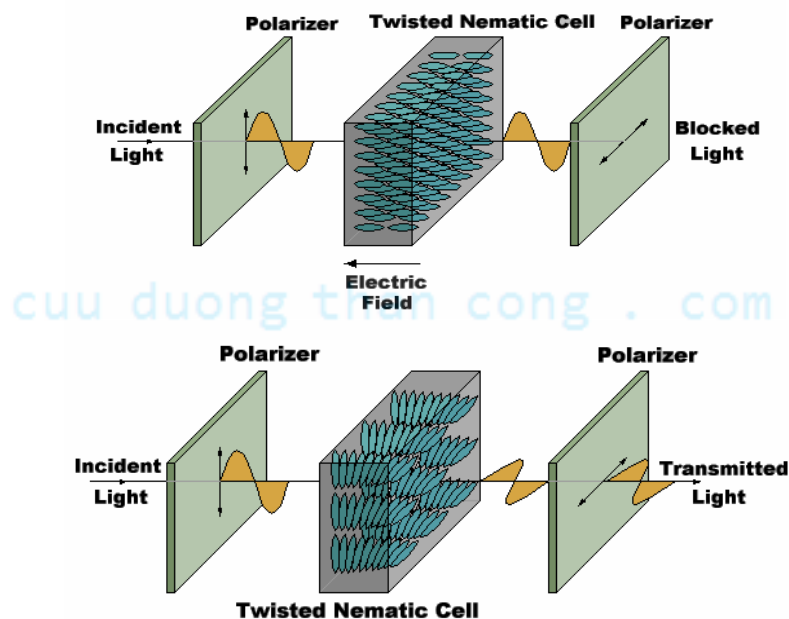
```
move to A.
```

```
Turn beam on,
```

```
move to B.
```

3.4. Màn hình tinh thể lỏng (Liquid Crystal Display – LCD)

Dựa vào công nghệ truyền ánh sáng qua điện cực mà đặt giữa là cuộn dây xoắn. Khi chưa có từ trường (chưa có dòng điện) ở cuộn dây thì ánh sáng truyền thẳng, khi có từ trường thì ánh sáng truyền đổi chiều.



Hình 1.11 Công nghệ truyền ánh sáng trong màn hình tinh thể lỏng

CRT Displays (màn hình CRT)

Advantages (ưu điểm)	Disadvantages (nhược điểm)
Đáp ứng nhanh (có độ phân giải cao)	Lớn và nặng (typ. 70x70 cm, 15 kg)
Màu sắc đa dạng (Có độ sâu và rộng)	Tiêu tốn nguồn điện cao (typ. 140W)
Màu sắc bão hoà và tự nhiên	Có hại cho sức khoẻ vì trường điện từ và từ tính
Công nghệ không quá đắt và hoàn thiện	Màn hình nhấp nháy (at 50-80 Hz)
Góc nhìn rộng, tương phản và độ sáng cao	Hình hay bị méo tại 4 góc

LCD Displays (màn hình tinh thể lỏng)

Advantages (ưu điểm)	Disadvantages (nhược điểm)
Hình dáng nhỏ, trọng lượng nhẹ (approx 1/6 of CRT, typ. 1/5 of CRT)	Giá thành cao (presently 3x CRT)
Tiêu tốn nguồn thấp (typ. 1/4 of CRT)	Góc nhìn hẹp hơn (typ. +/- 50 degrees)
Màn hình phẳng tuyệt đối nên không méo tại các góc	độ tương phản thấp (typ. 1:100)
Màu sắc đều, ảnh sinh động	độ chói (độ ngời) thấp hơn (typ. 200 cd/m ²)
Không bị hiệu ứng điện từ trường	
Có thể màn hình vừa lớn vừa rộng (>20 inch)	

Tóm tắt chương:

Sự ra đời của đồ họa máy tính thực sự là cuộc cách mạng trong giao tiếp giữa người dùng và máy tính. Với lượng thông tin trực quan, đa dạng và phong phú được truyền tải qua hình ảnh. Các ứng dụng đồ họa máy tính đã lôi cuốn nhiều người nhờ tính thân thiện, dễ dùng, kích thích khả năng sáng tạo và tăng đáng kể hiệu suất làm việc.

Đồ họa máy tính ngày nay được ứng dụng rất rộng rãi trong nhiều lĩnh vực khoa học, kỹ thuật, nghệ thuật, kinh doanh, quản lý... Các ứng dụng đồ họa rất đa dạng, phong phú và phát triển liên tục không ngừng. Ngày nay, hầu như không có chương trình ứng dụng nào mà không sử dụng kỹ thuật đồ họa để làm tăng tính hấp dẫn cho mình.

Một hệ thống đồ họa bao giờ cũng gồm hai phần chính đó là phần cứng và phần mềm. Phần cứng bao gồm các thiết bị hiển thị (thiết bị xuất) và các thiết bị nhập. Tiêu biểu nhất là màn hình, có hai loại thông dụng là CRT và LCD.

Bài tập:

1. Cấu tạo và nguyên lý hoạt động của màn hình dạng điểm. Nêu các khái niệm vùng đệm khung, độ phân giải, tỷ số phương.... của màn hình loại này?
2. Ý nghĩa và hoạt động của bảng tra LUT?

3. Tính Video Ram của các màn hình lần lượt có độ phân giải là 640x480, 1024x768, 1280x1024 mà có mỗi pixel được mô tả là 8bit, 12 bit, 24 bit.
4. Nếu chúng ta dùng các giá trị 12bit cho mỗi pixel trong một bảng tham chiếu lookup table, có bao nhiêu hạng mục mà lookup table có được?
5. Tại sao phải chuẩn hoá các phần mềm? Liệt kê và tìm hiểu các chuẩn hoá phần mềm đồ họa.

Bài tập trắc nghiệm:

1. Tỷ số phương (aspect ratio) của màn hình là 1,4 vậy một hình tròn khi hiển thị trên màn hình đó sẽ cho:
 - a. Hình tròn
 - b. Hình ellipse nằm ngang (bán kính theo trục x dài hơn bán kính theo trục y)
 - c. Hình ellipse đứng (bán kính theo trục x ngắn hơn bán kính theo trục y)
 - d. Hình thoi
2. Cho màn có độ phân giải 1024x1024 và mỗi pixel được mô tả 24bit vậy video RAM của màn hình là:
 - a. 1048576 bit
 - b. 2MB
 - c. 3MB
 - d. 4MB
3. Nếu ta dùng các giá trị 24 bit cho mỗi pixel trong một bảng LUT. Thì bảng LUT có số màu là:
 - a. 24 màu
 - b. 1024 màu
 - c. 16777216 màu
 - d. 16000000 màu

CHƯƠNG 2: CÁC GIẢI THUẬT SINH THỰC THỂ CƠ SỞ

1. CÁC ĐỐI TƯỢNG ĐỒ HOẠ CƠ SỞ

1.1. Hệ toạ độ thế giới thực và hệ toạ độ thiết bị

a. Hệ toạ độ thế giới thực (WCS: World Coordinate System)

WCS hay hệ toạ độ thực là hệ toạ độ được dùng mô tả các đối tượng trong thế giới thực. Một trong hệ toạ độ thực được dùng nhiều nhất là hệ toạ độ Descartes. Bất kì điểm nào trong mặt phẳng được mô tả bằng cặp toạ độ (x,y) trong đó $x,y \in \mathbb{R}$. Gốc toạ độ là điểm O có toạ độ $(0,0)$, O_x, O_y lần lượt là trục hoành và trục tung và x,y là hoành độ và tung độ.

Các toạ độ thế giới thực cho phép người sử dụng bất kì một thứ nguyên (dimension) qui ước: foot, cm, nm, km, inch....tùy ý.

b. Hệ toạ độ thiết bị (DCS: Device Coordinate System)

Hệ toạ độ thiết bị là hệ toạ độ được dùng bởi một thiết bị xuất cụ thể nào đó như máy in, màn hình...

Các điểm được biểu diễn bởi cặp toạ độ (x,y) , nhưng $x,y \in \mathbb{N}$. Điểm trong toạ độ thực được định nghĩa liên tục, còn trong toạ độ thiết bị thì rời rạc do tính chất của tập các số tự nhiên.

Các toạ độ (x,y) có giới hạn trong một khoảng nào đó.

1.2. Điểm và đoạn thẳng

a. Điểm

Trong hệ toạ độ hai chiều (x,y) , ngoài ra nó còn có tính chất màu sắc.

b. Đoạn thẳng

+ Biểu diễn tường minh: $y = f(x)$

Một đoạn thẳng được xác định nếu biết 2 điểm thuộc nó. Phương trình đoạn thẳng đi qua 2 điểm P (x_1, y_1) và Q (x_2, y_2) như sau:

$$(y - y_1) / (x - x_1) = (y_2 - y_1) / (x_2 - x_1)$$

$$(y - y_1)(x_2 - x_1) = (x - x_1)(y_2 - y_1)$$

$$(x_2 - x_1)y = (y_2 - y_1)x + y_1(x_2 - x_1) - x_1(y_2 - y_1)$$

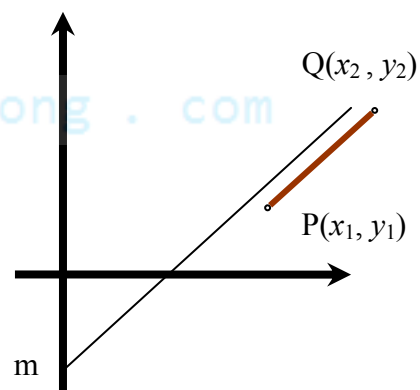
$$y = ((y_2 - y_1) / (x_2 - x_1))x + y_1 - ((y_2 - y_1) / (x_2 - x_1))x_1$$

$$y = kx + m$$

$$k = (y_2 - y_1) / (x_2 - x_1) \text{ Độ dốc hay hệ số góc của đường}$$

$$m = y_1 - kx_1 \text{ Đoạn chắn trên trục y}$$

$$\Delta y = k\Delta x \text{ (tức là khi } x \text{ thay đổi thì } y \text{ thay đổi theo)}$$



Hình 2.1 Vẽ đoạn thẳng PQ

+ Biểu diễn không tường minh: $ax + by + c = 0$

Ta có

$$(y_2 - y_1)x - (x_2 - x_1)y + (x_2 - x_1)y_1 - (y_2 - y_1)x_1 = 0$$

$$(y_2 - y_1)x - (x_2 - x_1)y + x_2y_1 - x_1y_2 = 0$$

hay $rx + sy + t = 0$

$$s = -(x_2 - x_1)$$

$$r = (y_2 - y_1) \text{ và } t = x_2y_1 - x_1y_2$$

+ Biểu diễn thông qua tham số:

$$P(u) = P_1 + u(P_2 - P_1) \quad u \in [0, 1]$$

$$x(u) = x_1 + u(x_2 - x_1)$$

$$y(u) = y_1 + u(y_2 - y_1)$$

2. CÁC GIẢI THUẬT XÂY DỰNG THỰC THỂ CƠ SỞ

2.1. Giải thuật vẽ đoạn thẳng thông thường

Nguyên lý chung: cho một thành phần tọa độ x hay y biến đổi theo từng đơn vị và tính độ nguyên còn lại sao cho gần với tọa độ thực nhất.

$$\text{Ta có } y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$

Cho x thay đổi tìm y, trong bài này cho x_1 thay đổi tiến tới x_2 ta chọn đơn vị nhỏ nhất của màn hình $\Delta x = 1$.

Giải thuật thông thường:

```
void dline(int x1, int y1, int x2, int y2, int color)
{
    float y;
    int x;
    for (x = x1; x <= x2; x++)
    {
        y = y1 + (x - x1) * (y2 - y1) / (x2 - x1);
        putpixel(x, Round(y), color);
    }
}
```

2.2. Thuật toán DDA (Digital Differential Analyzer)

Tiến hành tính tại mỗi bước vốn sử dụng kết quả từ bước trước đó. Giả sử bước i đã tính (x_i, y_i) , bước tiếp (x_{i+1}, y_{i+1}) sẽ nghiệm đúng với $\Delta y / \Delta x = k$.

$$\Delta y = y_{i+1} - y_i \quad \Delta x = x_{i+1} - x_i$$

$$\text{Vậy: } y_{i+1} = y_i + k \Delta x \text{ và } x_{i+1} = x_i + \Delta x$$

- $0 < k < 1$ (đảm bảo sự thay đổi của x trên trục tọa độ sẽ lớn hơn y)

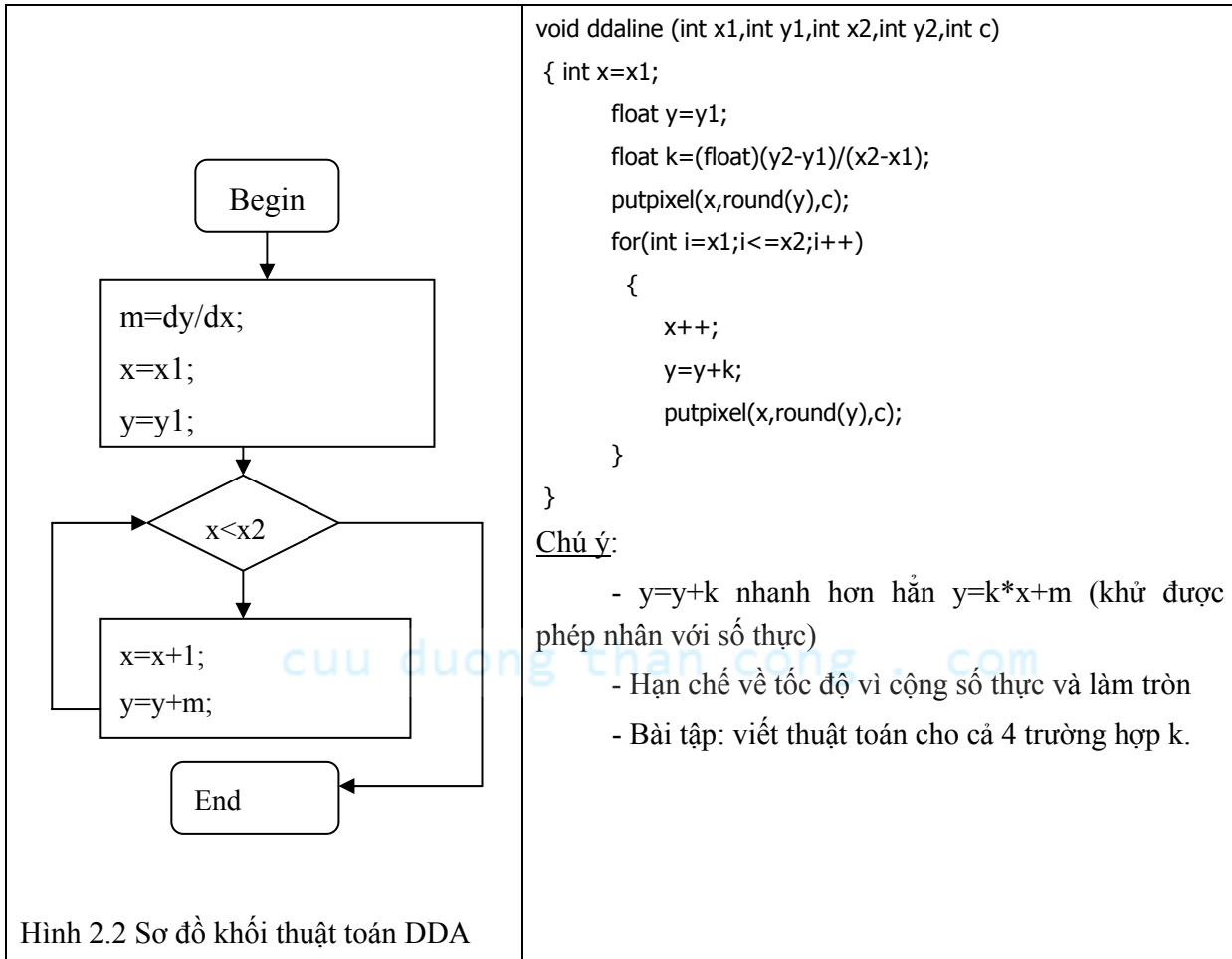
- Bắt đầu $x = x_1$ ($x_1 < x_2$) và $y = y_1$

$$x_{i+1} = x_i + 1 \text{ đặt } \Delta x = 1 \text{ (giá số theo x)}$$

$$y_{i+1} = y_i + k \text{ cứ như thế đến } x_2$$

- Khi $k > 1$ bắt đầu $y=y_1$ ($y_1 < y_2$) và $x=x_1$
- đặt $\Delta y = 1$ (gia số theo y)
- $x_{i+1} = x_i + 1/k$ tiếp tục đến y_2

Thuật toán

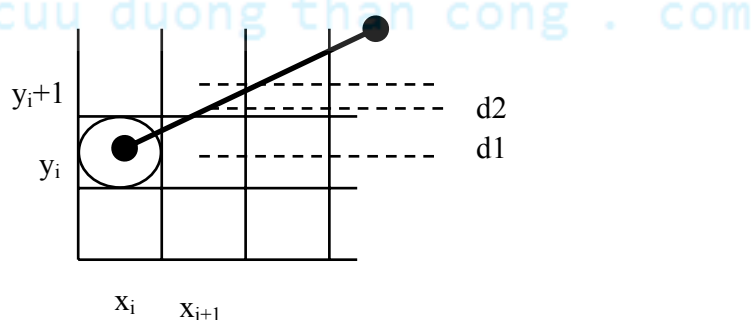


Hình 2.2 Sơ đồ khối thuật toán DDA

2.3. Giải thuật Bresenham

1960 Bresenham thuộc IBM theo nguyên lý tìm ra các điểm gần với đường thẳng dựa trên độ phân giải hữu hạn. Giải thuật này loại bỏ được các phép toán chia và phép toán làm tròn như ta đã thấy trong giải thuật DDA.

Xét đoạn thẳng với $0 < k < 1$



Hình 2.3 Mô tả giải thuật Bresenham

Gọi (x_{i+1}, y) là điểm thuộc đoạn thẳng, ta có $y = k(x_i + 1) + b$

$$d_1 = y - y_i = k(x_i + 1) + b - y_i$$

$$d_2 = y_{i+1} - y = y_{i+1} - k(x_i + 1) - b$$

$$\text{- Nếu } d_1 \leq d_2 \Rightarrow y_{i+1} = y_i$$

$$\text{- Ngược lại } d_1 > d_2 \Rightarrow y_{i+1} = y_i + 1$$

$$\text{Đặt } D = d_1 - d_2 = 2k(x_i + 1) - 2y_i + 2b - 1$$

$$\text{Có } k = \Delta y / \Delta x$$

$$\text{Đặt } P_i = \Delta x D = \Delta x (d_1 - d_2)$$

$$P_i = \Delta x (2\Delta y / \Delta x (x_i + 1) - 2y_i + 2b - 1)$$

$$= 2\Delta y x_i + 2\Delta y - 2\Delta x y_i + 2b\Delta x - \Delta x$$

Ta tính bước tiếp:

$$P_{i+1} = 2\Delta y x_{i+1} + 2\Delta y - 2\Delta x y_{i+1} + 2b\Delta x - \Delta x$$

$$P_{i+1} - P_i = -2\Delta x (y_{i+1} - y_i) + 2\Delta y (x_{i+1} - x_i)$$

Có $x_{i+1} = x_i + 1$ nên:

$$P_{i+1} - P_i = -2\Delta x (y_{i+1} - y_i) + 2\Delta y = 2\Delta y - 2\Delta x (y_{i+1} - y_i)$$

Nếu $P_i \leq 0$ thì $y_{i+1} = y_i$

$$P_{i+1} = P_i + 2\Delta y$$

Nếu $P_i > 0$ thì $y_{i+1} = y_i + 1$

$$P_{i+1} = P_i + 2\Delta y - 2\Delta x$$

Tính giá trị đầu: P_1 ?

$$P_1 = \Delta x (d_1 - d_2)$$

$$= \Delta x (2\Delta y / \Delta x (x_1 + 1) - 2y_1 + 2b - 1)$$

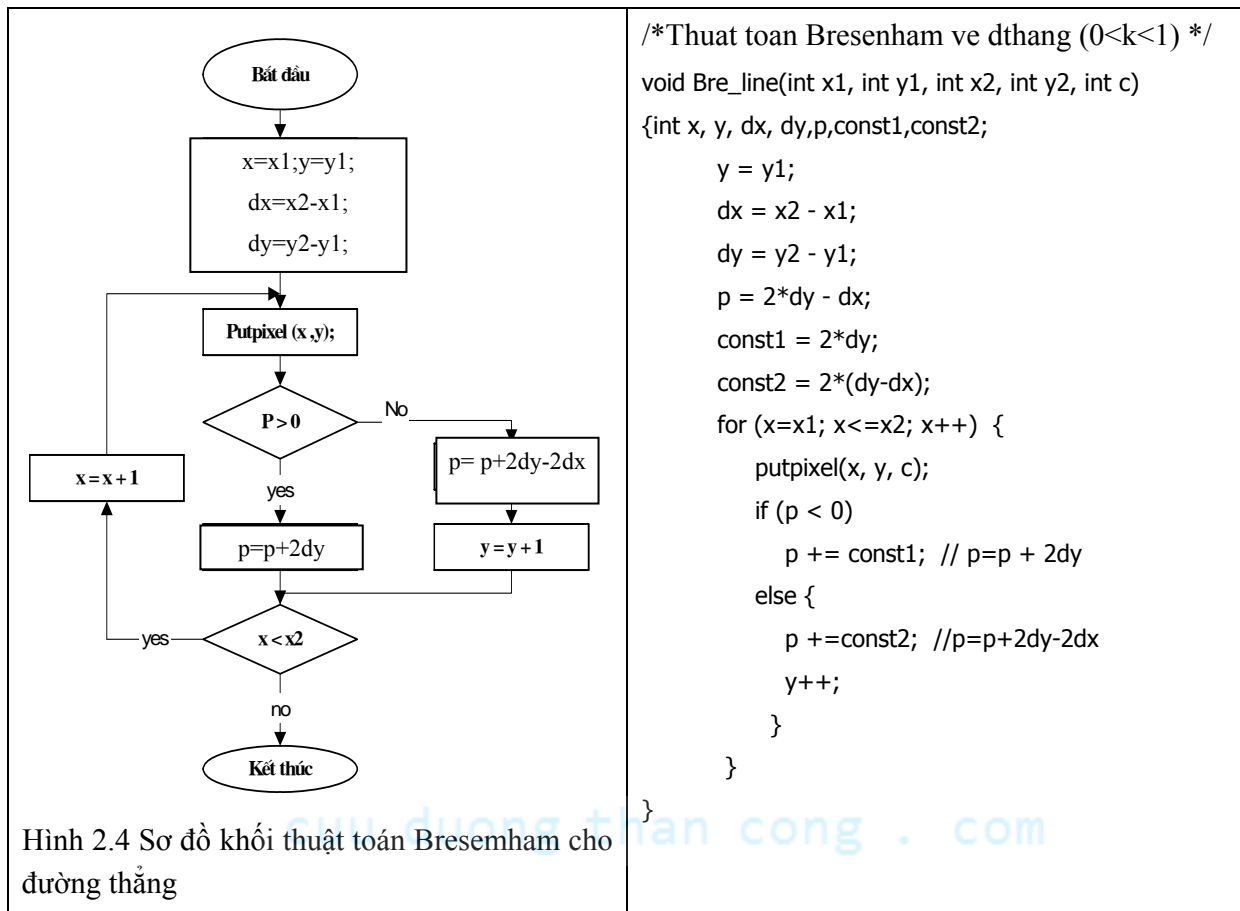
$$= 2\Delta y x_1 + 2\Delta y - 2\Delta x y_1 + 2b\Delta x - \Delta x$$

Có $y_1 = kx_1 + b = \Delta y / \Delta x x_1 + b$

$$P_1 = 2\Delta y x_1 + 2\Delta y - 2\Delta x ((\Delta y / \Delta x) x_1 + b) + 2b\Delta x - \Delta x$$

$$= 2\Delta y x_1 + 2\Delta y - 2\Delta y x_1 - 2b\Delta x + 2b\Delta x - \Delta x$$

$$P_1 = 2\Delta y - \Delta x$$



2.4. Giải thuật trung điểm-Midpoint

Jack Bresenham 1965 / Pitteway 1967, áp dụng cho việc sinh các đường thẳng và đường tròn 1985.

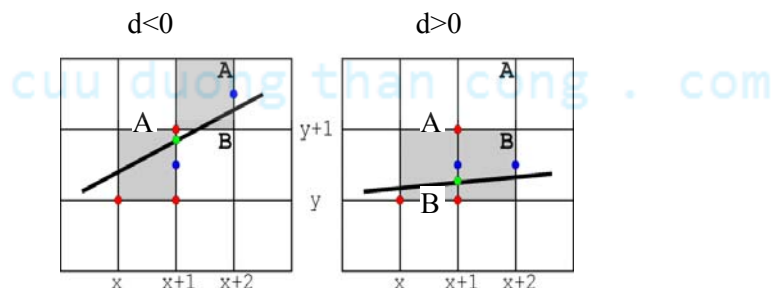
Xét trung điểm của đoạn AB (M)

Nếu M ở trên đoạn thẳng AB thì chọn B còn

M ở dưới đoạn thẳng AB chọn A

Công thức đơn giản hơn, tạo được các điểm tương tự như với Bresenham

$d = f(x_i + 1, y_i + 1/2)$ là trung điểm của đoạn AB



Hình 2.5 Mô tả giải thuật Midpoint

So sánh hay kiểm tra M sẽ được thay bằng việc xét giá trị d.

- $d > 0$ điểm B được chọn khi đó $y_{i+1} = y_i$

- nếu $d < 0$ điểm A được chọn khi đó $y_{i+1} = y_i + 1$

Trường hợp $d = 0$ chúng ta có thể chọn điểm bất kỳ hoặc A, hoặc B.

Sử dụng phương pháp biểu diễn không tường minh

$$f(x,y) = ax + by + c = 0 \quad (1) \quad dx = x_2 - x_1 \quad dy = y_2 - y_1$$

Biểu diễn tường minh:

$$y = (dy/dx)x + B \text{ hay}$$

$$f(x,y) = 0 = xdy - ydx + Bdx \quad (2)$$

So sánh (1) và (2)

$$a = dyb = -dx \quad c = Bdx$$

Có $f(x,y) = 0$ với mọi (x,y) thuộc đường thẳng

Đặt $d_i = f(x_i+1, y_i+1/2) = a(x_i+1) + b(y_i+1/2) + c$

+ Nếu chọn A ($d < 0$) thì M sẽ tăng theo 2 hướng x, y

$$d_{i+1} = f(x_i+2, y_i+3/2) = a(x_i+2) + b(y_i+3/2) + c$$

$$d_{i+1} - d_i = a + b$$

$$\text{Hay } d_{i+1} = d_i + dy - dx$$

+ Nếu chọn B ($d > 0$) thì M sẽ tăng theo x

$$d_{i+1} = f(x_i+2, y_i+1/2) = a(x_i+2) + b(y_i+1/2) + c$$

$$d_{i+1} - d_i = a \quad \text{cuu duong than cong . com}$$

$$\text{Hay } d_{i+1} = d_i + dy$$

Tính d_1 ?

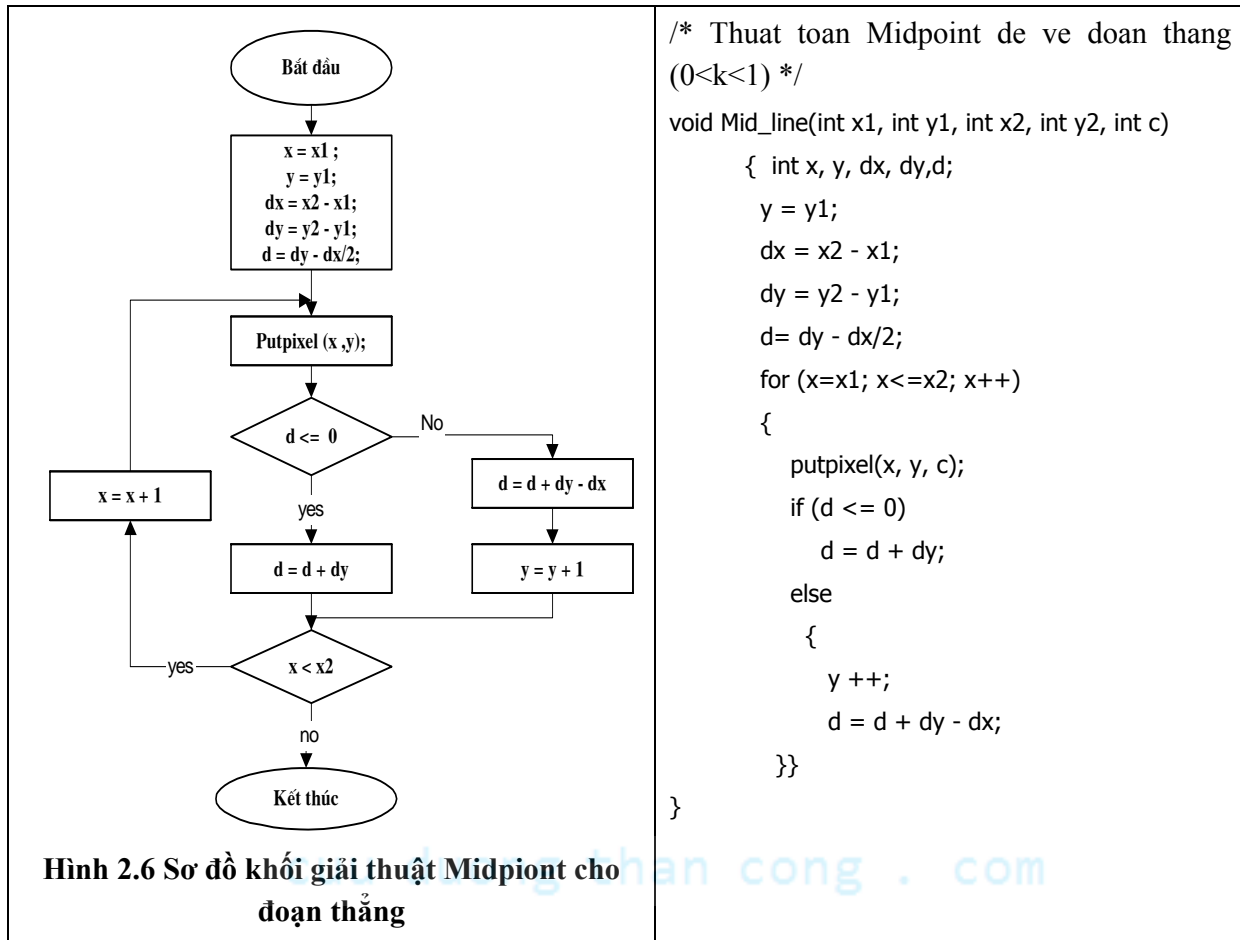
$$d_1 = f(x_1+1, y_1+1/2) = a(x_1+1) + b(y_1+1/2) + c$$

$$= ax_1 + by_1 + c + a + 1/2 b = f(x_1, y_1) + a + b/2$$

Có (x_1, y_1) là điểm bắt đầu, nằm trên đoạn thẳng nên $f(x_1, y_1) = 0$

Vậy $d_1 = a + b/2 = dy - dx/2$

cuu duong than cong . com

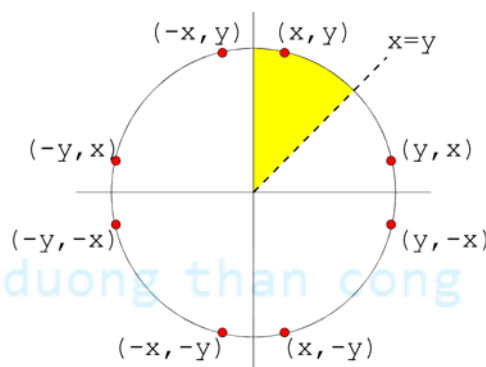


2.5. Giải thuật sinh đường tròn (Scan Converting Circles)(Bresenham)

- Phương trình đường tròn đi qua tâm có tọa độ (x_c, y_c) là:

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

Hình tròn là hình đối xứng tám cách



Hình 2.7 Hình tròn đối xứng 8 phần

Để đơn giản ta xét tâm trùng gốc 0: $x^2 + y^2 = r^2$

Ta xét các điểm tạo ra từ góc phần tư thứ 2: từ 90° đến 45° , thực hiện theo hướng +x, -y

Giả sử bắt đầu x_i vậy $x_{i+1} = x_i + 1$

$$y^2 = r^2 - (x_i + 1)^2$$

$$d_1 = y_i^2 - y^2 = y_i^2 - r^2 - (x_i + 1)^2$$

$$d_2 = y^2 - (y_i - 1)^2 = r^2 - (x_i + 1)^2 - (y_i - 1)^2$$

$$p_i = d_1 - d_2 = 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$$

Xét: $p_i < 0$ ($d_1 < d_2$) chọn điểm nằm ngoài đường tròn $y_{i+1} = y_i$

$p_i \geq 0$ ($d_1 \geq d_2$) chọn điểm nằm trong đường tròn $y_{i+1} = y_i + 1$

$$p_i = 2(x_i + 1)^2 + 2y_i^2 - 2y_i - 2r^2$$

$$p_{i+1} = 2(x_i + 2)^2 + 2y_{i+1}^2 - 2y_{i+1} + 1 - 2r^2$$

$$p_{i+1} = p_i + 4x_i + 6 + 2y_{i+1}^2 - 2y_i^2 - 2y_{i+1} + 2y_i$$

$$p_{i+1} = p_i + 4x_i + 6 + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i)$$

+ Nếu $p_i < 0$ hay $y_{i+1} = y_i$

$$p_{i+1} = p_i + 4x_i + 6$$

+ Nếu $p_i \geq 0$ hay $y_{i+1} = y_i + 1$

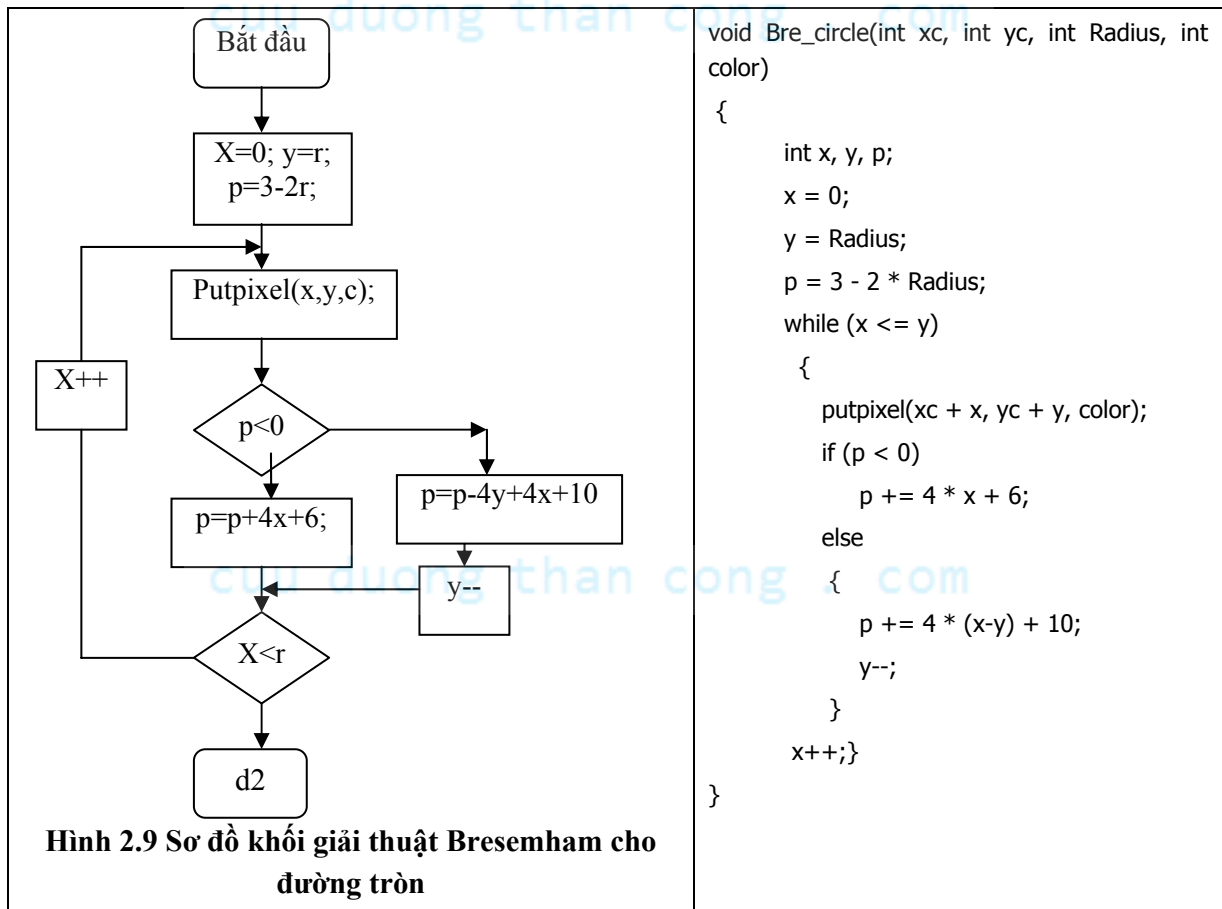
$$p_{i+1} = p_i + 4x_i + 6 - 4y_i + 2 + 2$$

$$p_{i+1} = p_i + 4(x_i - y_i) + 10$$

+ Tính P_1 ? khi đó ứng với $x_1=0$ và $y_1=r$

$$\begin{aligned} p_1 &= 2(x_1 + 1)^2 + y_1^2 + (y_1 - 1)^2 - 2r^2 \\ &= 2 + r^2 + (r-1)^2 - 2r^2 = 3 - 2r \end{aligned}$$

Giải thuật là:



Câu hỏi: lúc sử dụng tính đối xứng cho tám cách để vẽ một đường tròn đầy đủ từ các tọa độ pixel được tạo ứng với góc phần tư thứ hai. Một vài Pixel được vẽ hai lần, hiện tượng này gọi là Overstrike. Hãy chỉ định xem nơi nào xảy ra hiện tượng đó?

Trả lời: Tại $(r,0)$ hoặc $(0,r)$ và vị trí đường chéo: (α, α) trong đó $\alpha = 1/\sqrt{2} \approx 0.7071$

```
/* Thuật toán Bresenham để vẽ đường tròn */
#include <graphics.h>
#include <conio.h>
#define pc(xc,yc,x,y) {
    putpixel(xc + x, yc + y, color);
    putpixel(xc - x, yc - y, color);
    putpixel(xc - y, yc + x, color);
    putpixel(xc + y, yc - x, color);
}
void Bresenham_Circle(int xc, int yc, int Radius, int color){
    int x, y, p;
    x = 0;
    y = Radius;
    p = 3 - 2 * Radius;
    pc(xc,yc, Radius,0); //vẽ 4 điểm đặc biệt
    while (x < y){
        if (d < 0)
            p += 4 * x + 6;
        else{
            p += 4 * (x-y) + 10;
            y--;
        }
        x++;
        pc(xc,yc, x,y);
        pc(xc,yc, y,x);
    }
    pc(xc,yc, y,y); // vẽ 4 điểm phân giác x=y
}
void main(){
    int gr_drive = DETECT, gr_mode;
    initgraph(&gr_drive, &gr_mode, "");
    Bresenham_Circle(getmaxx() / 2, getmaxy() / 2, 150, 4);
    getch();
    closegraph();
}
```

2.7. Giải thuật sinh đường tròn Midpoint

Phương trình đường tròn không tường minh:

$$f(x,y) = x^2 + y^2 - R^2 = 0$$

Nếu $f(x,y) = 0$ thì nằm trên đường tròn

$f(x,y) > 0$ thì nằm bên ngoài đường tròn

$f(x,y) < 0$ thì nằm bên trong đường tròn

Thực hiện giải thuật trên 1/8 đường tròn và lấy đối xứng cho các góc còn lại.

Với M là điểm giữa của AB

Với d_i là giá trị của đường tròn tại một điểm bất kỳ

Ta có: $d_i = f(x_i+1, y_i - 1/2) = (x_i + 1)^2 + (y_i - 1/2)^2 - r^2$

+ $d_i < 0$ chọn A thì điểm kế cận sẽ dịch chuyển theo x một đơn vị

$$\begin{aligned} d_{i+1} &= f(x_i+2, y_i - 1/2) \\ &= (x_i+2)^2 + (y_i - 1/2)^2 - r^2 \\ d_{i+1} - d_i &= (x_i+2)^2 - (x_i+1)^2 = 2x_i+3 \\ d_{i+1} &= d_i + 2x_i+3 \end{aligned}$$

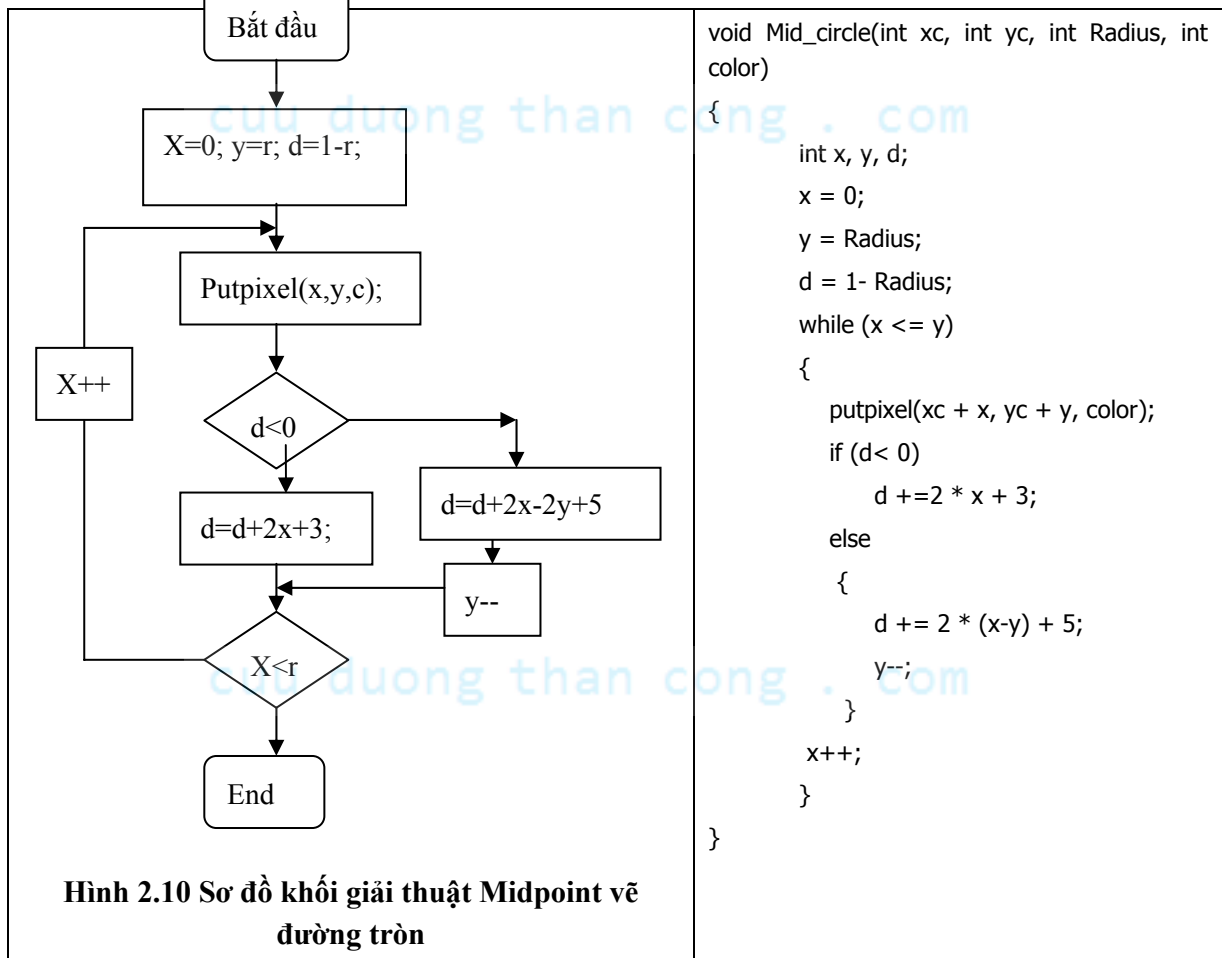
+ $d_i \geq 0$ chọn B thì điểm kế cận sẽ dịch chuyển theo x 1 đơn vị, theo y -1 đơn vị

$$\begin{aligned} d_{i+1} &= f(x_i+2, y_i - 3/2) \\ &= (x_i+2)^2 + (y_i - 3/2)^2 - r^2 \\ d_{i+1} - d_i &= 2x_i - 2y_i + 5 \\ d_{i+1} &= d_i + 2x_i - 2y_i + 5 \end{aligned}$$

Tính d_1 ? tại điểm (0, r)

$$\begin{aligned} d_1 &= f(1, r-1/2) = 1^2 + (r-1/2)^2 - r^2 \\ d_1 &= 5/4 - r \end{aligned}$$

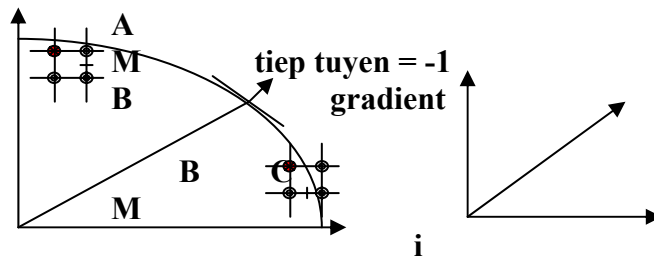
Thuật toán như sau:



2.8. Giải thuật sinh đường ellipse

Tính đối xứng được thực hiện trên 4 cách

$$F(x, y) = b^2x^2 + a^2y^2 - a^2b^2 = 0$$



Hình 2.11 Mô tả giải thuật sinh đường ellipse

Vector \perp với tiếp tuyến gradient = 1

Ta có tiếp tuyến với cung tròn (độ dốc) = -1 = $dy/dx = -f_x/f_y$

Trong đó $f_x = 2b^2x$ đạo hàm riêng phần của $f(x,y)$ với x

Và $f_y = 2a^2y$ đạo hàm riêng phần của $f(x,y)$ với y

Giả sử ta chỉ xét trên góc phần tư thứ nhất: giả sử ta chia cung từ $(0,b)$ đến $(a,0)$ tại Q , có độ dốc -1

Trên phần 1: x thay đổi thì y thay đổi theo

Trên phần 2: y thay đổi thì x thay đổi theo

+ Xét trên phần 1:

Bắt đầu từ $(0,b)$, bước thứ i (x_i, y_i) chọn tiếp

$$A(x_i+1, y_i)$$

$$B(x_i+1, y_i-1)$$

Tham số quyết định:

$$P_i = f(x_i+1, y_i-1/2) = b^2(x_i+1)^2 + a^2(y_i-1/2)^2 - a^2b^2$$

$$P_{i+1} = f(x_{i+1}+1, y_{i+1}-1/2) = b^2(x_{i+1}+1)^2 + a^2(y_{i+1}-1/2)^2 - a^2b^2$$

$$P_{i+1} - P_i = b^2((x_{i+1}+1)^2 - (x_i+1)^2) + a^2((y_{i+1}-1/2)^2 - (y_i-1/2)^2)$$

$$P_{i+1} = P_i + 2b^2x_{i+1} + b^2 + a^2((y_{i+1}-1/2)^2 - (y_i-1/2)^2)$$

- Nếu $P_i < 0$ chọn A

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i$$

$$P_{i+1} = P_i + 2b^2x_i(x_i+1) + b^2 = P_i + 2b^2x_i + 3b^2$$

$$\text{Hay } P_{i+1} = P_i + b^2(2x_i + 3)$$

- Nếu $P_i \geq 0$ chọn B

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i - 1$$

$$P_{i+1} = P_i + 2b^2x_i(x_i+1) + b^2 + a^2((y_i-1-1/2)^2 - (y_i-1/2)^2)$$

$$= P_i + 2b^2x_i + 3b^2 + a^2(-3y_i + 9/4 + y_i - 1/4)$$

$$= P_i + 2b^2x_i + 3b^2 + a^2(-2y_i + 2)$$

$$\text{Hay } P_{i+1} = P_i + b^2(2x_i + 3) + a^2(-2y_i + 2)$$

- Tính P_1 ? tại $(0,b)$

$$P_1 = f(x_1+1, y_1-1/2) = b^2 + a^2(b-1/2)^2 - a^2b^2$$

$$P_1 = b^2 - a^2b + a^2/4$$

+ Xét trên phần 2:

Ta lấy toạ độ của Pixel sau cùng trong phần 1 của đường cong để tính giá trị ban đầu cho phần 2.

Giả sử pixel (x_k, y_k) vừa chuyển quét cuối cùng của phần 1 nhập vào bước j cho phần 2 (x_j, y_j) .

Pixel kế tiếp có thể là:

$$C(x_j, y_j-1)$$

$$D(x_j+1, y_j-1)$$

Tham số quyết định:

$$q_j = f(x_j+1/2, y_j-1) = b^2(x_j+1/2)^2 + a^2(y_j-1)^2 - a^2b^2$$

$$q_{j+1} = f(x_{j+1}+1/2, y_{j+1}-1) = b^2(x_{j+1}+1/2)^2 + a^2(y_{j+1}-1)^2 - a^2b^2$$

$$q_{j+1} - q_j = b^2((x_{j+1}+1/2)^2 - (x_j+1/2)^2) + a^2((y_{j+1}-1)^2 - (y_j-1)^2)$$

$$q_{j+1} = q_j + b^2((x_{j+1}+1/2)^2 - (x_j+1/2)^2) + a^2 - 2a^2y_{j+1}$$

- Nếu $q_j < 0$ chọn D

$$y_{j+1} = y_j - 1$$

$$x_{j+1} = x_j + 1$$

$$q_{j+1} = q_j + b^2((x_j+3/2)^2 - (x_j+1/2)^2) + a^2 - 2a^2(y_j - 1)$$

$$q_{j+1} = q_j + b^2(3x_j + 9/4 - x_j - 1/4) + 3a^2 - 2a^2y_j$$

$$\text{Hay } q_{j+1} = q_j + b^2(2x_j + 2) + a^2(-2y_j + 3)$$

- Nếu $q_j \geq 0$ chọn C

$$y_{j+1} = y_j - 1$$

$$x_{j+1} = x_j$$

$$q_{j+1} = q_j + a^2 - 2a^2(y_j - 1)$$

$$\text{Hay } q_{j+1} = q_j + a^2(3 - 2y_j)$$

- Tính q_1 ?

$$q_1 = f(x_k+1/2, y_k-1) = b^2(x_k+1/2)^2 + a^2(y_k-1)^2 - a^2b^2$$

Câu hỏi: lúc lấy đối xứng 4 cách để tìm 1 Ellipse hoàn chỉnh từ các toạ độ pixel được tạo ra với cung phần tư thứ 1. Có hiện tượng overstrike xảy ra hay không?

Trả lời: hiện tượng overstrike xảy ra tại:

$$(0,b); (0,-b); (a,0); (-a,0)$$

Thuật toán

```
#include <graphics.h>
```

```
#include <conio.h>
```

```
#define ROUND(a) ((long)(a+0.5))
```

```
void plot(int xc, int yc, int x, int y, int color){
    putpixel(xc+x, yc+y, color);
```

```

        putpixel(xc-x, yc+y, color);
        putpixel(xc+x, yc-y, color);
        putpixel(xc-x, yc-y, color);
    }
    void Mid_ellipse(int xc, int yc, int a, int b, int color){
        long x, y, fx, fy, a2, b2, p;
        x = 0;
        y = b;
        a2 = a * a; //a2
        b2 = b * b; //b2
        fx = 0;
        fy = 2 * a2 * y; // 2a2y
        plot(xc, yc, x, y, color);
        p = ROUND(b2-(a2*b)+(0.25*a)); // p=b2 - a2b + a2/4
        while (fx < fy){
            x++;
            fx += 2*b2; //2b2
            if (p<0)
                p += b2*(2*x + 3); // p=p + b2 (2x + 3)
            else{
                y--;
                p+= b2*(2*x + 3) + a2*(-2*y + 2); // p = p + b2(2x + 3) + a2 (-2y + 2)
                fy -= 2*a2; // 2a2
            }
            plot(xc, yc, x, y, color);
        }
        p = ROUND(b2*(x+0.5)*(x+0.5) + a2*(y-1)*(y-1) - a2*b2); //b2(x+1/2)2+a2(y-1)2 - a2b2
        while (y>0){
            y--;
            fy -= 2*a2; // 2a2
            if (p>=0)
                p+=a2*(3 - 2*y); p = p + a2(3-2y)
            else{
                x++;
                fx += 2*b2; // 2b2
                p += b2*(2*x+2) + a2*(-2*y + 3); //p=p + b2(2x + 2) + a2(-2y + 3)
            }
            plot(xc, yc, x, y, color);
        }
    }
}

void main(){
    int gr_drive = DETECT, gr_mode;
    initgraph(&gr_drive, &gr_mode, "");
    Mid_Ellipse(getmaxx() / 2, getmaxy() / 2, 150, 80, 4);
    getch();
    closegraph();
}

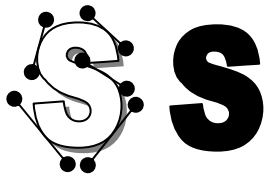
```

2.9. Giải thuật sinh ký tự

Trong màn hình text, truy xuất các ký tự trên màn hình được hỗ trợ bởi phần cứng. Các ký tự được lưu trữ trong bộ nhớ ROM, dưới dạng bitmap hay các ma trận ảnh. Phần cứng sẽ đưa ký tự lên màn hình tại vị trí xác định, tính toán cuộn trang và xuống dòng.

Trong đồ hoạ:

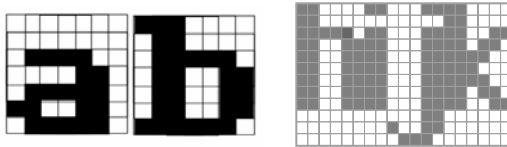
+ Vector: định nghĩa các ký tự theo những đường cong mềm bao ngoài của chúng, tốn kém về mặt tính toán.



- phức tạp (tính toán phương trình)
- lưu trữ gọn nhẹ
- các phép biến đổi dựa vào công thức biến đổi
- Kích thước phụ thuộc vào môi trường (không có kích thước cố định)

Hình 2.12 Ký tự vector

+ Bitmap: định nghĩa mỗi ký tự với 1 font chữ cho trước là 1 ảnh bitmap hình chữ nhật nhỏ.



- Đơn giản trong việc sinh ký tự (copypixel)
- Lưu trữ lớn
- Các phép biến đổi (I,B,U, scale) đòi hỏi lưu trữ thêm
- Kích thước không đổi

Hình 2.13 Ký tự bitmap

+ bitmap: sử dụng hàm copypixel (copy điểm ảnh) được lưu trữ trong bộ nhớ cố định - Fontcache, đưa vào bộ nhớ đệm hiển thị. Mỗi 1 ký tự như 1 ma trận 2 chiều của các điểm ảnh - mặt nạ.

```
Hàm_sinh_ki_tu (mask)
{
    xmax, ymax, xmin, ymin //các giới hạn của mặt nạ
    xo, yo//điểm gốc trên bộ đệm hiển thị
    for (i=ymin; i< ymax ;i++)
        for (j=xmin; j< xmax ; j++)
            if (mask(i,j) <> 0)
                copypixel ((mask(i,j), pixel(xo+j, yo+i));
    }
}
```

Ký tự fontcache bitmap đơn giản của SRGP lưu trữ các ký tự theo chuỗi liên tiếp nhau trong bộ nhớ. Nhưng độ rộng các ký tự khác nhau, truy nhập các fontcache thông qua bản ghi về cấu trúc cho từng ký tự.

Cấu trúc font chữ

```
typedef struct {
    int leftx;
    int width;
} Charlocation; //Vị trí của text

struct {
    int CacheId;
    int Height; // Độ rộng chữ
    int CharSpace; // Khoảng cách giữa các ký tự
    Charlocation Table [128]; //bảng ch ci
} fontcache;
```

+ Ký tự vector

Xây dựng theo phương pháp định nghĩa các ký tự bởi đường cong mềm bao ngoài của chúng dễ dàng thay đổi kích thước của ký tự cũng như nội suy ra các dạng của ký tự. Hoàn toàn độc lập với thiết bị.

+ Tối ưu nhất: lưu trữ font dưới dạng đường bao. Khi các chương trình ứng dụng sử dụng là bitmap tương ứng với chúng.

2.10. Giải thuật sinh đa giác (Polygon)

a. Thuật giải vẽ đường bao đa giác

Việc biểu diễn đa giác thông qua:

- Tập các đoạn thẳng
- Tập các điểm thuộc đa giác

Các loại đa giác:



triangular convex non-convex self-intersecting religious

Hình 2.14 Các loại đa giác

Đa giác lõm: là đa giác có đường thẳng nối bất kỳ 2 điểm bên trong nào của đa giác đều nằm trọn trong đa giác. Đa giác không lõm là đa giác lồi.

Các đường thẳng bao đa giác - cạnh của đa giác. Các điểm giao của cạnh - đỉnh của đa giác. Thông tin cần thiết để xác định đa giác:

- Số cạnh
- Tọa độ các đỉnh của đa giác

Giải thuật:

```
Polygon (arrayx, arrayy, n)
{ if (n < 3 // không phải đa giác
  exit;
  for (i = 1 ; i <= n - 1; i++)
    line(arrayx[i], arrayy[i], arrayx[i + 1], arrayy[i + 1]);
    line(arrayx[i + 1], arrayy[i + 1], arrayx[1], arrayy[1]);
}
```

b. Các thuật toán tô miền kín đa giác

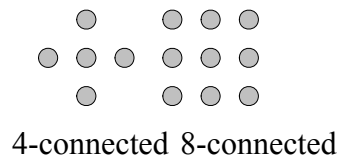
Lợi thế của hiển thị raster là: khả năng lưu trữ, copy, tô màu một vùng... Có hai dạng vùng tô thường gặp đó là: tô bằng một màu thuần nhất (solid fill), tô theo mẫu tô (fill pattern) nào đó.

Còn thiết bị vector thì hạn chế do các vùng tô màu tạo ra bởi một tập các đoạn thẳng sát nhau - làm chậm quá trình làm tươi.

- + Giải thuật đường biên (Boundary - fill Algorithm)
- Bắt đầu từ 1 điểm (x,y) trong vùng cần được tô màu:
 - + Xác định màu điểm: getpixel(x,y,c)
 - + Tô màu putpixel(x,y,c)

- Bước tiếp: kiểm tra thuộc tính màu các điểm lân cận
 - + điểm lân cận đã tô màu (exit)
 - + trùng với màu đường biên(exit)
 - + Nếu không thì tô màu

Các phương pháp xác định điểm lân cận



Hình 2.15 Phương pháp tịnh tiến giải thuật

Giải thuật tô màu đường biên:

```
#include <graphics.h>
#include <conio.h>
void FloodFill (int x, int y, int in_color, int new_color){
    if (getpixel(x, y) == in_color){
        putpixel(x, y, new_color);
        FloodFill(x-1, y, in_color, new_color);
        FloodFill(x+1, y, in_color, new_color);
        FloodFill(x, y-1, in_color, new_color);
        FloodFill(x, y+1, in_color, new_color);
    }
}
void main(){
    int gr_drive = DETECT, gr_mode;
    initgraph(&gr_drive, &gr_mode, "");
    circle(getmaxx() / 2, getmaxy() / 2, 15);
    FloodFill(getmaxx() / 2, getmaxy() / 2, 0, 4);
    getch();
    closegraph();
}
```

+Giải thuật dòng quét (scanline) cho việc tô màu vùng

Giải thuật dựa trên ý tưởng sử dụng một đường quét trên trục y của màn hình đi từ y_{\max} đến y_{\min} của vùng cần được tô màu.

Với mỗi giá trị $y = y_i$ đường thẳng quét cắt các đường biên của vùng cần tô tạo ra đoạn thẳng $y = y_i$ với $x \in [x_{\min}, x_{\max}]$. Trên đoạn thẳng đó chúng ta tô màu các điểm tương ứng đi từ x_{\min} đến x_{\max} có các điểm tô $(x_i, y_i) \in y = y_i$.

- Đơn giản nhất ví dụ tô màu hình chữ nhật:

```
void scanline_rectg(x1,y1,x2,y2,c){ int i,j;
    for(i=y1; i>=y2; i--){
        for(j=x1; j<= x2;j++)
            putpixel(i,j,c);
    }
}
```

- Phép tô màu 1 đa giác bất kỳ sẽ phức tạp hơn rất nhiều so với hình chữ nhật

Giả sử vùng tô được cho bởi 1 đa giác n đỉnh: $p_i (x_i, y_i)$, $i=0,1,...,n-1$. Đa giác này có thể là đa giác lồi, đa giác lõm hay đa giác tự cắt....

Các bước tóm tắt chính của thuật toán:

- Tìm y_{top} , y_{bottom} lần lượt là giá trị lớn nhất, nhỏ nhất của tập các tung độ của các đỉnh của đa giác đã cho.

$$y_{top} = \max \{y_i, (x_i, y_i) \in P\},$$

$$y_{bottom} = \min \{y_i, (x_i, y_i) \in P\}.$$

- Ứng với mỗi dòng quét $y=k$, với k thay đổi từ y_{bottom} đến y_{top} lặp:

+ Tìm tất cả các hoành độ giao điểm của dòng quét $y=k$ với các cạnh của đa giác

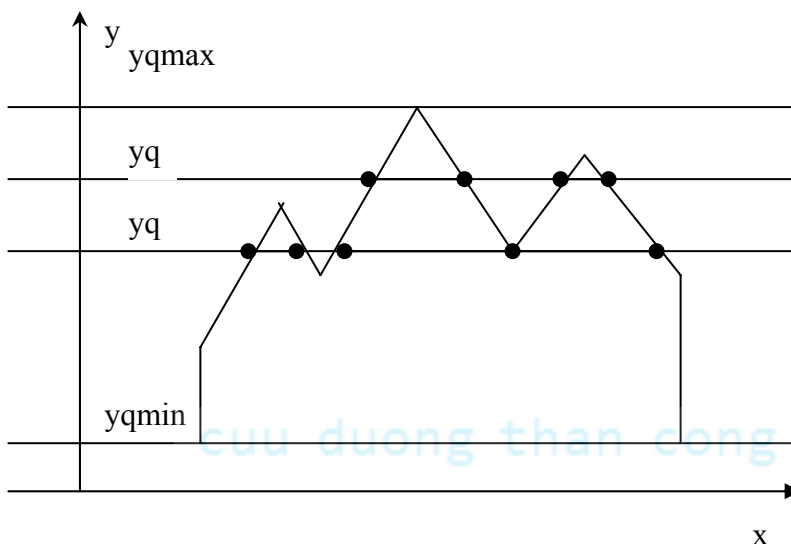
+ Sắp xếp các hoành độ giao điểm theo thứ tự tăng dần: $x_0, x_1, ...$

+ Tô màu các đoạn thẳng trên đường thẳng $y=k$ lần lượt được giới hạn bởi các cặp (x_0, x_1) , (x_2, x_3) , ..., (x_{2k}, x_{2k+1}) .

Chúng ta sẽ gặp 1 số vấn đề sau:

- Ứng với mỗi dòng quét không phải lúc nào tất cả các cạnh của đa giác cũng tham gia cắt dòng quét. Do đó để cải thiện tốc độ cần phải có một cách nào đó để hạn chế được số cạnh cần tìm giao điểm ứng với mỗi dòng quét.

- Nếu số giao điểm tìm được giữa các cạnh đa giác và dòng quét là lẻ (điều này chỉ xảy ra khi dòng quét sẽ đi qua các đỉnh của đa giác) khi đó ta sẽ tính số điểm là 2 thì có thể tô không chính xác. Ngoài ra, việc tìm giao điểm của dòng quét với các cạnh nằm ngang là trường hợp đặt biệt...



Hình 2.16 Giải thuật scanline cho một đa giác bất kỳ

Để giải quyết các vấn đề trên ta có các phương pháp sau:

+ Danh sách các cạnh kích hoạt (AET - Active Edge Table)

Mỗi cạnh của đa giác được xây dựng từ 2 đỉnh kề nhau $P_i(x_i, y_i)$ và $P_{i+1}(x_{i+1}, y_{i+1})$ gồm các thông tin sau:

y_{min} : giá trị nhỏ nhất trong 2 đỉnh của cạnh

$x_{\text{Intersect}}$: hoành độ giao điểm của cạnh với dòng quét hiện đang xét

$DxPerScan$: giá trị $1/m$ (m là hệ số góc của cạnh)

Δy : khoảng cách từ dòng quét hiện hành tới đỉnh y_{max}

Danh sách các cạnh kích hoạt AET: danh sách này dùng để lưu các tập cạnh của đa giác có thể cắt ứng với dòng quét hiện hành và tập các điểm giao tương ứng. Nó có một số đặc điểm:

Các cạnh trong danh sách được sắp xếp theo thứ tự tăng dần của các hoành độ giao điểm để có thể tô màu các đoạn giao một cách dễ dàng.

Thay đổi ứng với mỗi dòng quét đang xét, do đó danh sách này sẽ được cập nhật liên tục trong quá trình thực hiện thuật toán. Đầu tiên ta có danh sách chứa toàn bộ các cạnh của đa giác gọi là ET (Edge Table) được sắp xếp theo thứ tự tăng dần của y_{min} , rồi sau mỗi lần dòng quét thay đổi sẽ di chuyển các cạnh trong ET thỏa điều kiện sang AET.

Một dòng quét $y=k$ chỉ cắt 1 cạnh của đa giác khi và chỉ khi $k \geq y_{\text{min}}$ và $\Delta y > 0$. Chính vì vậy mà với các tổ chức của ET (sắp theo thứ tự tăng dần của y_{min}) điều kiện để chuyển các cạnh từ ET sang AET sẽ là $k \geq y_{\text{min}}$; và điều kiện để loại một cạnh ra khỏi AET là $\Delta y \leq 0$

+ Công thức tìm giao điểm nhanh

Nếu gọi x_k, x_{k+1} lần lượt là các hoành độ giao điểm của một cạnh nào đó với các dòng quét $y=k$ và $y=k+1$ ta có:

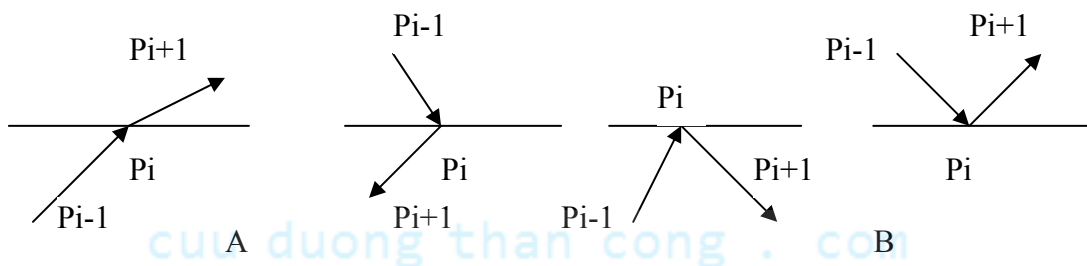
$$x_{k+1} - x_k = 1/m ((k+1) - k) = 1/m \text{ hay } x_{k+1} = x_k + 1/m$$

Như vậy nếu lưu hoành độ giao điểm ứng với dòng quét trước lại, cùng với hệ số góc của cạnh, ta xác định được hoành độ giao điểm ứng với dòng quét kế tiếp theo công thức trên. Nên thông tin của cạnh có 2 biến: $DxPerScan$, $x_{\text{Intersect}}$.

+ Trường hợp dòng quét đi ngang qua một đỉnh:

Tính 1 giao điểm nếu chiều của 2 cạnh kề của đỉnh đó có xu hướng tăng hay giảm

Tính 2 giao điểm nếu chiều của 2 cạnh kề của đỉnh đó có xu hướng thay đổi, nghĩa là tăng-giảm hay giảm-tăng.



Hình 2.17 Qui tắc tính: một giao điểm (A) và hai giao điểm (B)

+ Giải thuật tô vùng kín theo mẫu (Pattern filling)

object (ảnh mẫu)

$A[m,n]$

Vấn đề: xác định điểm ở mẫu và nhiều điểm tương ứng với chúng trên màn hình.

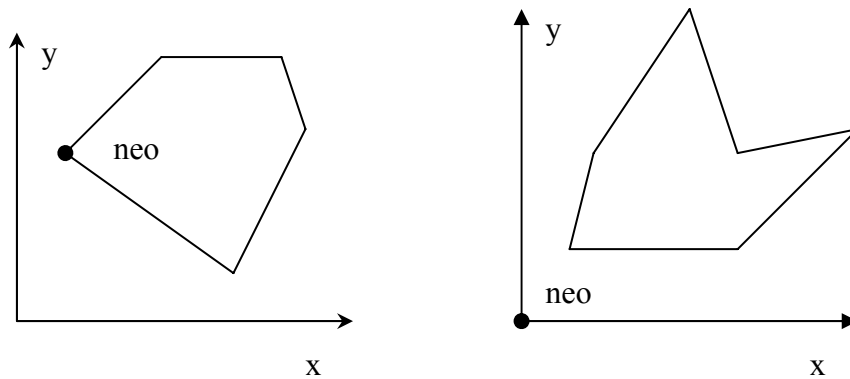
Phương pháp 1:

Tìm điểm neo ở đầu trái nhất hàng đầu tiên của đa giác.

Nhược điểm: không có điểm định vị trí phân biệt một cách rõ ràng cho mẫu tô trong một đa giác bất kỳ.

Phương pháp 2: sử dụng cho SRGP

Lấy điểm neo ở gốc toạ độ, giả sử ta coi cả màn hình được lát bởi màu tô các thực thể là các đường biên cho các vùng tô, vậy nếu ngoài các thực thể các màu tô không được phép thể hiện.



Hình 2.18 Phương pháp lấy điểm neo

Tóm tắt chương:

Các đối tượng đồ hoạ cơ sở cung cấp các công cụ cơ bản nhất cho việc xây dựng các ảnh đồ hoạ của các đối tượng phức tạp. Các đoạn thẳng, đường cong, vùng tô, ký tự....là các đối tượng đồ hoạ cơ sở được hầu hết tất cả các công cụ lập trình đồ hoạ hỗ trợ.

Để có thể hiển thị các đối tượng đồ hoạ trên thiết bị hiển thị dạng điểm mà điển hình là màn hình, cần phải có một quá trình chuyển các mô tả hình học của các đối tượng này trong hệ toạ độ thế giới thực về dãy các pixel tương ứng gần với chúng nhất trên toạ độ thiết bị. Quá trình này còn được gọi là quá trình chuyển đổi bằng dòng quét. Yêu cầu quan trọng nhất đối với quá trình này ngoài việc phải cho kết quả xấp xỉ tốt nhất còn phải cho tốc độ tối ưu.

Ba cách tiếp cận để vẽ đoạn thẳng gồm thuật toán DDA, thuật toán Bresenham, thuật toán Midpiont đều tập trung vào việc đưa ra cách chọn một trong hai điểm nguyên kế tiếp khi đã biết điểm nguyên ở bước trước. Thuật toán DDA đơn giản chỉ dùng thao tác làm tròn nên phải dùng các phép toán trên số thực, trong khi đó thuật toán Bresenham và Midpiont đưa ra cách chọn phức tạp hơn nhưng cho kết quả tốt hơn. Tương tự dùng hai giải thuật Bresenham và Midpiont để vẽ đường tròn và ellipse và một số đường cong khác.

Các thuật toán tô màu vùng gồm thuật toán loang (đệ qui) hay thuật toán dòng quét. Có thể tô cùng một màu hay tô theo mẫu.

Bài tập:

1. Chỉ định các vị trí mảnh nào sẽ được chọn bởi thuật toán Bresenham lúc chuyển quét một đường thẳng từ toạ độ pixel (1,1) sang toạ độ pixel (8,5).
2. Dùng giải thuật Bresenham viết hàm sinh đoạn thẳng (xét tất cả các trường hợp của hệ số góc).
3. Dùng giải thuật Midpiont viết hàm sinh đoạn thẳng (xét tất cả các trường hợp của hệ số góc).

4. Dùng giải thuật Midpiont viết hàm sinh đường tròn (toạ độ tâm (xc,yc) và bán kính r).
5. Dùng giải thuật Midpiont viết hàm sinh đường ellipse (toạ độ tâm (xc,yc) và bán kính rx và ry).
6. Từ hàm vẽ đường thẳng thiết kế và cài đặt hàm vẽ các hình sau: hình chữ nhật, đa giác, ngôi nhà....
7. Viết giải thuật tìm giao điểm hai đoạn thẳng.
8. Viết chương trình tô màu Floodfill (sử dụng đệ qui với 4-connected).
9. Đưa ra lưu đồ thuật toán tô màu theo dòng quét.
10. Viết thuật toán tô màu scan-line.

Bài tập trắc nghiệm:





1. Xây dựng giải thuật tổng quát để vẽ đường thẳng ta có xét hệ số k (hệ số góc của đường) sẽ có tất cả các trường hợp của k:
 - a. 4
 - b. 3
 - c. 2
 - d. 1
2. Để biểu diễn đoạn thẳng thông qua phương trình tham số như sau:
 - a. $y=f(x)$ hay $y=kx+b$
 - b. $f(x,y)=0$ hay $ax + by + c = 0$
 - c. $x(v)=x_1 +v(x_2-x_1)$ và $y(v) = y_1 +v(y_2-y_1)$ có $v \in [0,1]$
 - d. $P(u) = P_1 + u(P_2-P_1)$ có $u \in [0,1]$
3. Khi xây dựng giải thuật vẽ đường tròn đầy đủ ta chỉ cần viết phương trình cho 1/8 đường tròn, rồi gọi đối xứng 8 cách. Khi đó xảy ra hiện tượng overstrike. Vậy điểm xảy ra hiện tượng đó là: (r là bán kính của đường tròn)
 - a. (0,r) và $\left(\frac{1}{2}r, \frac{1}{2}r\right)$
 - b. (r,0) hoặc (0,r) và $\left(\frac{1}{\sqrt{2}}r, \frac{1}{\sqrt{2}}r\right)$
 - c. (0,r) và (-r,0)
 - d. (r,0) và (0,r)
4. Giải thuật sau là giải thuật gì? Và đã dùng bao nhiêu điểm lân cận?

```
void Function (int x, int y, int c1, int c2){
    if (getpixel(x, y) == c1){
        putpixel(x, y, c2);
        Function (x-1, y, c1, c2);
        Function (x+1, y, c1, c2);
        Function (x+1, y+1, c1, c2);
        Function (x-1, y-1, c1, c2);
        Function (x, y-1, c1, c2);
    }
```

```

    Function (x, y+1, c1, c2);
  }
}

```

Giải thuật scanline, số điểm lân cận là:	Giải thuật tô màu đường biên, số điểm lân cận là:	Giải thuật tô màu loang, số điểm lân cận là:	Giải thuật tô màu theo mẫu, số màu lân cận là:
			
A	B	c	d

5. Giải thuật vẽ đường thẳng sau vẽ cho trường hợp k là:

```

void Midline(int x1,int y1,int x2,int y2,int c){
  int x=x1,y=y1,dx=x2-x1,dy=y2-y1,p=2*dx-dy;
  while(y<y2) {
    putpixel(x+320,240-y,c);
    if(p<=0){
      p=p+2*dx;
    }
    else{
      p=p+2*dx-2*dy;
      x++;
    }
    y++;
  }
}

```

- $0 \leq k \leq 1$
- $k > 1$
- $k \leq -1$
- $-1 < k < 0$

6. Giải thuật vẽ đường thẳng sau vẽ cho trường hợp k là:

```

void Midline(int x1,int y1,int x2,int y2,int c){
  int x=x1,y=y1,dx=x2-x1,dy=y2-y1,p=2*dy+dx;
  while(x<x2) {
    putpixel(x+320,240-y,c);
    if(p<=0){
      p=p+2*dy+2*dx;
      y--;
    }
    else{
      p=p+2*dy;
    }
    x++;
  }
}

```

- $k \geq 0$ và $k \leq 1$
- $k > 1$
- $k \leq -1$
- $-1 < k < 0$

7. Giải thuật sau là giải thuật nào?

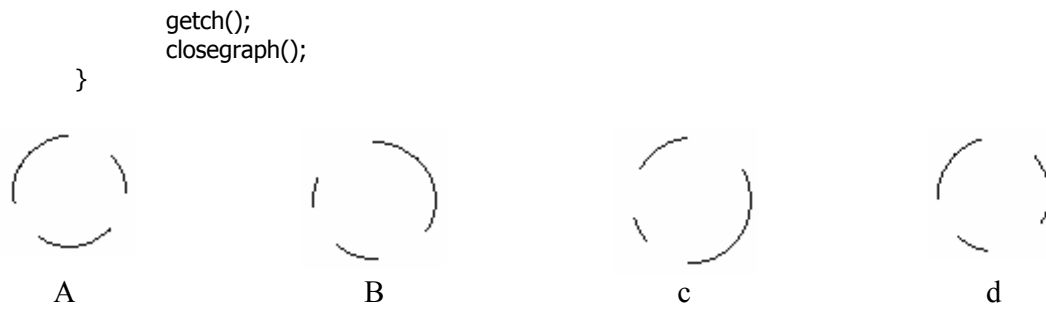
```
void Function(int xt, int yt, int r, int c){
    int x, y, d;
    x = 0;
    y = r;
    d = 3 - 2 * r;
    while (x <= y){
        putpixel(xt + x, yt + y, c);
        putpixel(xt - x, yt + y, c);
        putpixel(xt + x, yt - y, c);
        putpixel(xt - x, yt - y, c);
        putpixel(xt + y, yt + x, c);
        putpixel(xt - y, yt + x, c);
        putpixel(xt + y, yt - x, c);
        putpixel(xt - y, yt - x, c);
        if (d < 0)
            d += 4 * x + 6;
        else{
            d += 4 * (x-y) + 10;
            y--;
        }
        x++;
    }
}
```

- a. Giải thuật Bresenham xây dựng đường tròn
- b. Giải thuật Midpoint xây dựng đường tròn
- c. Giải thuật Bresenham xây dựng đường ellipse
- d. Giải thuật Midpiont xây dựng đường ellipse

8. Chương trình sau đưa gì ra hình?

```
#include <graphics.h>
#include <conio.h>
void Function(int xc, int yc, int r, int c){
    int x, y, d;
    x = 0;
    y = r;
    d = 3 - 2 * r;
    while (x <= y){
        putpixel(xc + x, yc + y, c);
        putpixel(xc - x, yc + y, c);
        putpixel(xc - x, yc - y, c);
        putpixel(xc + y, yc - x, c);
        putpixel(xc - y, yc - x, c);
        if (d < 0)
            d += 4 * x + 6;
        else{
            d += 4 * (x-y) + 10;
            y--;
        }
        x++;
    }
}

void main(){
    int gr_drive = DETECT, gr_mode;
    initgraph(&gr_drive, &gr_mode, "");
    Function(getmaxx() / 2, getmaxy() / 2, 150, 4);
}
```



cuu duong than cong . com

cuu duong than cong . com

CHƯƠNG 3: CÁC PHÉP BIẾN ĐỔI ĐỒ HOẠ

1. CÁC PHÉP BIẾN ĐỔI HÌNH HỌC HAI CHIỀU

1.1. Phép biến đổi Affine (Affine Transformations)

Phép biến đổi Affine là phép biến đổi tuyến tính tọa độ điểm đặc trưng của đối tượng thành tập tương ứng các điểm mới để tạo ra các hiệu ứng cho toàn đối tượng.

Ví dụ: phép biến đổi tọa độ với chỉ 2 điểm đầu cuối của đoạn thẳng tạo thành 2 điểm mới mà khi nối chúng với nhau tạo thành đoạn thẳng mới. Các điểm nằm trên đoạn thẳng sẽ có kết quả là điểm nằm trên đoạn thẳng mới với cùng phép biến đổi thông qua phép nội suy.

1.2. Các phép biến đổi đối tượng

Các đối tượng phẳng trong đồ họa 2 chiều mô tả tập các điểm phẳng. Điểm trong đồ họa 2 chiều biểu diễn thông qua tọa độ, viết dưới dạng ma trận gọi là vectơ vị trí.

Có 2 dạng biểu diễn:

Một hàng và 2 cột: $[x \quad y]$

Hai hàng và 1 cột: $\begin{bmatrix} x \\ y \end{bmatrix}$

Trong tài liệu này chúng ta chọn biểu diễn điểm là ma trận hàng (một hàng và 2 cột).

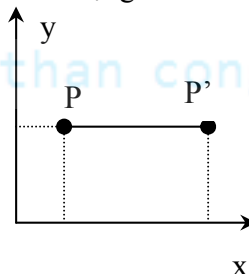
Tập các điểm được lưu trữ trong máy tính sẽ được viết dưới dạng ma trận vị trí của chúng. Chúng có thể là đường thẳng, đường cong, ảnh...thật dễ dàng kiểm soát các đối tượng thông qua các phép biến đổi chúng, thực chất các phép biến đổi đồ họa này được mô tả dưới dạng các ma trận.

1.2.1. Phép biến đổi vị trí

Giả sử ta có điểm $P = [x \quad y]$ trong mặt phẳng với $[x \quad y]$ là vectơ vị trí của P, kí hiệu là $[X]$.

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Gọi ma trận T là ma trận biến đổi sẽ có dạng:



Hình 3.1 Phép biến đổi vị trí

Ta có điểm P sau phép biến đổi thành P' có giá trị $[x' \quad y']$.

Thực thi phép biến đổi đúng trên 1 điểm ảnh sẽ đúng với toàn bộ đối tượng.

$$[X]*[T]=[x \ y]*\begin{bmatrix} a & b \\ c & d \end{bmatrix}=[(ax+cy) \ (bx+dy)]=[x' \ y']$$

Hay ta có: $x' = ax + cy$

$$y' = bx + dy$$

Xét ma trận biến đổi T:

① Phép bất biến:

Khi đó: $a = d = 1$ và $b = c = 0$ và ma trận cho phép bất biến là:

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

$$[X]*[T]=[x \ y]*\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}=[x \ y]=[x' \ y']$$

Vậy $x' = x$ và $y = y'$ hay là $P' = P$ chứng tỏ bất biến qua phép biến đổi.

② Phép biến đổi tỷ lệ (scaling):

- Nếu $d=1$ và $b = c = 0$ thì ma trận biến đổi là:

$$T = \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix}$$

$$x' = ax$$

$$y' = y$$

P' dịch chuyển theo trục x với tỷ lệ a xác định.

$$[X]*[T]=[x \ y]*\begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix}=[(ax) \ y]=[x' \ y']$$

- Nếu $b = c = 0$ thì ma trận biến đổi là:

$$T = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$$

$$[X]*[T]=[x \ y]*\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}=[ax \ dy]=[x' \ y']$$

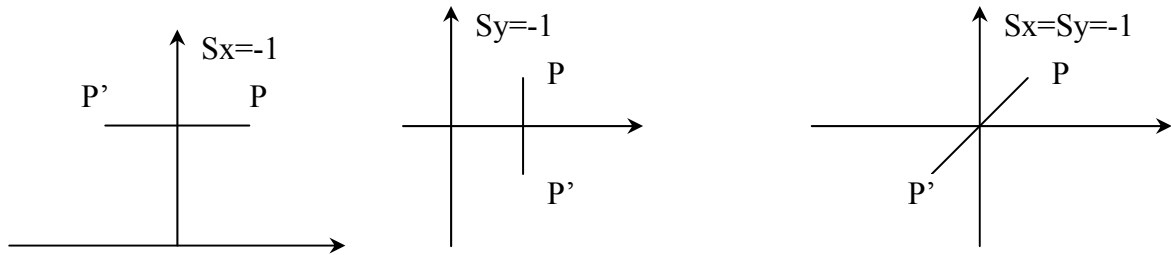
Hay tổng quát hơn gọi S_x, S_y lần lượt là tỷ lệ theo trục x và trục y, thì ma trận tỷ lệ sẽ là:

$$T = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

Khi $S_x, S_y > 1$ gọi là phép phóng to

Khi $S_x, S_y < 1$ gọi là phép thu nhỏ

- Các trường hợp đặc biệt:



Đối xứng qua y

Đối xứng qua x

Đối xứng qua gốc tọa độ

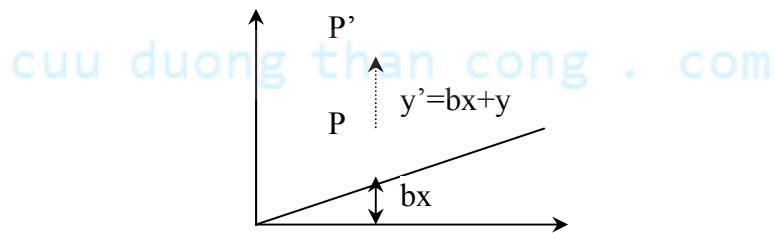
Hình 3.2 Các phép đối xứng trên 2D

③ Phép biến dạng

Khi $a = d = 1$ thì tọa độ của P' phụ thuộc vào thay đổi của b và c

- Xét $c = 0$

$$[X] * [T] = [x \ y] * \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} = [x \ bx + y] = [x' \ y']$$

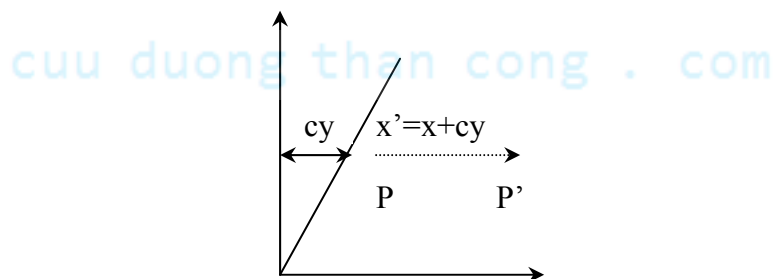


Hình 3.3 Phép biến dạng theo trục oy

Có P' không thay đổi giá trị tọa độ x , còn y' thay đổi phụ thuộc vào cả b và x

- Xét $b = 0$

$$[X] * [T] = [x \ y] * \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix} = [x + cy \ y] = [x' \ y']$$

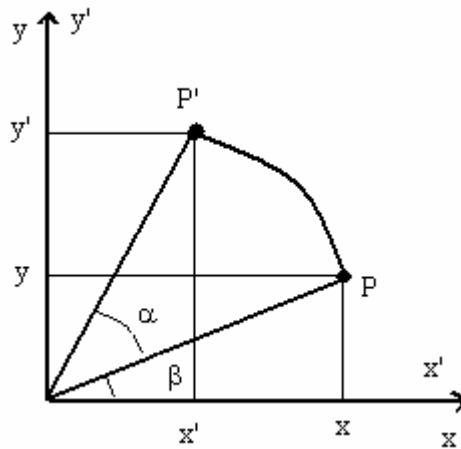


Hình 3.4 Phép biến dạng theo trục ox

Chú ý: điểm gốc tọa độ $P[0 \ 0]$ bất biến với mọi phép biến đổi

④ Phép quay

Có $\alpha > 0$ ngược chiều kim đồng hồ



Hình 3.5 Phép quay trên 2D

Ta có:

$$P[x \ y] = [r \cos \beta \ r \sin \beta]$$

$$P'[x' \ y'] = [r \cos(\alpha + \beta) \ r \sin(\alpha + \beta)]$$

$$P'[x' \ y'] = [r(\cos \alpha \cos \beta - \sin \alpha \sin \beta) \ r(\cos \alpha \sin \beta + \sin \alpha \cos \beta)]$$

$$= [(x \cos \alpha - y \sin \alpha) \ (x \sin \alpha + y \cos \alpha)]$$

Vậy: $x' = x \cos \alpha - y \sin \alpha$

$$y' = x \sin \alpha + y \cos \alpha$$

Ta có:

$$[X'] = [X] * [T] = [(x \cos \alpha - y \sin \alpha) \ (x \sin \alpha + y \cos \alpha)]$$

Vậy T tổng quát khi quay đối tượng quanh gốc tọa độ 1 góc α bất kỳ là:

$$[T] = \begin{bmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{bmatrix}$$

1.2.2. Phép biến đổi tổng hợp

Phương pháp biến đổi sử dụng phép nhân ma trận với tọa độ điểm thông qua các vector vị trí thật sự hiệu quả và đem lại công cụ mạnh về đồ họa cho người sử dụng. Nhưng thực tế các thao tác thường cần không chỉ một mà nhiều phép biến đổi khác nhau. Ta có phép hoán vị khi nhân ma trận là không thực hiện nhưng khả năng tổ hợp các phép nhân lại cho phép tạo ra một ma trận biến đổi duy nhất. Làm giảm bớt đáng kể khối lượng tính toán trong quá trình biến đổi, làm tăng tốc các chương trình ứng dụng và tạo điều kiện cho việc quản lý các biến đổi trong ứng dụng.

Giả sử ta có P với $[X] = [x \ y]$, có hai phép biến đổi $[T1]$ quay quanh gốc tọa độ 90° :

$$T1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Và $[T2]$ lấy đối xứng P qua gốc tọa độ:

$$T2 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Ta có:

$$[X'] = [X] * [T1] = \begin{bmatrix} x & y \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} -y & x \end{bmatrix}$$

$$[X''] = [X'] * [T2] = \begin{bmatrix} -y & x \end{bmatrix} * \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} y & -x \end{bmatrix}$$

Giả sử [T3] là ma trận tổng hợp [T1] và [T2]

$$[X^*] = [X] * [T3] = \begin{bmatrix} x & y \end{bmatrix} * \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} y & -x \end{bmatrix}$$

Kết luận: biến đổi qua nhiều ma trận thành phần sẽ tương đương với phép biến đổi qua ma trận tổng hợp từ các phép biến đổi đó.

2. TỌA ĐỘ ĐỒNG NHẤT VÀ CÁC PHÉP BIẾN ĐỔI

2.1. Tọa độ đồng nhất

Ta xét phép tịnh tiến:

$$x' = x + dx$$

$$y' = y + dy$$

Vậy $P' = P + [T]$

$$[T] = \begin{bmatrix} dx \\ dy \end{bmatrix}$$

Vậy phép biến đổi tổng hợp:

$$P'' = P' * [T'] + [T''] = (P + [T]) * [T'] + [T'']$$

Rõ ràng không thể biểu diễn thông qua ma trận tổng hợp 2 chiều 2x2 được. Phép tịnh tiến đưa ra ma trận biến đổi tổng hợp là không thể.

Thế nào là phương pháp biểu diễn tọa độ đồng nhất? là phương pháp biểu diễn mở rộng thông qua tọa độ đồng nhất của các vector vị trí không đồng nhất $[x \ y]$ là ứng dụng của phép biến đổi hình học mà ở đó tọa độ điểm được mô tả dưới ma trận $[x^* \ y^* \ h]$ với $x = x^*/h$, $y = y^*/h$ có h là một số thực tùy ý. Vậy một vector vị trí $[xy]$ bất kỳ trên mặt phẳng xoy bằng tập vô số các điểm đồng nhất $[hx \ hyh]$. Ví dụ: $[2 \ 5]$ sẽ biểu diễn bằng $[4102]$, $[6153]$,..... đơn giản nhất là $[251]$.

Vậy tọa độ đồng nhất của vector vị trí $[X] = [x \ y \ 1]$. Khi đó ma trận biến đổi sẽ là 3x3:

$$[T] = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ dx & dy & 1 \end{bmatrix}$$

$$P' = P * [T] = \begin{bmatrix} x & y & 1 \end{bmatrix} * \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ dx & dy & 1 \end{bmatrix}$$

$$x' = ax + cy + dx$$

$$y' = bx + dy + dy$$

$$[X'] = [x' \ y' \ 1]$$

Trong đó dx, dy là khoảng tịnh tiến theo trục x và y.

2.2. Phép biến đổi với tọa độ đồng nhất

Ma trận biến đổi đồng nhất

$$[T] = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ m & n & 1 \end{bmatrix}$$

① Phép tịnh tiến

Có a=d=1 và b=c=0

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix} = [x+m \ y+n \ 1]$$

Chú ý: trong mặt phẳng tọa độ, mọi điểm kể cả gốc tọa độ đều có thể biến đổi.

Phép biến đổi tổng hợp của hai phép tịnh tiến theo khoảng $[m1 \ n1]$ và $[m2 \ n2]$ bằng phép biến đổi duy nhất một khoảng có giá trị bằng tổng của hai phép biến đổi $[m1+m2 \ n1+n2]$.

Thật vậy:

$$[T1] * [T2] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m1 & n1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m2 & n2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m1+m2 & n1+n2 & 1 \end{bmatrix}$$

② Phép tỷ lệ

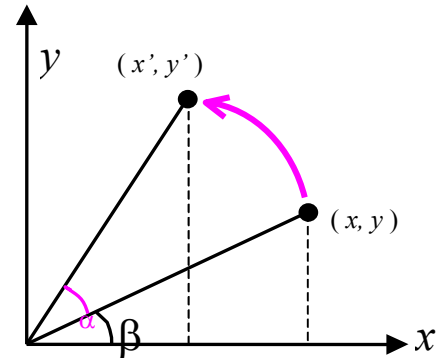
Tương tự ma trận tỷ lệ:

$$[Ts] = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad [x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x.Sx \ y.Sy \ 1]$$

Chú ý: Phép biến đổi tổng hợp của hai phép tỷ lệ $Sx1, Sx2$ và $Sy1, Sy2$ bằng phép biến đổi duy nhất có tỷ lệ là tích hai phép biến đổi trên $Sx1*Sx2, Sy1*Sy2$.

③ Phép quay

$$\begin{aligned} \begin{bmatrix} x' & y' & 1 \end{bmatrix} &= \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} (x \cos \alpha - y \sin \alpha) & (x \sin \alpha + y \cos \alpha) & 1 \end{bmatrix} \end{aligned}$$



Hình 3.6 Phép quay

3. CÁC PHÉP BIẾN ĐỔI HÌNH HỌC BA CHIỀU

Các phép biến đổi chuyển vị - translation, tỉ lệ-scaling và quay-rotation sử dụng trong không gian 2D đều có thể mở rộng trong không gian 3D.

3.1. Biểu diễn điểm trong không gian 3 chiều

$$[x^* \ y^* \ z^* \ h] \text{ hay } [x^*/h \ y^*/h \ z^*/h \ 1]$$

Viết gọn hơn: $[xy \ z \ 1]$

Ma trận biến đổi tổng quát trong không gian 3D với tọa độ đồng nhất (4x4)

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

$$\text{Hay } T = \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & i & h & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

3.2. Phép tịnh tiến

Đây là phép biến đổi đơn giản nhất, mở rộng từ phép biến đổi trong không gian 2D ta có:

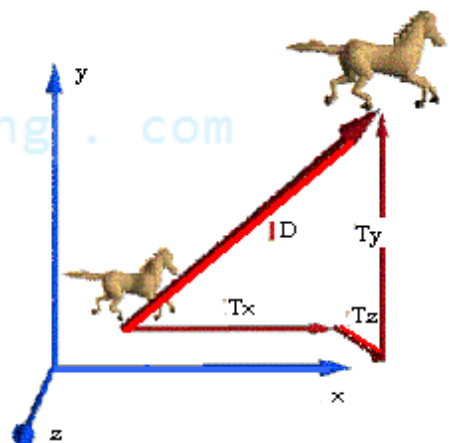
$$[T(dx, dy, dz)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

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

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

]

$$= \begin{bmatrix} x+dx & y+dy & z+dz & 1 \end{bmatrix}$$



Hình 3.7 Phép tịnh tiến trên 3D

3.3. Phép tỉ lệ

Tương tự trong 2D ta có phép tỉ lệ trong 3D là:

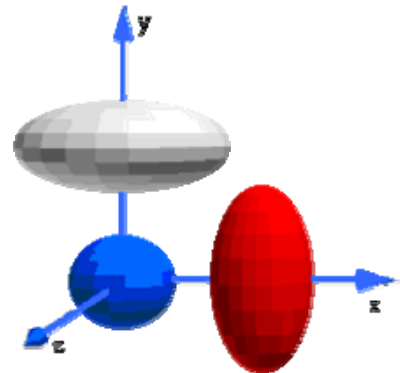
$$Ts = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ta có Sx, Sy và Sz là các hệ số tỉ lệ trên các trục toạ độ

$$[X'] = [X] \cdot [T(Sx, Sy, Sz)]$$

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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



Hình 3.8 Phép tỷ lệ trên 3D

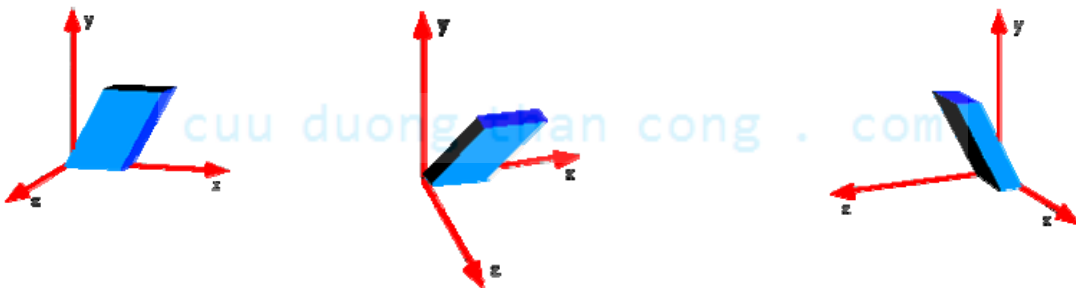
3.4. Phép biến dạng

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

$$[X'] = [X] \cdot [Tsh]$$

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ g & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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



Hình 3.9 Các phép biến dạng trên 3D

3.5. Phép lấy đối xứng

Đối xứng qua trục ox

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

Đối xứng qua mặt xoy

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

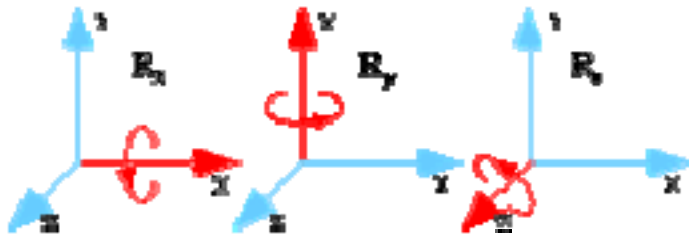
Đối xứng qua gốc O

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

3.6. Phép quay 3 chiều

2.6.1. Quay quanh các trục toạ độ

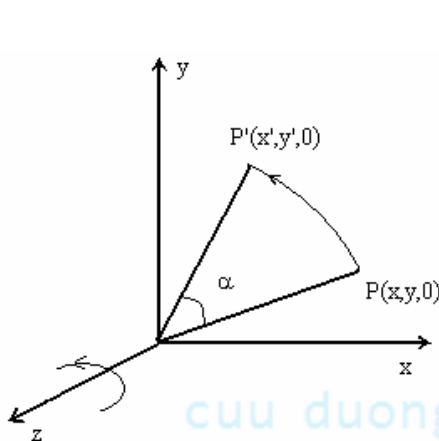
Đơn giản nhất là phép quay quanh các trục toạ độ ox, oy và oz với góc dương:



Hình 3.10 Xác định góc quay dương trên 3 trục toạ độ

Khi này phép quay lại đưa về phép quay không gian 2D quanh gốc toạ độ

- Quay quanh trục oz:

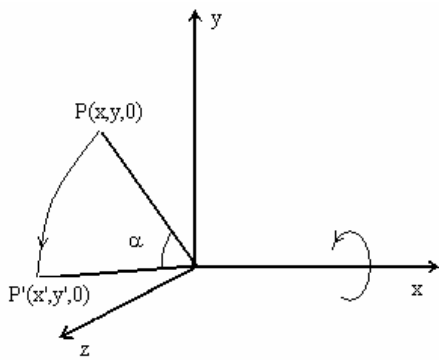


$$\begin{cases} x' = x \cos \alpha - y \sin \alpha \\ y' = x \sin \alpha + y \cos \alpha \\ z' = z \end{cases}$$

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

Hình 3.11 Quay quanh trục oz

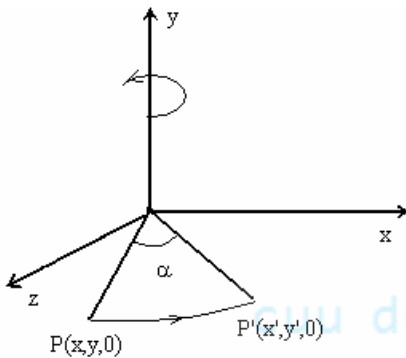
- Quay quanh trục ox:



Hình 3.12 quay quanh trục ox

$$\begin{cases} x' = x \\ y' = y \cos \alpha - z \sin \alpha \\ z' = y \sin \alpha + z \cos \alpha \end{cases} \quad [Tx] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \phi & \sin \phi & 0 \\ 0 & -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Quay quanh trục oy:



Hình 3.13 quay quanh trục oy

$$\begin{cases} x' = x \cos \alpha + z \sin \alpha \\ y' = y \\ z' = -x \sin \alpha + z \cos \alpha \end{cases} \quad [Ty] = \begin{bmatrix} \cos \theta & 0 & -\sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ghi chú: phép quay trong không gian 3D sử dụng phương pháp nhân ma trận biến đổi không có khả năng hoán vị các ma trận.

3.6.2. Quay quanh một trục bất kỳ song song với các trục tọa độ

- Đầu tiên chuyển dịch đối tượng cho đến khi tọa độ địa phương của đối tượng trùng với trục tọa độ mà trục địa phương song song.

- Quay đối tượng xung quanh trục của nó (chính là trục tọa độ)

- Đưa đối tượng về tọa độ trước khi dịch chuyển ta có ma trận tổng hợp là:

$$[T_{\alpha//}] = [T_{tt-}] \cdot [T_{\alpha}] \cdot [T_{tt+}]$$

Ví dụ: quay đối tượng xung quanh một trục // với trục z với khoảng dịch chuyển là x,y và góc quay là α .

Giải:

$$\begin{aligned}
 [T_{\alpha//z}] &= [T_{tt-}][T_{\alpha}][T_{tt+}] \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x & -y & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \alpha & \sin \alpha & 0 & 0 \\ -\sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \alpha & \sin \alpha & 0 & 0 \\ -\sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ (-x \cos \alpha + y \sin \alpha) & (-x \sin \alpha - y \cos \alpha) & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \alpha & \sin \alpha & 0 & 0 \\ -\sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x(1 - \cos \alpha) + y \sin \alpha & y(1 - \cos \alpha) - x \sin \alpha & 0 & 1 \end{bmatrix}
 \end{aligned}$$

3.6.3. Quay đối tượng quanh một trục bất kỳ

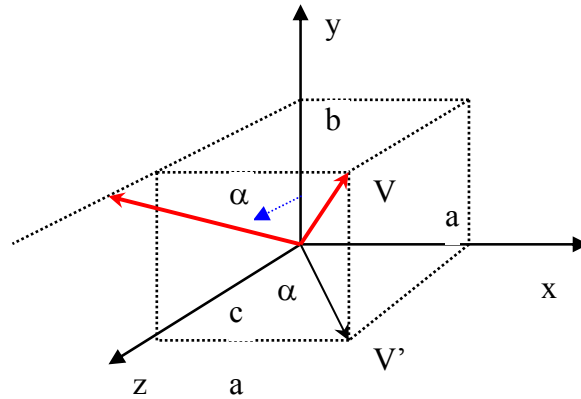
① Xét bài toán sau, hãy tìm ma trận biến đổi để đưa một trục bất kỳ có hướng: $V = ax + by + cz$ về trùng với trục oz theo chiều dương.

Hình 3.14 Quay đối tượng quanh một trục bất kỳ đi qua tâm

Ta thực hiện các bước như sau:

- Quay V quanh trục y một góc $(-\alpha)$ sao cho nằm trên mặt phẳng (y, z) được V_1
- Quay V_1 quanh trục x một góc β sao cho trùng với trục z được V_2

Bước 1:



Hình 3.15 Quay $V=ax+by+cz$ quanh trục oy

Ta tính α ?

Chiếu V trên mặt phẳng (x,z) được V' . Ta có:

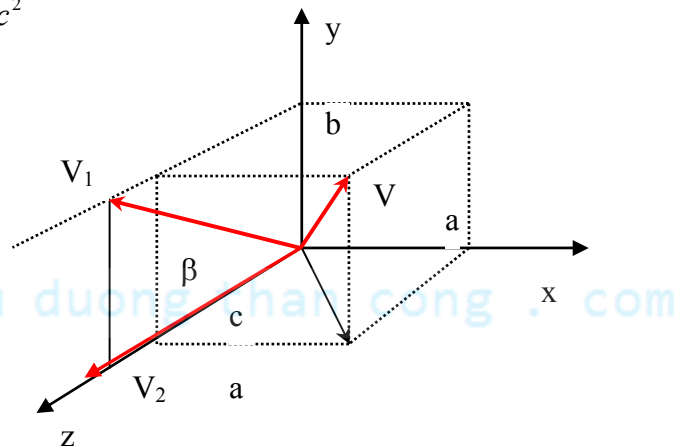
$$\sin \alpha = \frac{a}{\sqrt{a^2 + c^2}}$$

$$\cos \alpha = \frac{c}{\sqrt{a^2 + c^2}}$$

$$[T_{-\alpha y}] = \begin{bmatrix} \cos \alpha & 0 & -\sin(-\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(-\alpha) & 0 & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{c}{\sqrt{a^2 + c^2}} & 0 & \frac{a}{\sqrt{a^2 + c^2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{a}{\sqrt{a^2 + c^2}} & 0 & \frac{c}{\sqrt{a^2 + c^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vector: $V_1(0, b, \sqrt{a^2 + c^2})$

Bước 2:



Hình 3.16 Quay vector V_1 quanh trục ox

$$\sin \beta = \frac{b}{\sqrt{a^2 + b^2 + c^2}}$$

$$\cos \beta = \frac{\sqrt{a^2 + c^2}}{\sqrt{a^2 + b^2 + c^2}}$$

$$[T_{\beta x}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \beta & \sin \beta & 0 \\ 0 & -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{0}{\sqrt{a^2 + c^2}} & \frac{b}{\sqrt{a^2 + b^2 + c^2}} & 0 \\ 0 & \frac{\sqrt{a^2 + b^2 + c^2}}{\sqrt{a^2 + b^2 + c^2}} & \frac{\sqrt{a^2 + c^2}}{\sqrt{a^2 + b^2 + c^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vậy $[T_v] = [T_{-\alpha y}] \cdot [T_{\beta x}]$

Nếu đưa ngược lại ta được:

$$[T_v^{-1}] = ([T_{-\alpha y}][T_{\beta x}])^{-1} = [T_{-\beta x}][T_{\alpha y}]$$

② Cho trục quay L (vector $V = ax + by + cz$) và một điểm P nằm trên trục L. Hãy tìm ma trận biến đổi quay đối tượng xung quanh trục L một góc θ .

Giải:

Ta thực hiện như sau:

1. Tịnh tiến P về gốc tọa độ.
2. Chuyển trục L về trùng với trục OZ
3. Quay đối tượng xung quanh trục OZ một góc θ .
4. Thực hiện ngược lại 2,1

Vậy:

$$[T_{\theta, L}] = [T_{tt-}][T_v][T_{\theta, Z}][T_v^{-1}][T_{tt+}]$$

Tóm tắt:

Các phép biến đổi hình học cho phép dễ dàng thao tác trên các đối tượng đã được tạo ra. Chúng làm thay đổi mô tả về tọa độ của các đối tượng, từ đó đối tượng sẽ được thay đổi về hướng, kích thước và hình dạng. Các phép biến đổi hình học cơ sở bao gồm tịnh tiến, quay và biến đổi tỷ lệ. Ngoài ra một số phép biến đổi khác cũng thường được áp dụng đó là phép đối xứng và biến dạng.

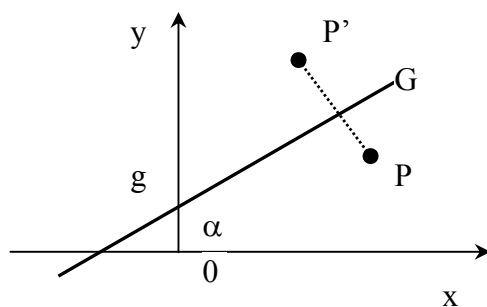
Các phép biến đổi hình học 2D đều được biểu diễn dưới dạng ma trận đồng nhất 3x3 để tiện cho việc thực hiện các thao tác kết hợp giữa chúng. Trong hệ tọa độ đồng nhất, tọa độ của một điểm được mô tả bởi một vector dòng bao gồm ba giá trị, hai giá trị đầu tương ứng với tọa độ Descartes của điểm đó, và giá trị thứ ba là 1. Với cách biểu diễn này, ma trận của phép biến đổi có được từ sự kết hợp của các phép biến đổi cơ sở sẽ bằng tích của các ma trận của các phép biến đổi thành phần.

Phép biến đổi hình học 3D là sự mở rộng của phép biến đổi 2D. Tương tự nó cũng có các phép: tịnh tiến, tỷ lệ, biến dạng và quay. Phức tạp nhất là phép quay, nên chúng ta khảo sát lần lượt từ đơn giản đến phức tạp: quay đối tượng quanh các trục tọa độ, quanh 1 trục song song với trục tọa độ, quanh 1 trục bất kỳ....Do khảo sát các phép biến đổi affine với biểu diễn dạng ma trận đồng nhất (4x4 với 3D) nên công việc khá đơn giản và nhất quán.

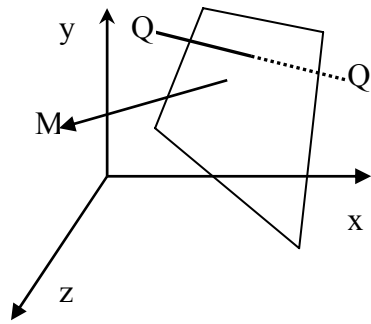
Lưu ý phép tịnh tiến và quay có chung thuộc tính là: sau khi biến đổi hình dạng và kích thước của đối tượng không thay đổi mà chúng chỉ bị thay đổi vị trí và định hướng trong không gian.

Bài tập:

1.
 - a. Hãy tìm ma trận biểu thị phép quay một đối tượng một góc 60° quanh gốc tọa độ.
 - b. Tìm tọa độ mới của $P(-3,3)$ sau khi thực hiện phép quay trên?
2.
 - a. Hãy viết dạng tổng quát của ma trận điều chỉnh tỷ lệ tương ứng với một điểm cố định $Q(a,b)$?
 - b. Phóng lớn tứ giác có các đỉnh $A(0,0)$, $B(1,3)$, $C(4,2)$ và $D(3,1)$ lên hai lần kích thước ban đầu của nó trong khi vẫn giữ điểm $D(3,1)$?
3.
 - a. Mô tả phép biến đổi nhằm quay 1 đối tượng một góc α xung quanh một tâm cố định $Q(a,b)$?
 - b. Thực hiện phép quay tam giác ABC có $A(0,0)$, $B(1,1)$ và $C(4,2)$ một góc 45° xung quanh điểm $(-1, -1)$?
4. Cho $\triangle ABC$ có các tọa độ đỉnh là $A(2,2)$, $B(3,1)$ và $C(4,3)$. Xác định ma trận biến đổi affine biến đổi tam giác này thành $A'B'C'$ biết ảnh $A'(4,3)$, $B'(4,5)$ và $C'(7,3)$.
5.
 - a. hãy tìm một ma trận dành cho phép phản chiếu đối xứng gương xung quanh một đường thẳng G với độ dốc m và tung độ gốc $(0,g)$



- b. Tạo phản xạ đối xứng gương đa giác mà đỉnh của nó: $A(-1,0)$, $B(0,-2)$, $C(1,0)$ và $D(0,2)$ xung quanh đường G trong các trường hợp sau:
 - i. $x=2$
 - ii. $y=3$
 - iii. $y=2x+3$
6. Cho hình chóp ABCD có các tọa độ $A(0,0,0)$, $B(1,0,0)$, $C(0,1,0)$ và $D(0,0,1)$. Quay hình chóp quanh đường L ($v=x+y+z$) đi qua điểm C một góc 45° . Tìm tọa độ hình chóp mới.
7.
 - a. Hãy tìm phép biến đổi dành cho phép đối xứng gương tương ứng với một mặt phẳng đã cho (có vector pháp tuyến M, trên đó có một điểm P).



- b. Áp dụng tìm ma trận cho phép đối xứng gương với mặt phẳng đi qua gốc toạ độ và vector pháp tuyến có hướng $M=x+y+z$.
8. Viết chương trình với đối tượng (đường thẳng, tam giác, tứ giác...) trong mặt phẳng 2D
 - a. Dịch chuyển (dùng các phím dịch chuyển)
 - b. Phóng lớn, thu nhỏ (dùng phím dịch chuyển hay gõ vào từ bàn phím)
 - c. Quay với góc quay được gõ vào từ bàn phím (góc gõ vào được tính bằng độ)
9. Viết chương trình với đối tượng (hình kim cương, hình lập phương ...) trong 3D
 - a. Dịch chuyển
 - b. Phóng lớn, thu nhỏ
 - c. Quay một góc

Bài tập trắc nghiệm: CuuDangThanCong.com

1. Cho ΔABC có các toạ độ $A(1,1)$, $B(1,2)$ và $C(3,4)$ thu nhỏ tam giác chỉ còn $\frac{1}{4}$ mà vẫn giữ cố định điểm A. Toạ độ mới của tam giác là:
 - a. $A'(1,1)$, $B'(1/4, 1/2)$ và $C'(3/4,1)$
 - b. $A'(1,1)$, $B'(0.5, 1)$ và $C'(1.25,1.5)$
 - c. $A'(0.25,0.25)$, $B'(1, 1.5)$ và $C'(0.75,2)$
 - d. $A'(1,1)$, $B'(1, 1.25)$ và $C'(0.5,1.75)$
2. Cho đoạn thẳng AB có toạ độ $A(2,3)$ và $B(6,1)$ quay AB một góc 60° vẫn giữ cố định B. Toạ độ mới của AB là:
 - a. $A'((1-\sqrt{3}), (-\sqrt{3}+2))$ và $B'(6,1)$
 - b. $A'((4-\sqrt{3}), (-2\sqrt{3}+2))$ và $B'(6,1)$
 - c. $A'((\frac{1}{4}-2\sqrt{3}), (-\sqrt{3}+1))$ và $B'(1,6)$
 - d. $A'((-1+\sqrt{3}), (4\sqrt{3}+1))$ và $B'(6,1)$
3. Cho hình chữ nhật ABCD có các toạ độ là $A(-1,-1)$, chiều rộng HCN là 4 và chiều dài là 3. Người ta phóng lớn HCN để nó cao lên gấp 2 lần và rộng gấp 1.5 lần mà vẫn giữ cố định A. Toạ độ mới của HCN đó:
 - a. $A'(-1,-1)$, $B'(5,-1)$, $C'(5,-7)$ và $D'(-1,-7)$
 - b. $A'(-1,-1)$, $B'(-1,5)$, $C'(5,-1)$ và $D'(-7,-1)$
 - c. $A'(2,2)$, $B'(3,4)$, $C'(5,-7)$ và $D'(-7,5)$
 - d. $A'(-1,-1)$, $B'(0,-1)$, $C'(-1,-7)$ và $D'(-1,5)$

4. Cho đường tròn có tâm tại (1,4), một điểm trên đường tròn A(1,0). Quay đường tròn một góc 90^0 quanh điểm A, tâm mới của đường tròn là:
 - a. (0.5, 2)
 - b. (-3,1)
 - c. (2,5)
 - d. (-2,3)
5. Cho đoạn thẳng AB trong không gian có tọa độ A(2,1,1) và B(1,3,4), quay đoạn thẳng quanh trục oz một góc 30^0 vẫn cố định A. Tọa độ mới của AB là:
 - a. $A'(2,1,1), B'((1-\sqrt{3}), (\sqrt{3}+2), 3)$
 - b. $A'((2-\sqrt{3}), (\sqrt{3}+\frac{1}{2}), 4), B'(1,3,4)$
 - c. $A'(1,3,5), B'((1-\sqrt{3}), (\sqrt{3}+2), 3)$
 - d. $A'(2,1,1), B'((2-1\frac{\sqrt{3}}{2}), \sqrt{3}, 4)$
6. Cho ΔABC trong không gian có tọa độ A(1,1,1), B(4,6,0) và C(2,-1,3) kéo dãn cho tam giác rộng ra (theo hướng trục ox) lên 2 lần vẫn giữ cố định B. Tọa độ mới của ΔABC là:
 - a. $A'(2,1,1), B'(4,6,0)$ và $C'(2,-1,3)$
 - b. $A'(-2,1,1), B'(4,6,0)$ và $C'(0,-1,3)$
 - c. $A'(2,1,1), B'(2,3,1)$ và $C'(2,-1,1)$
 - d. $A'(2,2,2), B'(4,6,0)$ và $C'(1,-0.5,1.5)$
7. Cho hình kim cương ABCD có các tọa độ là A(4,6,1), B(1,2,3), C(2,2,5) và D(7,2,4). Đối xứng gương hình kim cương qua trục ox, tọa độ mới của hình kim cương là:
 - a. $A'(4,6,-1), B'(1,2,-3), C'(2,2,-5)$ và $D'(7,2,-4)$
 - b. $A'(-4,6,-1), B'(-1,2,-3), C'(-2,2,-5)$ và $D'(-7,2,-4)$
 - c. $A'(4,-6,-1), B'(1,-2,-3), C'(2,-2,-5)$ và $D'(7,-2,-4)$
 - d. $A'(-4,-6,-1), B'(-1,-2,-3), C'(-2,-2,-5)$ và $D'(-7,-2,-4)$
8. Cho hình kim cương ABCD có các tọa độ là A(5,6,1), B(0,0,0), C(3,2,5) và D(8,2,4). Quay hình kim cương quanh trục oy và vẫn giữ cố định B. Tọa độ mới của hình kim cương là:
 - a. $A'(-1,5,-1), B'(0,0,0), C'(2,3,-3)$ và $D'(2,4,8)$
 - b. $A'(1,6,-5), B'(1,2,2), C'(5,2,-3)$ và $D'(2,-4,8)$
 - c. $A'(-1,5,-1), B'(0,0,0), C'(2,3,5)$ và $D'(2,4,8)$
 - d. $A'(1,6,-5), B'(0,0,0), C'(2,5,-3)$ và $D'(2,4,-8)$
9. Trong đoạn mã sau các số từ ①, ②, ③, ④ lần lượt là các phép mũi tên dịch chuyển:


```

c = getch();
switch (c)
{
    case 75: x -= 10; // ①
                break;
    case 77: x += 10; // ②
                break;
    case 72: y -= 10; // ③
                break;
    case 78: y += 10; // ④
                break;
}
            
```



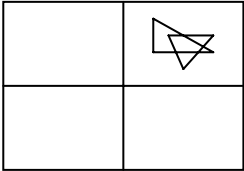
```
        break;
    case 80: y += 10; // ④
        break;
}
```

- a. Trái, phải, trên và dưới
- b. Trái, phải, dưới và trên
- c. Trên, dưới, trái và phải
- d. Trên, dưới, phải và trái

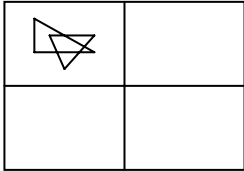
10. Chương trình sau đưa ra cái gì?

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#define RADS 0.017453293
void Function1() {
    line (10,getmaxy()/2,getmaxx()-10,getmaxy()/2);
    line (getmaxx()/2,10,getmaxx()/2,getmaxy()-10);
}
void Function2(int x1,int y1,int x2,int y2,int x3,int y3){
    line (getmaxx()/2+x1,getmaxy()/2-y1,getmaxx()/2+x2,getmaxy()/2-y2);
    line (getmaxx()/2+x2,getmaxy()/2-y2,getmaxx()/2+x3,getmaxy()/2-y3);
    line (getmaxx()/2+x3,getmaxy()/2-y3,getmaxx()/2+x1,getmaxy()/2-y1);
}
void Function3(int x1,int y1, int x2, int y2, int x3, int y3,int xq, int yq,float goc ){
    float x11,y11,x22,y22,x33,y33;
    float anpha = RADS *goc;
    x11 = int(x1*cos(anpha) - y1*sin(anpha) + (1 - cos(anpha))*xq + sin(anpha)*yq);
    y11 = int(x1*sin(anpha) + y1*cos(anpha) - sin(anpha)*xq + (1 - cos(anpha))*yq);
    x22 = int( x2*cos(anpha) - y2*sin(anpha) + (1 - cos(anpha))*xq + sin(anpha)*yq);
    y22 =int( x2*sin(anpha) + y2*cos(anpha) - sin(anpha)*xq + (1 - cos(anpha))*yq);
    x33 =int( x3*cos(anpha) - y3*sin(anpha) + (1 - cos(anpha))*xq + sin(anpha)*yq);
    y33=int( x3*sin(anpha) + y3*cos(anpha) - sin(anpha)*xq + (1 - cos(anpha))*yq);
    Function2(x11,y11,x22,y22,x33,y33);
}
void main() {
    clrscr();
    int driver = DETECT, mode;
    initgraph(&driver,&mode,"c:\\tc\\bgi");
    int x1 =50,y1 = 20,x2 = 50,y2 = 100,x3 = 200,y3 = 20;
    int xq=100,yq=100;
    float goc=30 ;
    Function1();
    Function2(x1,y1,x2,y2,x3,y3);
```

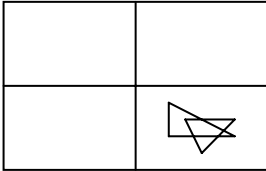
```
Function3(x1,y1,x2,y2,x3,y3,xq,yq,goc);  
getch();  
closegraph();  
}
```



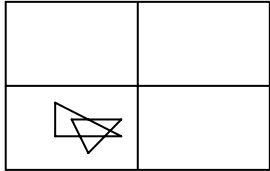
A



b



c



D

cuu duong than cong . com

cuu duong than cong . com

CHƯƠNG 4: CÁC GIẢI THUẬT ĐỒ HOẠ CƠ SỞ

1. HỆ TỌA ĐỘ VÀ MÔ HÌNH CHUYỂN ĐỔI

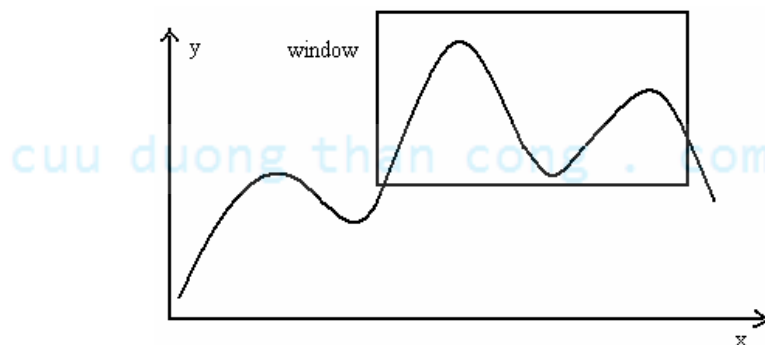
1.1. Các hệ thống tọa độ trong đồ họa

Một hình ảnh được tạo ra bằng máy tính thường trình bày một khung xem (viewing) từng phần của quang cảnh lớn. Giống hệt như ta nhìn cảnh vật qua cửa sổ hay một kính ngắm camera. Để tạo ra hình ảnh đối tượng trên các thiết bị hiển thị bắt buộc người sử dụng phải có bước biến đổi trung gian.

1.1.1. Hệ tọa độ thực (WCS – World Coordinate System)

Hệ tọa độ của đối tượng được các chương trình ứng dụng sử dụng để mô tả tọa độ các đối tượng trong thế giới thực.

Đơn vị trong hệ thống tọa độ phụ thuộc vào không gian và kích thước của đối tượng được mô tả: A^0 , nm, mm, m, km...



Hình 4.1 Hệ tọa độ thực

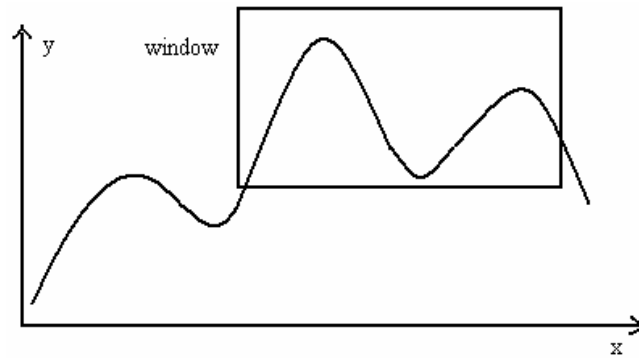
1.1.2. Hệ tọa độ thiết bị (DCS – Device Coordinate System)

Là hệ thống tọa độ của thiết bị nơi hiển thị hình ảnh và không gian của đối tượng mà ứng dụng mô tả. Không gian của hệ thống của tọa độ này chính là kích thước của thiết bị hiển thị được sử dụng.

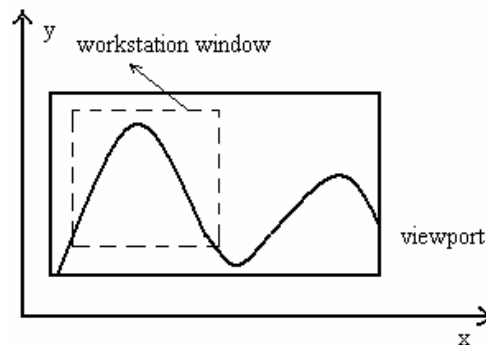
Ví dụ: màn hình VGA 640x480, SVGA 600x800...

1.1.3. Hệ tọa độ thiết bị chuẩn (NDCS – Normalized Device Coordinate System)

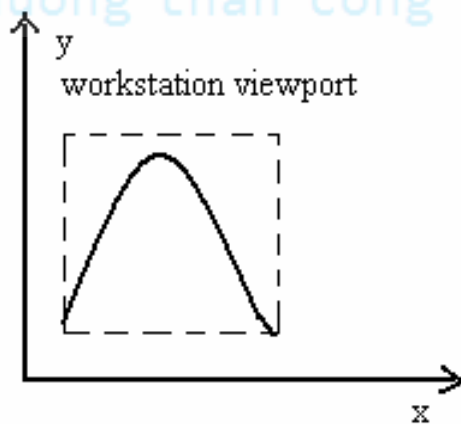
Chuyển đổi đối tượng từ không gian thực vào tọa độ thiết bị hiển thị. Phần mềm đồ họa được viết sẽ thực hiện sự chuyển đổi mà chưa thể xác định rõ kích thước của thiết bị cũng như độ phân giải. Ta có công cụ độc lập với thiết bị nhằm mô tả vùng hiển thị ra hệ tọa độ thiết bị chuẩn hoá. Coi NDCS như hệ tọa độ thiết bị có kích thước màn hình hiển thị là một hình vuông có cạnh là một đơn vị (1x1).



Hình 4.2 WCS – World Coordinate System



Hình 4.3 NDCS – Normalized Device Coordinate System

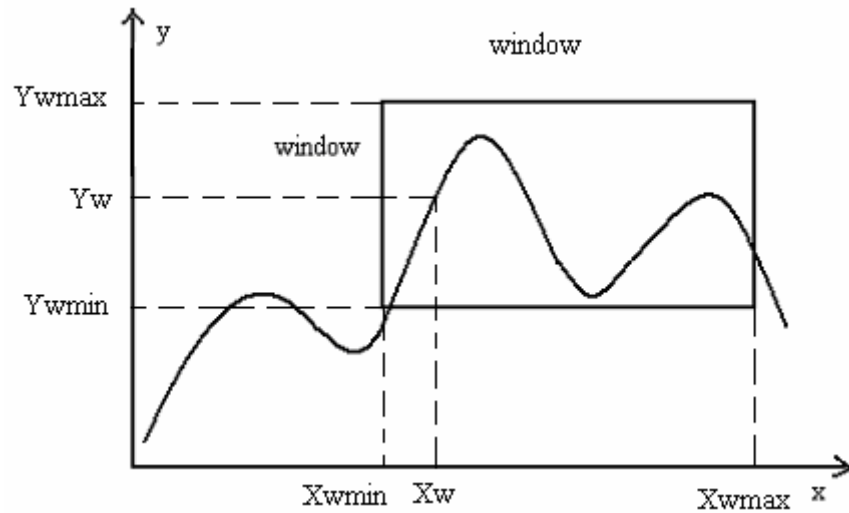


Hình 4.4 DCS – Device Coordinate System

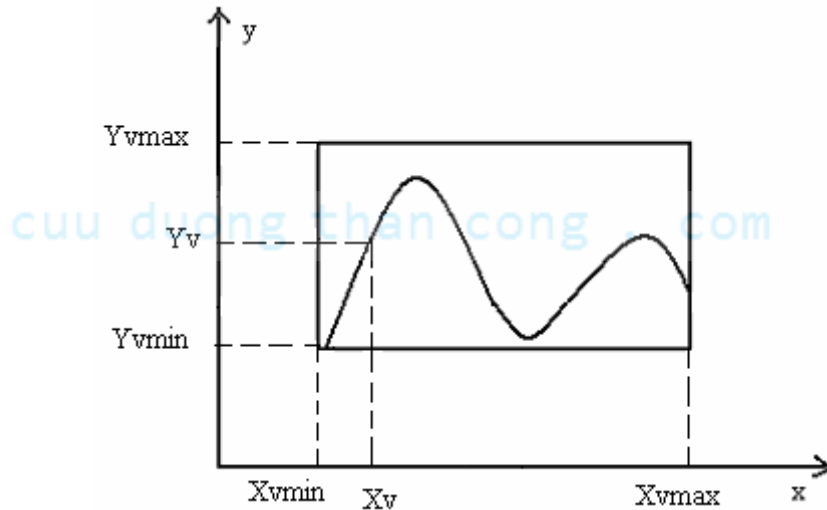
- Qui trình để chuyển đổi các đối tượng trong WCS sang NDCS được gọi là phép ánh xạ cửa sổ sang cổng xem hay phép biến đổi chuẩn hoá (Window to Viewport mapping or Normalization Transformation)

- Qui trình có thể áp các tọa độ thiết bị hiển thị chuẩn hoá sang các thiết bị rời rạc được gọi là phép biến đổi trạm làm việc (Workstation Transformation)

1.2. Phép ánh xạ từ cửa sổ vào cổng xem



Hình 4.5 Đối tượng trong cửa sổ



Hình 4.6 Đối tượng tại cổng xem

- Một cửa sổ (window) được chỉ định bởi bốn tọa độ thực (WCS): X_{wmin} , X_{wmax} , Y_{wmin} , Y_{wmax}

- Một cổng xem (viewport) được mô tả bởi bốn tọa độ thiết bị chuẩn hoá (NDCS): X_{vmin} , X_{vmax} , Y_{vmin} , Y_{vmax}

Mục đích của phép ánh xạ này là chuyển đổi các tọa độ thực (X_w, Y_w) của một điểm tùy ý sang thiết bị chuẩn hoá tương ứng (X_v, Y_v). Để giữ lại khoảng cách của điểm trong cổng xem bằng với khoảng cách của điểm trong cửa sổ, với yêu cầu:

$$\frac{x_v - x_{vmax}}{x_{vmax} - x_{vmin}} = \frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}$$

$$\frac{y_v - y_{vmax}}{y_{vmax} - y_{vmin}} = \frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}}$$

Ta có:

$$x_v = \frac{(x_{v\max} - x_{v\min})}{(x_{w\max} - x_{w\min})}(x_w - x_{w\min}) + x_{v\max}$$

$$y_v = \frac{(y_{v\max} - y_{v\min})}{(y_{w\max} - y_{w\min})}(y_w - y_{w\min}) + y_{v\max}$$

Ta có tám giá trị ở trên để xác định cửa sổ và cổng xem đều là hằng số. Vậy chúng ta hoàn toàn tính được (X_v, Y_v) từ (X_w, Y_w) qua phép biến đổi:

$$[X_v Y_v 1] = [X_w Y_w 1] \cdot [T]$$

$$[T] = [T_{tt-}] \cdot [T_s] \cdot [T_{tt}]$$

- Ma trận tịnh tiến theo window:

$$[T_{tt-}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_w & -y_w & 1 \end{bmatrix}$$

- Ma trận chuyển đổi tỷ lệ window vào viewport là:

$$[T_s] = \begin{bmatrix} \frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}} & 0 & 0 \\ 0 & \frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Ma trận tịnh tiến theo tọa độ viewport:

$$[T_{tt}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_v & y_v & 1 \end{bmatrix}$$

Vậy ma trận biến đổi từ cửa sổ vào cổng xem:

$$[T] = \begin{bmatrix} \frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}} & 0 & 0 \\ 0 & \frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}} & 0 \\ x_{v\min} - x_{w\min} \cdot \frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}} & y_{v\min} - y_{w\min} \cdot \frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}} & 1 \end{bmatrix}$$

2. CÁC GIẢI THUẬT XÉN TỈA (CLIPPING)

2.1. Khái niệm

Xén tỉa là tiến trình xác định các điểm của một đối tượng nằm trong hay ngoài cửa sổ hiển thị. Nằm trong được hiển thị, nằm ngoài loại bỏ.

Việc loại từng điểm ảnh của đối tượng thường chậm nhất là khi đối tượng mà phần lớn nằm ngoài cửa sổ hiển thị.

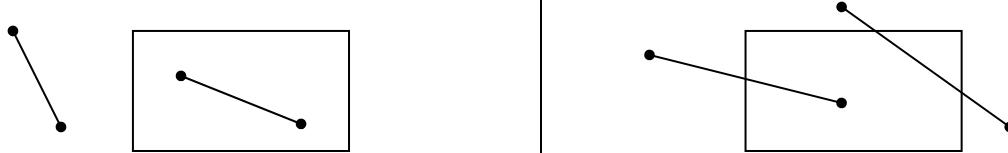
2.2. Clipping điểm

Giả sử (x,y) là toạ độ của một điểm, vậy điểm đó được hiển thị khi thoả mãn:

$$X_{min} \leq x \leq X_{max}$$

$$Y_{min} \leq y \leq Y_{max}$$

2.3. Xén tỉa đoạn thẳng



Các đoạn thẳng không cắt cửa sổ thì: hoặc nằm trong hoàn toàn, hoặc nằm ngoài hoàn toàn

Nếu đoạn thẳng cắt cửa sổ thì phân chia qua điểm cắt phần nằm trong và phần nằm ngoài

Hình 4.7 Các dạng xén tỉa đoạn thẳng

2.3.1. Thuật toán Cohen-Sutherland

Ý tưởng: Các đoạn thẳng có thể rơi vào các trường hợp sau

- Hiển thị (visible): cả hai đầu cuối của đoạn thẳng đều nằm bên trong cửa sổ

- Không hiển thị (not visible): đoạn thẳng xác định nằm ngoài cửa sổ. Điều này xảy ra khi đoạn thẳng từ (x_1,y_1) đến (x_2,y_2) thoả mãn bất kỳ một trong bốn bất đẳng thức sau:

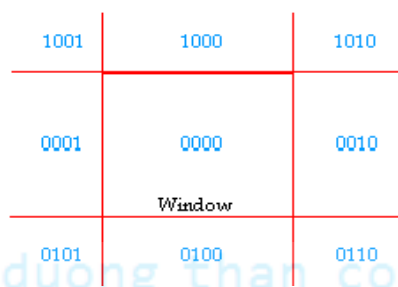
$$x_1, x_2 > x_{max} \vee y_1, y_2 > y_{max}$$

$$x_1, x_2 < x_{min} \vee y_1, y_2 < y_{min}$$

- Xén tỉa: đoạn thẳng cần xén tỉa

Việc cài đặt giải thuật chia làm hai bước:

① Gán mã vùng 4-bit cho mỗi điểm cuối của đoạn thẳng



Hình 4.8 Mặt phẳng mã trong các trường hợp clipping đoạn thẳng

Mã vùng được xác định theo 9 vùng của mặt phẳng mà các điểm cuối nằm vào đó. Một bit được cài đặt true (1) hoặc false (0).

Bit 1: điểm cuối ở bên trên cửa sổ = $\text{sign}(y - y_{max})$

Bit 2: điểm cuối ở bên dưới cửa sổ = $\text{sign}(y_{min} - y)$

Bit 3: điểm cuối ở bên phải cửa sổ = $\text{sign}(x - x_{max})$

Bit 4: điểm cuối ở bên trái cửa sổ = $\text{sign}(x_{min} - x)$

Qui ước $\text{sign}(a) = 1$ nếu a dương
 $= 0$ nếu a âm

② Quá trình kiểm tra vị trí của đoạn thẳng so với cửa sổ. Tất cả điểm đầu và điểm cuối của đoạn thẳng đã có mã.

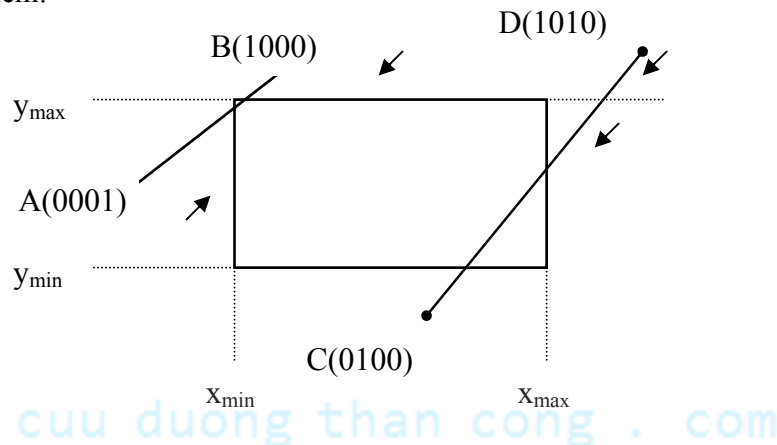
❶ Nếu mã của P_1 hoặc P_2 đều = 0000 thì toàn bộ đoạn thẳng thuộc phần hiển thị.

If ($P_1.\text{Mã OR } P_2.\text{Mã} == 0000$) then “ cả đoạn thẳng thuộc cửa sổ hiển thị”

❷ Nếu mã của P_1 và P_2 có cùng một vị trí mà ở đó P_1 và $P_2 \Rightarrow$ cùng phía

If ($P_1.\text{Mã AND } P_2.\text{Mã} \neq 0000$) then “ 2 điểm nằm về 1 phía của cửa sổ”

❸ Xét giao điểm:



Hình 4.9 Mô tả thuật toán Cohen Sutherland outcode

- Tìm giao điểm của đường thẳng với cửa sổ, chính xác hơn là với phần mở rộng của đường biên.

- Chú ý: các đường biên mà điểm cuối được chọn sẽ “đẩy ngang qua” nhằm thay đổi mã “1” thành “0”

- Nếu:

Bít 1 là 1: cắt $y=y_{\max}$

Bít 2 là 1: cắt $y=y_{\min}$

Bít 3 là 1: cắt $x=x_{\max}$

Bít 4 là 1: cắt $x=x_{\min}$

Nhìn trên hình ta có: gọi điểm cuối của đoạn (x_1, y_1)

Nếu C được chọn thì đường $y=y_{\min}$ chọn để tính phần cắt nhau (bít 2 = 1)

Nếu D được chọn thì $y=y_{\max}$ hoặc $x=x_{\max}$ (bít 1 và bít 3 =1)

Toạ độ cắt:

$$\begin{cases} x_i = x_{\min} / x_{\max} \\ y_i = y_1 + m(x_i - x_1) \end{cases} \begin{cases} x = x_{\min} \\ x = x_{\max} \end{cases}$$

Hoặc:

$$\begin{cases} x_i = x_1 + (y_i - y_1) / m \\ y_i = y_{\min} / y_{\max} \end{cases} \begin{cases} y = y_{\min} \\ y = y_{\max} \end{cases}$$

Có m là độ dốc của đoạn thẳng

$$m = (y_2 - y_1) / (x_2 - x_1)$$

Bây giờ thay điểm cuối (x_1, y_1) với điểm cắt (x_i, y_i)

Ví dụ: Cho cửa sổ cắt tia hình chữ nhật có góc trái dưới L(-3,1), góc phải trên R(2,6)

Hãy tìm mã vùng dành cho các điểm cuối của các đoạn AB có A(-2,3), B(1,2); CD có C(-4,7), D(-2,10); IK có I(-4,2), K(-1,7)

Tìm hạng mục cắt tia của chúng

Giải:

- Mã vùng

$$\text{Bít 1} = \text{sign}(y - y_{\max}) = \text{sign}(y - 6)$$

$$\text{Bít 2} = \text{sign}(y_{\min} - y) = \text{sign}(1 - y)$$

$$\text{Bít 3} = \text{sign}(x - x_{\max}) = \text{sign}(x - 2)$$

$$\text{Bít 4} = \text{sign}(x_{\min} - x) = \text{sign}(-3 - x)$$

$$\begin{aligned} \text{Qui ước } \text{sign}(a) &= 1 \text{ nếu } a \text{ dương} \\ &= 0 \text{ nếu } a \text{ âm} \end{aligned}$$

Vậy:

A(-2,3) có (0000)

B(1,2) có (0000)

C(-4,7) có (1001)

D(-2,10) có (1000)

I(-4,2) có (0001)

K(-1,7) có (1000)

- Hạng mục cắt tia

AB có mã vùng đều là (0000) nên nó nằm hoàn toàn trong

CD có (1001) and (1000) = (1000) (!= (0000)) nên hoàn toàn nằm ngoài

IK có (0001) or (1000) = (1001) và (0001) and (1000) = (0000) nên phải xén tia.

Xét I(0001) dựa trên đường biên $x_{\min} = -3$

$$\begin{cases} x_c = x_{\min} \\ y_c = y_1 + m(x_c - x_1) \end{cases}$$

$$x_{\min} = -3$$

$$m = (y_2 - y_1) / (x_2 - x_1) = (7 - 2) / (-1 - (-4)) = 5/3$$

$$y_c = 2 + 5/3 (-3 - (-4)) = 11/3$$

Thay I bởi $I_1 (-3, 11/3)$ (có mã vùng 0000)

Xét K(1000) cắt $y_{\max} = 6$

$$\begin{cases} x_c = x_1 + (y_c - y_1) / m \\ y_c = y_{\max} \end{cases}$$

$$y_c = 6$$

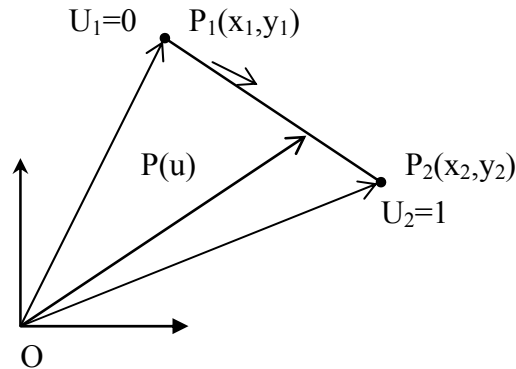
$$x_c = -1 + (6 - 7) / (5/3) = -8/5$$

Thay K bởi $K_1 (-8/5, 6)$ có mã vùng 0000)

Vậy sau khi xén tia đoạn IK thu được I_1K_1

2.3.2. Giải thuật Lyangbarsky

Biểu diễn đoạn thẳng theo tham biến: đoạn thẳng bất kỳ đi qua hai điểm $P_1(x_1, y_1)$ và $P_2(x_2, y_2)$ chúng ta có phương trình tham biến:



Hình 4.10 Phương trình tham số cho đoạn thẳng

$$x = x(u) \quad \text{có } 0 \leq u \leq 1$$

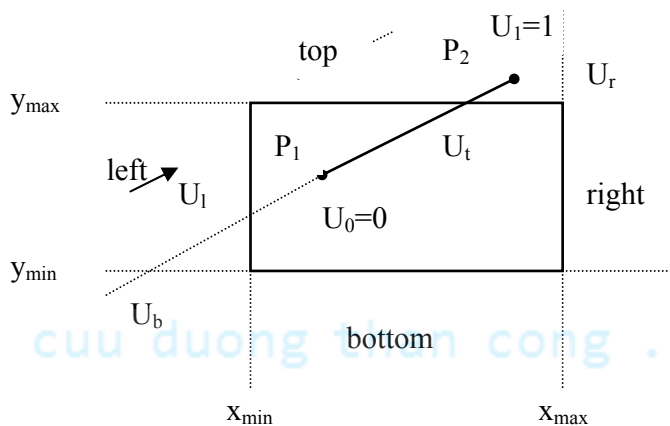
$$y = y(u)$$

Vì vector chỉ có một chiều nên vector vị trí:

$$P(u) = P_1 + (P_2 - P_1) \cdot u$$

$$x = x_1 + (x_2 - x_1)u = x_1 + \Delta x \cdot u$$

$$y = y_1 + (y_2 - y_1)u = y_1 + \Delta y \cdot u$$



Hình 4.11 Mô tả giải thuật LyaBarsky

+ Khi ta chuyển động dọc theo đường mở rộng với u tăng từ $-\infty$ tới $+\infty$ thì ta phải:

- Chuyển từ bên ngoài vào bên trong hai đường biên của cửa sổ cắt tia (ở dưới và bên trái)
- Sau đó chúng ta di chuyển từ trong ra bên ngoài của hai đường biên khác (từ trên và từ bên phải)

+ Với điểm (x,y) nằm bên trong cửa sổ cắt tia:

$$x_{\min} \leq x_1 + \Delta x.u \leq x_{\max}$$

$$y_{\min} \leq y_1 + \Delta y.u \leq y_{\max}$$

Suy ra:

$$-\Delta x.u \leq x_1 - x_{\min} \quad k=1$$

$$\Delta x.u \leq x_{\max} - x_1 \quad k=2$$

$$-\Delta y.u \leq y_1 - y_{\min} \quad k=3$$

$$\Delta y.u \leq y_{\max} - y_1 \quad k=4$$

Viết tổng quát:

$$P_k.u \leq q_k \text{ có } k=1,2,3,4$$

Trong đó:

$$P_1 = -\Delta x q_1 = x_1 - x_{\min} \quad (\text{phải})$$

$$P_2 = \Delta x q_2 = x_{\max} - x_1 \quad (\text{trái})$$

$$P_3 = -\Delta y q_3 = y_1 - y_{\min} \quad (\text{trên})$$

$$P_4 = \Delta y q_4 = y_{\max} - y_1 \quad (\text{dưới})$$

Xét:

+ Nếu $P_k = 0$: điều đó tương đương với việc đoạn thẳng đang xét song song với cạnh thứ k của hình chữ nhật clipping.

a) Nếu $q_k < 0$ đoạn thẳng nằm ngoài cửa sổ (hệ bất phương trình trên vô nghiệm)

b) Nếu $q_k \geq 0$ thì đoạn thẳng nằm trong hoặc nằm trên cạnh của cửa sổ clipping. Yêu cầu khảo sát tiếp.

+ Nếu $P_k \neq 0$: đoạn thẳng đang xét sẽ cắt cạnh k tương ứng của cửa sổ clipping tại vị trí trên đoạn thẳng $u_k = q_k/P_k$

a) $P_k < 0$ đoạn thẳng có dạng đi từ ngoài vào trong của đường biên tương ứng

b) $P_k > 0$ thì đoạn thẳng mở rộng tiến hành từ trong ra ngoài của đường biên tương ứng.

Kết hợp lại ta có các bước sau:

(Với $u_1 \leq u \leq u_2$)

① Nếu $P_k = 0$

$q_k < 0$ ứng với bất kỳ giá trị nào của k . Loại bỏ đoạn thẳng \Rightarrow Exit

$q_k \geq 0$ thực hiện bước tiếp theo

② Tính:

$$U_1 = \max \left(\{0\} \cup \left\{ u_k : u_k = \frac{q_k}{P_k}, P_k < 0 \right\} \right)$$

$$U_2 = \min \left(\{1\} \cup \left\{ u_k : u_k = \frac{q_k}{P_k}, P_k > 0 \right\} \right)$$

u_1 là các giá trị cực đại của tập hợp đang chứa 0 và các giá trị u_k được tính

u_2 là các giá trị cực tiểu của tập hợp đang chứa 1 và các giá trị u_k được tính

Ví dụ: cho cửa sổ cắt tia hình chữ nhật có góc trái dưới L(1,2) và góc phải trên R(9,8). Có các đoạn AB toạ độ A(11,10) và B(11,6), đoạn CD toạ độ C(3,2) và D(8,4), đoạn IJ toạ độ I(-1,7) và J(11,1). Tìm các hạng mục cắt tia.

Giải:

① Xét AB

$$P_1 = 0, q_1 = 10$$

$$P_2 = 0, q_2 = -2$$

$$P_3 = -4, q_3 = 4$$

$$P_4 = 4, q_4 = 2$$

Có $P_2 = 0$ mà $q_2 = -2 < 0$ nên AB nằm hoàn toàn ngoài

② Xét CB

$$P_1 = -5, q_1 = 2 \quad u_1 = -2/5 \quad (u_k = q_k/p_k)$$

$$P_2 = 5, q_2 = 6 \quad u_2 = 6/5$$

$$P_3 = -2, q_3 = 0 \quad u_3 = 0$$

$$P_4 = 2, q_4 = 6 \quad u_4 = 3$$

Vậy $u_1 = \max(0, -2/5, 0) = 0$ (với $P_k < 0$)

$u_2 = \min(1, 6/5, 3) = 1$ (với $P_k > 0$)

Vậy CD nằm hoàn toàn trong cửa sổ

③ Xét IJ

$$P_1 = -12, q_1 = -2 \quad u_1 = 1/6$$

$$P_2 = 12, q_2 = 10 \quad u_2 = 5/6$$

$$P_3 = 6, q_3 = 5 \quad u_3 = 5/6$$

$$P_4 = -6, q_4 = 1 \quad u_4 = -1/6$$

Vậy $u_1 = \max(0, 1/6, -1/6) = 1/6$ (với $P_k < 0$)

$u_2 = \min(1, 5/6, 5/6) = 5/6$ (với $P_k > 0$)

Vì $u_1 < u_2$ nên hai điểm cuối của đường cắt tia là:

$$X_{c1} = x_1 + \Delta x \cdot u_1 = -1 + (11 - (-1)) \cdot 1/6 = 1$$

$$Y_{c1} = y_1 + \Delta y \cdot u_1 = 7 + (1 - 7) \cdot 1/6 = 6$$

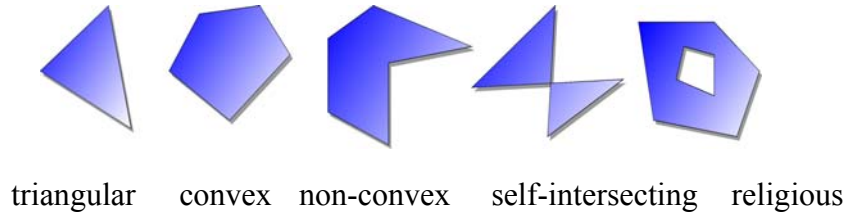
$$X_{c2} = x_1 + \Delta x \cdot u_2 = -1 + (11 - (-1)) \cdot 5/6 = 9$$

$$Y_{c2} = y_1 + \Delta y \cdot u_2 = 7 + (1 - 7) \cdot 5/6 = 2$$

Vậy đường sau khi cắt tia là: I_1I_2 có $I_1(1,6)$ và $I_2(9,2)$

2.4. Giải thuật xén tia đa giác (Sutherland Hodgman)

2.4.1. Khái niệm

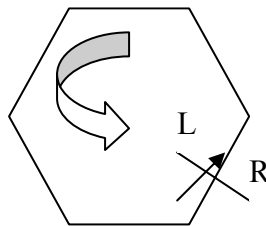


Hình 4.12 Các loại đa giác

- Đa giác lõm: là đa giác có đường thẳng nối bất kỳ 2 điểm bên trong nào của đa giác đều nằm trọn trong đa giác. Đa giác không lõm là đa giác lồi.
- Theo qui ước: một đa giác với các đỉnh P_1, \dots, P_N (các cạnh là $P_{i-1}P_i$ và P_NP_1) được gọi là theo hướng dương nếu các hình theo thứ tự đã cho tạo thành một mạch ngược chiều kim đồng hồ.

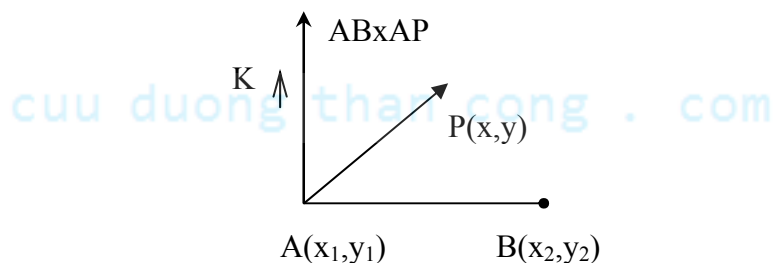
Nếu bàn tay trái của một người dọc theo bất kỳ cạnh $P_{i-1}P_i$ hoặc P_NP_1 cũng chỉ về bên trong đa giác.

cuu duong than cong . com



Hình 4.13 Qui tắc tìm hướng dương của một đa giác hướng

- Khảo sát một điểm so với một đường thẳng:



Hình 4.14 Khảo sát một điểm so với đường thẳng

Có điểm $A(x_1, y_1)$, $B(x_2, y_2)$ và $P(x, y)$

Theo định nghĩa thì tích của hai vectơ ($AB \times AP$) chỉ theo hướng của vectơ $K \perp$ với mặt phẳng (xoy). Có:

$$AB = (x_2 - x_1)i + (y_2 - y_1)j$$

$$AP = (x - x_1)i + (y - y_1)j$$

$$\text{Vậy } AB \times AP = [(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)] \cdot K$$

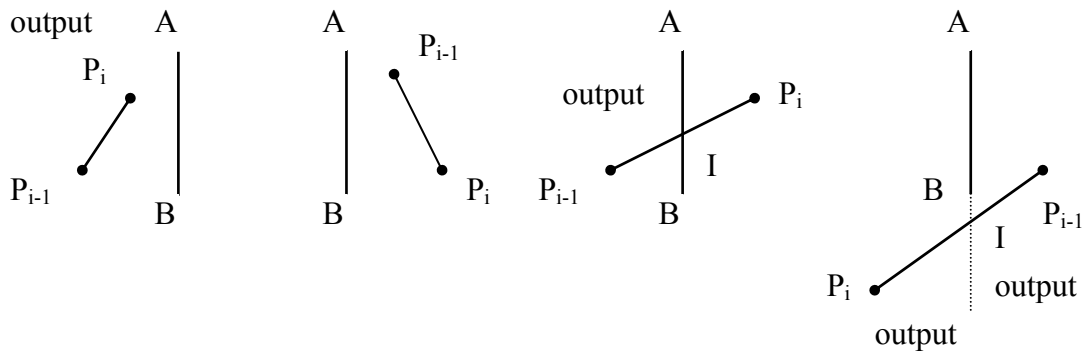
Do đó hướng được xác định như sau:

$$\bar{C} = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

- Nếu C dương thì P nằm bên trái AB
- Nếu C âm thì P nằm bên phải AB

2.4.2. Giải thuật Hodgman

Cho P_1, \dots, P_N là danh sách các đỉnh của đa giác đã bị cắt tía. Cho cạnh AB (điểm A, B) là cạnh bất kỳ. Đa giác cắt tía lỗi hướng dương.

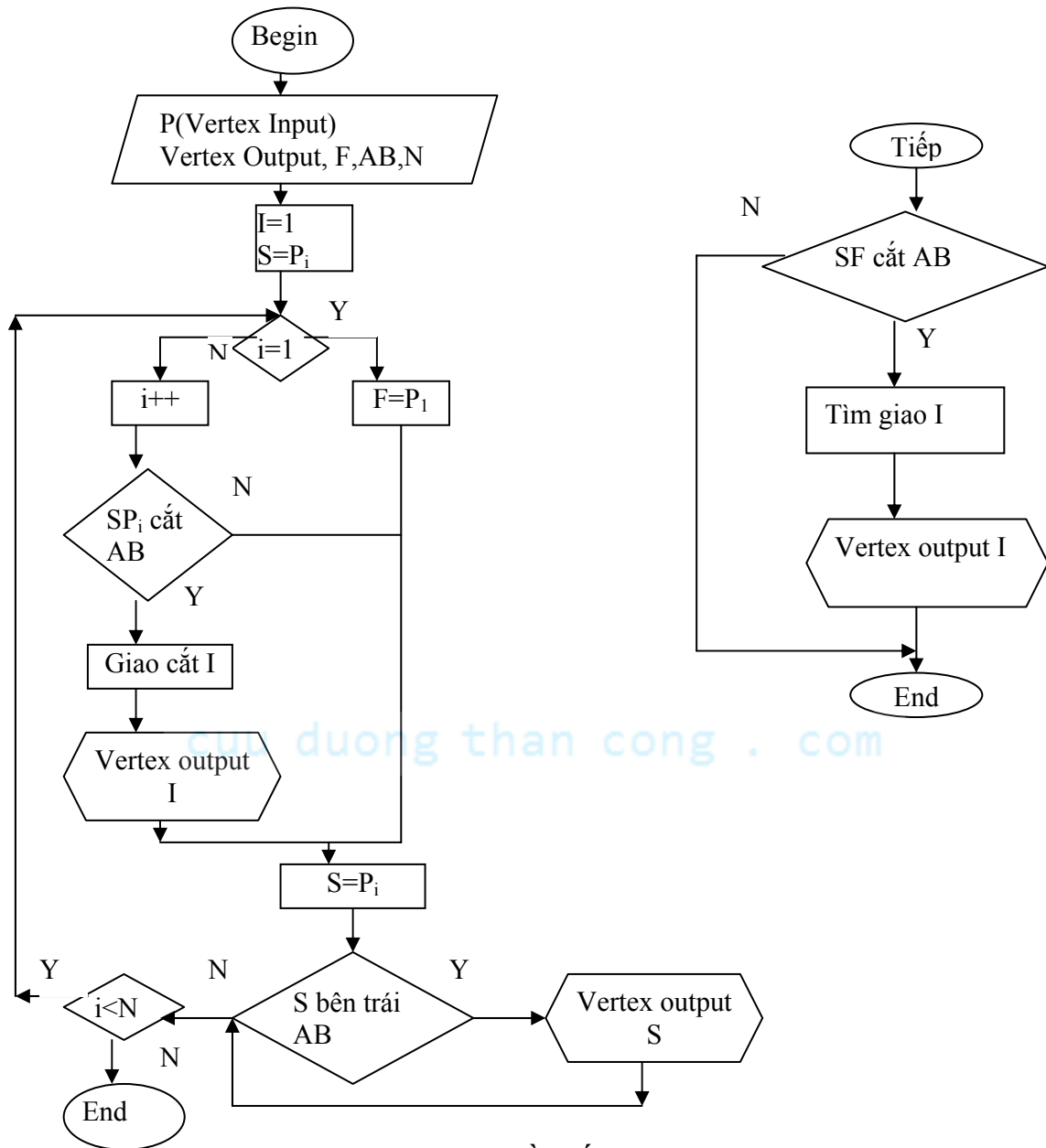


Hình 4.15 Các trường hợp trong giải thuật Hodgman

- ① Nếu cả P_{i-1} và P_i đều nằm bên trái của cạnh này thì P_i được lưu lại (đưa vào output) của đa giác cắt tía.
- ② Nếu cả P_{i-1} và P_i đều nằm bên phải của cạnh thì không có đỉnh nào được lưu lại.
- ③ Nếu P_{i-1} nằm bên trái và P_i nằm bên phải của cạnh thì giao điểm I của $P_{i-1}P_i$ với cạnh sẽ được lưu lại.
- ④ Nếu P_{i-1} nằm bên phải và P_i nằm bên trái thì cả giao điểm I và P_i đều được lưu lại.

2.4.3. Minh họa thuật toán Sutherland – Hodgman

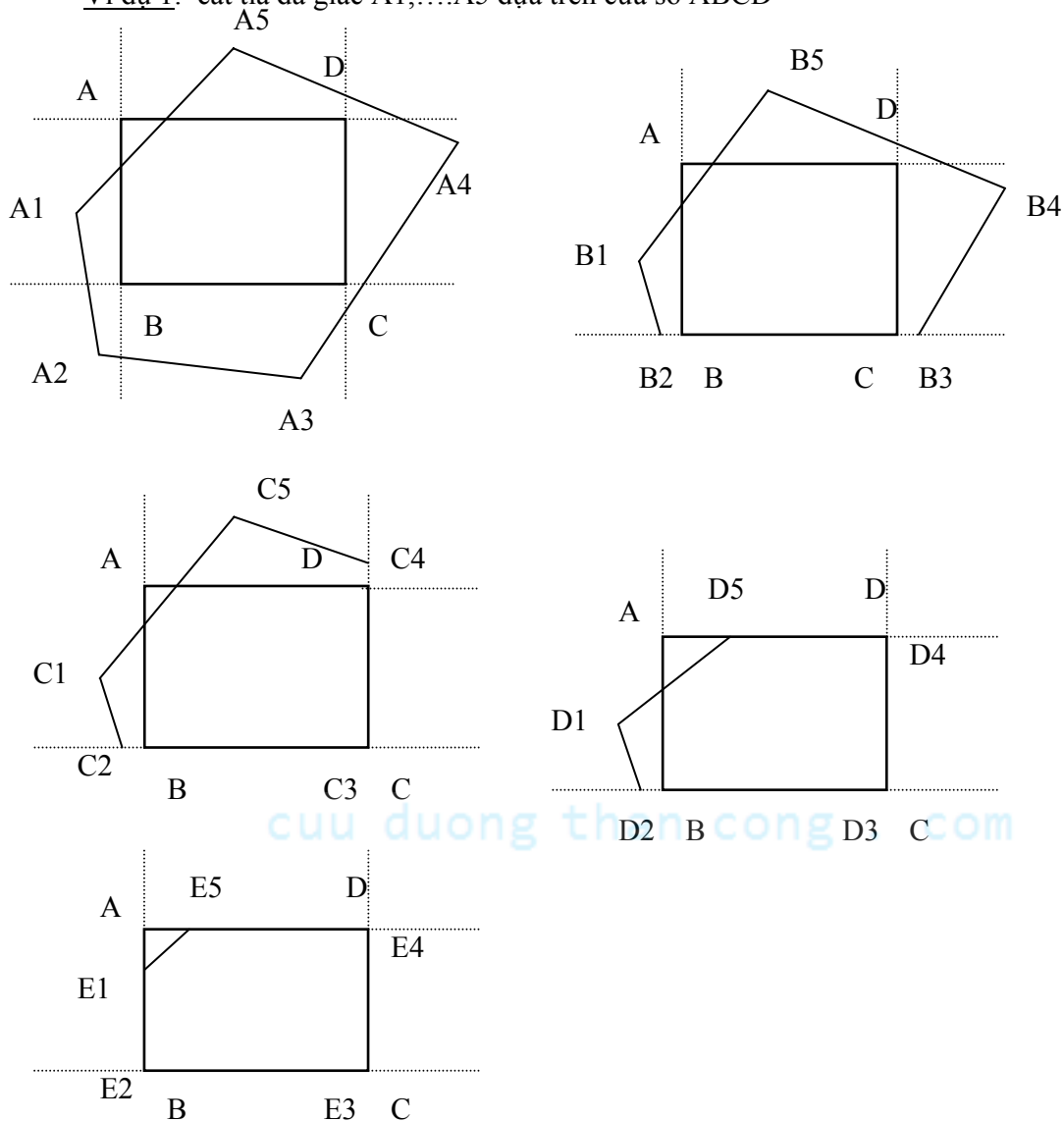
$F = P_1$ (F là đỉnh được đưa vào đầu tiên, S là đỉnh đưa vào trước P - đỉnh được đưa). Sơ đồ bên phải khép kín đa giác bởi $P_N P_1$



Hình 4.16 Sơ đồ khối thuật toán Hodgman

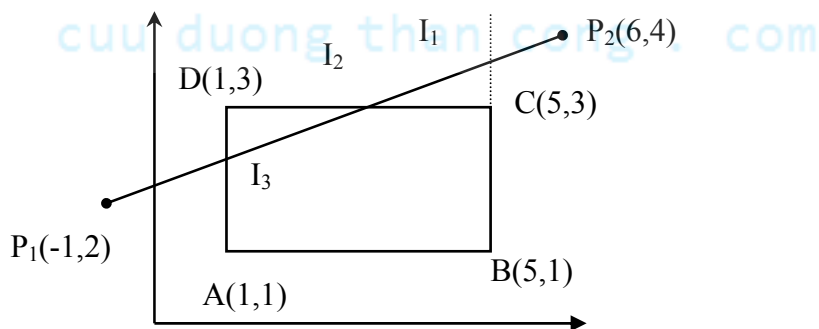
2.4.4. Ví dụ minh họa:

Ví dụ 1: cắt tia đa giác A_1, \dots, A_5 dựa trên cửa sổ ABCD



Hình 4.17 Ví dụ dùng giải thuật Hodgman xén tia đa giác $A_1 \dots A_5$

Ví dụ 2: hãy sử dụng thuật toán Hodgman để cắt xén đoạn thẳng nối $P_1(-1,2)$ đến $P_2(6,4)$ trên cửa sổ $A(1,1)$, $B(5,1)$, $C(5,3)$ và $D(1,3)$



Hình 4.18 Ví dụ dùng thuật toán Hodgman xén tia đoạn P_1P_2

Theo thuật toán Hodgman ta xén P_1P_2 dựa trên từng cạnh.

① AB: ta xét $C=(x_2-x_1)(y-y_1)-(y_2-y_1)(x-x_1)$

Điểm P_1 : $C=(5-1)(2-1)-(1-1)(-1-1) = 4 > 0$ nằm bên trái

Điểm P_2 : $C=(5-1)(4-1)-(1-1)(-1-1) = 12 > 0$ nằm bên trái

Vậy P_1P_2 đều được lưu

② BC

Điểm P_1 : $C=(5-5)(2-1)-(3-1)(-1-5) = 12 > 0$ nằm bên trái

Điểm P_2 : $C=(3-1)(6-5) = -2 < 0$ nằm bên phải

Giao điểm I_1 :

$$m = \frac{4-2}{6-(-1)} = \frac{2}{7}$$

$$\begin{cases} x_c = 5 \\ y_c = y_1 + (x_c - x_1)m = 2 + (5 - (-1)) \cdot \frac{2}{7} = 3\frac{5}{7} \end{cases}$$

$I_1(5, 26/7)$ vậy P_1I_1 lưu lại

③ CD

Điểm P_1 : $C=(1-5)(2-3) = 4 > 0$ nằm bên trái

Điểm I_1 : $C=(1-5)(26/7-3) = -20/7 < 0$ nằm bên phải

Giao điểm I_2 :

$$\begin{cases} y_c = 3 \\ x_c = x_1 + (y_c - y_1)/m \\ = -1 + (3 - 2) \cdot \frac{7}{2} = \frac{5}{2} \end{cases}$$

Vậy $I_2(5/2, 3)$ có P_1I_2 được lưu

④ DA

Điểm P_1 : $C=-(1-3)(-1-1) = -4$ nằm bên phải

Điểm I_2 : $C=-(1-3)(5/2-1) = 7/2$ nằm bên trái

Giao điểm I_3 :

$$\begin{cases} x_c = 1 \\ y_c = y_1 + (x_c - x_1)m \\ = 2 + (1 - (-1)) \cdot \frac{2}{7} = 2\frac{4}{7} \end{cases}$$

$I_3(1, 18/7)$

Tóm lại: Cắt xén đoạn P_1P_2 được đoạn I_2I_3 có $I_2(5/2, 3)$ và $I_3(1, 18/7)$

Tóm tắt chương:

Hiện thị đối tượng là quá trình đưa các mô tả đối tượng từ thế giới thực sang một thiết bị xuất cụ thể nào đó. Mỗi đối tượng có thể được quan sát ở nhiều vị trí và góc độ khác nhau. Thông

thường mỗi hình ảnh mà chúng ta quan sát được trên màn hình thiết bị được gọi là một góc nhìn (view) của đối tượng.

Quá trình loại bỏ các phần hình ảnh nằm ngoài một vùng cho trước được gọi là xén tia. Vùng được dùng để xén tia gọi là cửa sổ xén tia.

Các thuật toán xén tia đoạn thẳng Cohen-Sutherland, LyaBarsky đều tập trung giải quyết hai vấn đề chính nhằm tối ưu tốc độ thuật toán đó là: loại bỏ việc tìm giao điểm đối với các đoạn thẳng chắc chắn không cắt cửa sổ (như nằm hoàn toàn trong, nằm hoàn toàn ngoài), và với đoạn thẳng có khả năng cắt cửa sổ, cần phải tìm cách hạn chế số lần tìm giao điểm với các biên không cần thiết. Thuật toán Cohen-Sutherland giải quyết hai ý này thông qua kiểu dữ liệu mã vùng mà về bản chất đó chỉ là sự mô tả vị trí tương đối của vùng đang xét so với các biên của cửa sổ. Còn thuật toán LyaBarsky thì tuy dựa vào phương trình tham số của đoạn thẳng để lập luận nhưng thực chất là dựa trên việc xét các giao điểm có thể có giữa đoạn thẳng kéo dài với các biên của cửa sổ (cũng được kéo dài). Các giao điểm này tương ứng với các giá trị $r_k = q_k/p_k$. Cả hai thuật toán này đều có thể được mở rộng cho việc xén hình trong đồ họa 3D.

Bài tập:

1. Tìm phép biến đổi chuẩn hoá tạo ánh xạ cho một cửa sổ mà góc bên trái phía dưới của nó (1,1) và góc bên phải trên (3,5) vào:
 - a. Là màn hình thiết bị được chuẩn hoá toàn bộ.
 - b. Một công xem có góc bên trái dưới (0,0) và góc bên trái trên (1/2,1/2)
2. Cho cửa sổ cắt tia hình chữ nhật có góc bên trái dưới (-3,-2) và góc phải trên (2,3), dùng giải thuật Cohen Sutherland xén tia đoạn thẳng AB có tọa độ là A(3,2) và B(-2,-4).
3. Cho cửa sổ cắt tia hình chữ nhật có góc bên trái dưới (-3,-2) và góc phải trên (2,3), dùng giải thuật LyaBarsky xén tia đoạn thẳng AB có tọa độ là A(3,2) và B(-2,-4).
4. Viết chương trình cài đặt thuật toán Cohen Sutherland.
5. Viết chương trình cài đặt thuật toán LyaBarsky
6. Viết chương trình cài đặt thuật toán Sutherland Hodgman

Bài tập trắc nghiệm:

1. Hệ tọa độ thiết bị chuẩn (NDCS) có kích thước màn hình hiển thị là hình chữ nhật có chiều dài gấp đôi chiều rộng. Vậy nếu một hình chữ nhật đứng (có chiều dài gấp đôi chiều rộng) khi hiển thị trên màn hình sẽ cho:
 - a. Hình chữ nhật nằm ngang (chiều dài gấp đôi chiều rộng)
 - b. Hình vuông
 - c. Vẫn là hình chữ nhật đứng
 - d. Hình chữ nhật có chiều dài gấp 1.5 chiều rộng
2. Phép biến đổi chuẩn hoá tạo ánh xạ cho một cửa sổ mà góc bên trái phía dưới (1,3) và góc bên phải trên (4,7) vào màn hình được chuẩn hoá toàn bộ là:

a.
$$\begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ -\frac{1}{3} & -3\frac{1}{4} & 1 \end{bmatrix}$$

b.
$$\begin{bmatrix} -\frac{1}{3} & 1 & 2 \\ 0 & -2 & 0 \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

c.
$$\begin{bmatrix} \frac{1}{3} & 3 & 0 \\ 0 & -\frac{1}{3} & 0 \\ \frac{1}{4} & -3\frac{1}{4} & 1 \end{bmatrix}$$

d.
$$\begin{bmatrix} -2 & 0 & \frac{1}{3} \\ 0 & 1\frac{1}{3} & 0 \\ 0 & -3\frac{1}{4} & 1 \end{bmatrix}$$

3. Cho cửa sổ cắt tia góc trái dưới (-2,2) và góc phải trên (4,7). Mã vùng 4bít của điểm A(-3,-1) là:
- 1000
 - 1010
 - 0101
 - 0001
4. Cho cửa sổ cắt tia có góc trái dưới (2,-3) và phải trên (5,6), mã vùng 4bít của điểm M(6,10) là:
- 1010
 - 1100
 - 0101
 - 0001
5. Cho cửa sổ cắt tia có góc trái dưới (2,2) và góc phải trên (5,10). Cho đoạn AB có A(1,5) và B(6,8), sau khi xén tia ta thu được tọa độ mới là:
- $\left(3\frac{3}{5}, 5\right) \left(3\frac{2}{5}, 3\right)$

b. $\left(3\frac{1}{5}, 2\right) \quad \left(3, 3\frac{3}{5}\right)$

c. $\left(5, 3\frac{1}{5}\right) \quad \left(7\frac{2}{5}, 2\right)$

d. $\left(2, 5\frac{3}{5}\right) \quad \left(5, 7\frac{2}{5}\right)$

6. Cho cửa sổ cắt tỉa có góc trái dưới (1,2) và phải trên (9,8). Đoạn thẳng MN có M(-1,7) và N(11,1), sau khi xén tỉa ta thu được đoạn thẳng có tọa độ là:

- a. (2,6) và (7,1)
- b. (5,2) và (9,1)
- c. (1,6) và (9,2)
- d. (3,9) và (6,3)

7. Giải thuật LyaBarsky dựa vào phương trình đường thẳng:

- a. Tường minh $y=f(x)$
- b. Không tường minh $f(x,y) = 0$
- c. Tham số $x = x(t)$, $y = y(t)$ có $t \in [0,1]$
- d. Do ông đưa ra

8. Trong các câu nói sau câu nào sai ?

- a. Đoạn thẳng được hiển thị khi cả hai đầu cuối đều trong cửa sổ hiển thị
- b. Đoạn thẳng nằm hoàn toàn ngoài khi nó phạm bất kỳ một trong bốn bất đẳng thức sau:

$$x_1, x_2 > x_{\max} \vee x_1, x_2 < x_{\min} \vee y_1, y_2 > y_{\max} \vee y_1, y_2 < y_{\min}$$

- c. Đoạn thẳng được hiển thị khi:

$$P_1.m \text{ or } P_2.m == 0000 \quad (P_1, P_2 \text{ là hai điểm cuối})$$

- d. Đoạn thẳng nằm hoàn toàn ngoài khi nó thỏa mãn một trong bốn bất đẳng thức sau:

$$x_1, x_2 >= x_{\max} \vee x_1, x_2 <= x_{\min} \vee y_1, y_2 >= y_{\max} \vee y_1, y_2 <= y_{\min}$$

9. Đoạn mã sau cài đặt giải thuật Cohen Sutherland, gán mã vùng 4bit cho mỗi điểm cuối của đoạn thẳng xén tỉa. Từ giải thuật đã được học bạn hãy cho biết thứ tự từ ①, ②, ③, ④ sẽ có mã vùng lần lượt sẽ là:

① #define EDGE_1 0x1

② #define EDGE_2 0x2

③ #define EDGE_3 0x4

④ #define EDGE_4 0x8

- a. Trái, phải, trên và dưới
- b. Trái, phải, dưới và trên
- c. Trên, dưới, trái và phải
- d. Phải, trái, trên và dưới

10. Hàm sau là một hàm trong giải thuật Cohen Sutherland. Bạn hãy cho biết lần lượt các dòng lệnh ①, ②, ③, ④ sẽ cho các mã vùng 4bit là: (chiều theo định nghĩa mã vùng 4bit trong sách kỹ thuật đồ họa)

```
unsigned char encode(double x, double y, double xmin, double ymin,
double xmax, double ymax)
{
    unsigned char code = 0x00;
    if (x < xmin)
        code = code | LEFT_EDGE; //①
    if (x > xmax)
        code = code | RIGHT_EDGE; //②
    if (y < ymin)
        code = code | TOP_EDGE; //③
    if (y > ymax)
        code = code | BOTTOM_EDGE; //④
    return code;
}
```

- a. 0x8, 0x4, 0x2 và 0x1
- b. 0x1, 0x2, 0x4 và 0x8
- c. 0x1, 0x2, 0x8 và 0x4
- d. 0x2, 0x8, 0x4 và 0x1

11. Bạn hãy cho biết hàm sau là một trong những hàm để cài đặt cho giải thuật xén tia nào?

```
#define TRUE 1
#define FALSE 0
int cliptest(double p, double q, double *u0, double *u1)
{
    double r;
    int retVal = TRUE;
    if (p < 0.0)
    {
        r = q / p;
        if (r > *u1)
            retVal = FALSE;
        else
            if (r > *u0)
                *u0 = r;
    }
    else
        if (p > 0.0)
        {
            r = q / p;
            if (r < *u0)
                retVal = FALSE;
            else
                if (r < *u1)
                    *u1 = r;
        }
    else

```

```
    {  
        if (q < 0.0)  
            retVal = FALSE;  
    }  
    return (retVal);  
}
```

a. Cohen Sutherland
b. Lyabarsky
c. Hodgman
d. Không phải giải thuật xén tia

cuu duong than cong . com

cuu duong than cong . com

CHƯƠNG 5: PHÉP CHIẾU – PROJECTION

1. KHÁI NIỆM CHUNG

1.1. Nguyên lý về 3D (three-Dimension)

Đồ họa 3 chiều (3D computer graphics) bao gồm việc bổ xung kích thước về chiều sâu của đối tượng, cho phép ta biểu diễn chúng trong thế giới thực một cách chính xác và sinh động hơn.

Tuy nhiên các thiết bị truy xuất hiện tại đều là 2 chiều, Do vậy việc biểu diễn được thực thi thông qua phép tô chất (render) để gây ảo giác (illusion) về độ sâu

Đồ họa 3D là việc chuyển thế giới tự nhiên dưới dạng các mô hình biểu diễn trên các thiết bị hiển thị thông qua kỹ thuật tô chất (rendering).

1.2. Đặc điểm của kỹ thuật đồ họa 3D

Có các đối tượng phức tạp hơn các đối tượng trong không gian 2D

- Bao bởi các mặt phẳng hay các bề mặt
- Có các thành phần trong và ngoài

Các phép biến đổi hình học phức tạp

Các phép biến đổi hệ tọa độ phức tạp hơn

Thường xuyên phải bổ xung thêm phép chiếu từ không gian 3D vào không gian 2D

Luôn phải xác định các bề mặt hiển thị

1.3. Các phương pháp hiển thị 3D

Với các thiết bị hiển thị 2D thì chúng ta có các phương pháp sau để biểu diễn đối tượng 3D:

- Kỹ thuật chiếu (projection): Trục giao (orthographic)/phối cảnh (perspective)
- Kỹ thuật đánh dấu độ sâu (depth cueing)
- Nét khuất (visible line/surface identification)
- Tô chất bề mặt (surface rendering)
- Cắt lát (exploded/cutaway scenes, cross-sections)

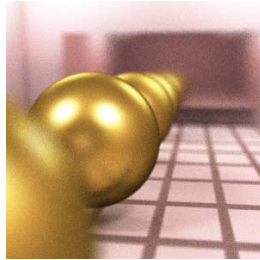
Các thiết bị hiển thị 3D:

- Kính stereo - Stereoscopic displays*
- Màn hình 3D – Holograms

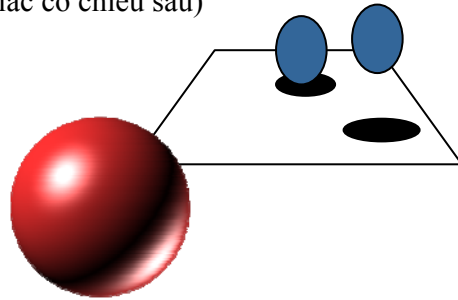
Exploded/cutaway
scenes (cắt lát)



Perspective (phối
cảnh) and Depth of
Field (chiều sâu)



Shadows as depth cues (tạo bóng cảm
giác có chiều sâu)



Hình 5.1 Các cách mô tả đối tượng 3D



Hình chiếu cạnh



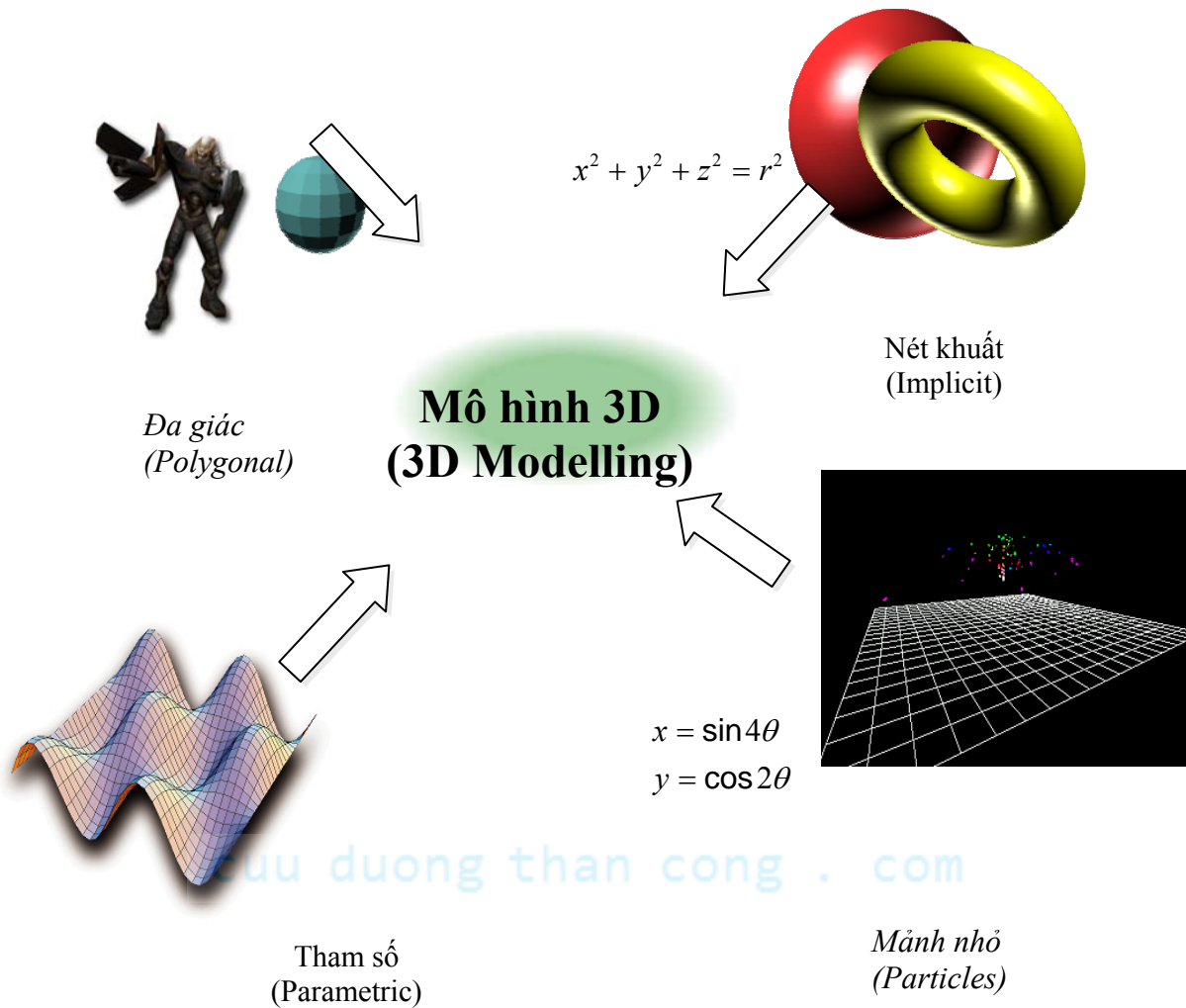
Hình chiếu bằng



Hình chiếu đứng

Hình 5.2 Các góc nhìn khác nhau của mô hình 3D

cuu duong than cong . com



Hình 5.3 Các cách mô tả đối tượng 3D

2. PHÉP CHIẾU

Định nghĩa về phép chiếu

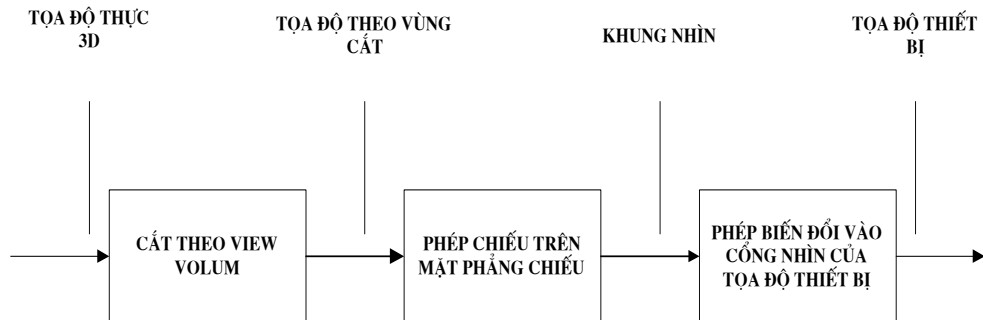
Một cách tổng quát, phép chiếu là phép chuyển đổi những điểm của đối tượng trong hệ thống tọa độ n chiều thành những điểm trong hệ thống tọa độ có số chiều nhỏ hơn n .

Định nghĩa về hình chiếu

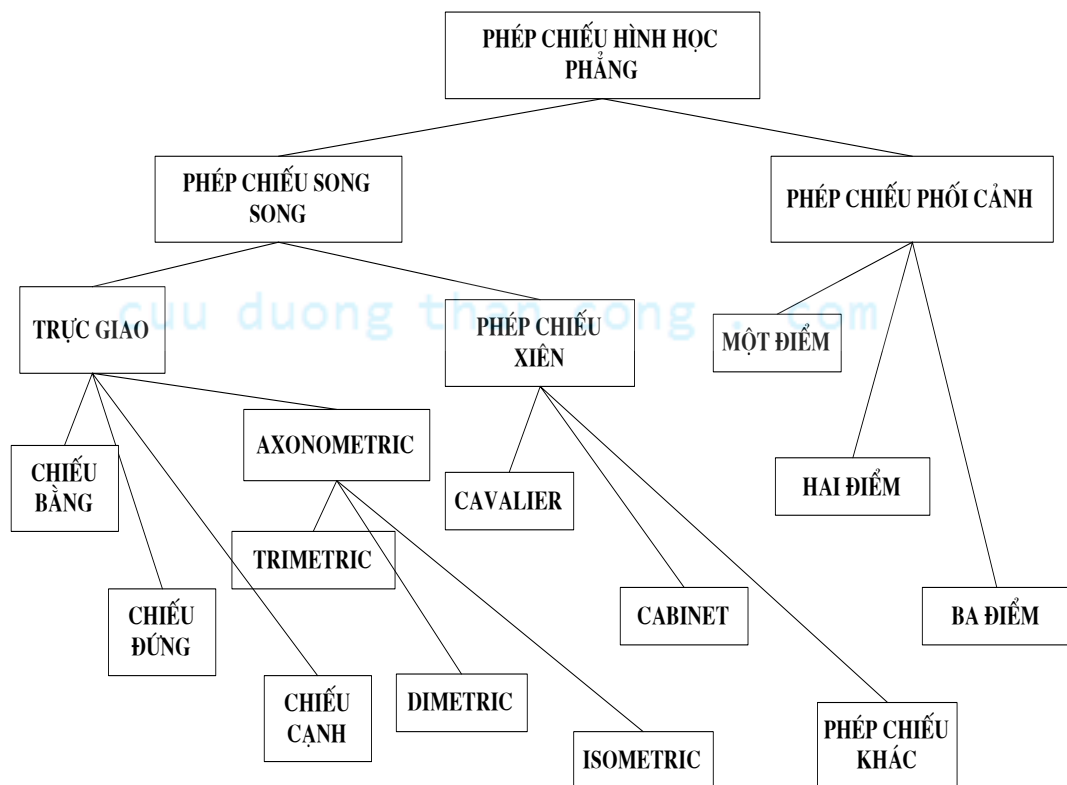
Ảnh của đối tượng trên mặt phẳng chiếu được hình thành từ phép chiếu bởi các đường thẳng gọi là tia chiếu (projector) xuất phát từ một điểm gọi là tâm chiếu (center of projection) đi qua các điểm của đối tượng giao với mặt chiếu (projection plan).

Các bước xây dựng hình chiếu

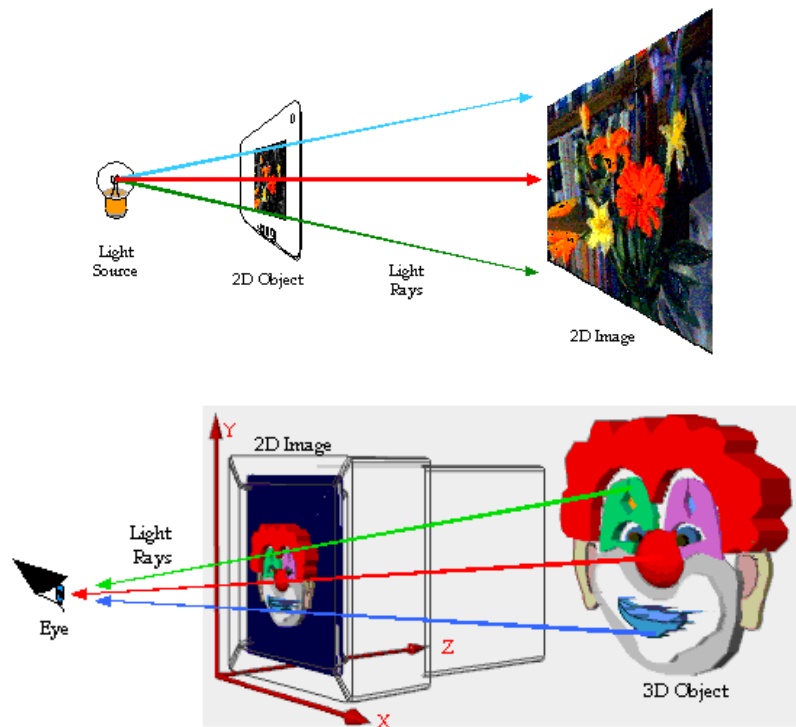
1. Đối tượng trong không gian 3D với tọa độ thực được cắt theo một không gian xác định gọi là view volume.
2. View volume được chiếu lên mặt phẳng chiếu. Diện tích choán bởi view volume trên mặt phẳng chiếu đó sẽ cho chúng ta khung nhìn.
3. Là việc ánh xạ khung nhìn vào trong một cổng nhìn bất kỳ cho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



Hình 5.4 Mô hình nguyên lý của tiến trình biểu diễn đối tượng 3D



Hình 5.6 Phân loại các phép chiếu



Hình 5.5 Ví dụ minh họa các phép chiếu phối cảnh

cuu duong than cong . com

3. 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*).

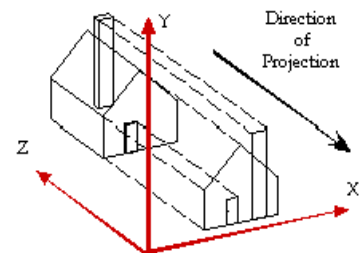
3.1. Phép chiếu trực giao (Orthographic projection)

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.

cuu duong than cong . com

$$[T_y] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [T_x] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [T_z] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Hình 5.7 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 bằng 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 tọa 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.

3.2. 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 phẳng chiếu là $z=0$).

Có 3 phép chiếu

- Phép chiếu Trimetric
- Phép chiếu Dimetric
- Phép chiếu Isometric

3.2.1. Phép chiếu Trimetric

Là phép chiếu hình thành từ việc quay tự do đối tượng trên một trục hay tất cả các trục của hệ tọa độ và chiếu đối tượng đó bằng phép chiếu song song lên mặt phẳng chiếu (thường là mặt phẳng $z = 0$).

Ngoại trừ những mặt phẳng của đối tượng song song với mặt phẳng chiếu không thay đổi. Các mặt khác của đối tượng hay hình dạng của đối tượng thường biến dạng sau phép chiếu. Tuy nhiên tỉ lệ co (Shortening Factor - SF) là tỷ số của độ dài đoạn thẳng chiếu so với độ dài thực tế của đối tượng. Trên cơ sở SF phép chiếu trục lượng được chia làm ba loại sau:

- Phép chiếu Trimetric
- Phép chiếu Dimetric
- Phép chiếu Isometric

Việc tính các giá trị SF này của các trục tương ứng dựa vào công thức $[U]^* [T]$.

$$[U] = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad [T] = \begin{bmatrix} x'_x & y'_x & 0 & 1 \\ x'_y & y'_y & 0 & 1 \\ x'_z & y'_z & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[U]: là ma trận vector đơn vị của các trục x, y, z bất biến

[T]: là ma trận chiếu tổng hợp tương ứng

SF- tỉ lệ co theo các trục là:

$$f_x = \sqrt{x_x'^2 + y_x'^2}$$

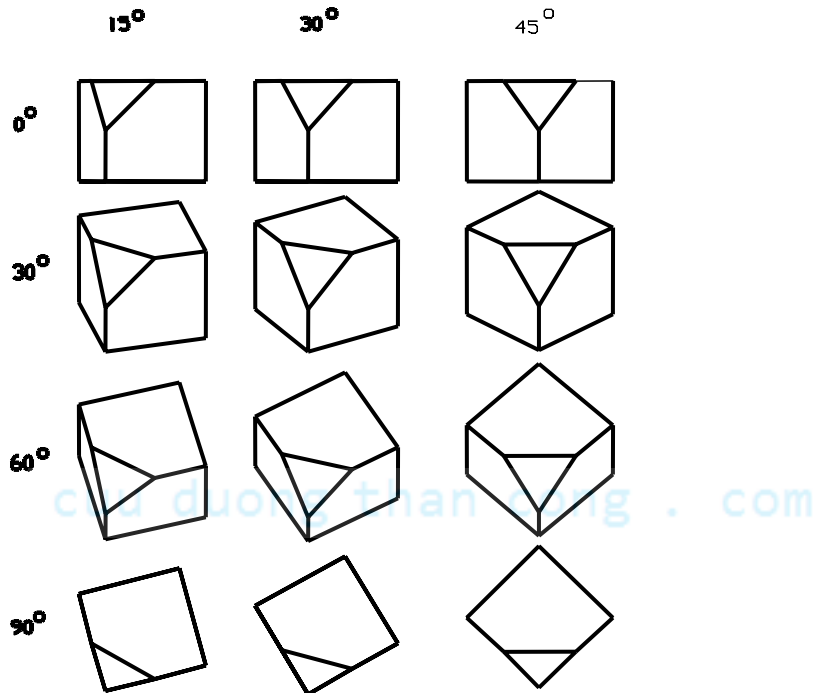
$$f_y = \sqrt{x_y'^2 + y_y'^2}$$

$$f_z = \sqrt{x_z'^2 + y_z'^2}$$

3.2.2. Phép chiếu Dimetric

Là phép chiếu Trimetric với 2 hệ số tỉ lệ co bằng nhau, giá trị thứ 3 còn lại là tùy ý.

Phép chiếu được xây dựng bằng cách quay đối tượng quanh trục y theo một góc ϕ , tiếp đó quanh trục ox một góc φ và sau cùng là phép chiếu trên mặt phẳng $z=0$ với tâm chiếu tại vô tận.



Hình 5.8 Ảnh của phép chiếu Trimetric với các tham số góc xoay thay đổi

$$[T] = [Ry][Rx][Pz]$$

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

$$[T] = \begin{bmatrix} \cos \phi & \sin \phi \sin \varphi & 0 & 0 \\ 0 & \cos \varphi & 0 & 0 \\ \sin \phi & -\cos \phi \sin \varphi & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_x' & y_x' & 0 & 0 \\ x_y' & y_y' & 0 & 0 \\ x_z' & y_z' & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$f_x^2 = (x_x'^2 + y_x'^2) = \cos^2 \phi + \sin^2 \phi \cdot \sin^2 \varphi$$

$$f_y^2 = (x_y'^2 + y_y'^2) = \cos^2 \varphi$$

Tỷ lệ co trên x và y bằng nhau nên ta có:

$$\cos^2 \phi = \cos^2 \phi + \sin^2 \phi \cdot \sin^2 \phi$$

$$1 - \sin^2 \phi = 1 - \sin^2 \phi + \sin^2 \phi \cdot \sin^2 \phi$$

$$\sin^2 \phi (\sin^2 \phi - 1) = -\sin^2 \phi$$

$$\sin^2 \phi = \frac{\sin^2 \phi}{1 - \sin^2 \phi}$$

$$f_z^2 = (x_z'^2 + y_z'^2) = \sin^2 \phi + \cos^2 \phi \sin^2 \phi$$

$$= \sin^2 \phi + \sin^2 \phi (1 - \sin^2 \phi) = \sin^2 \phi + \sin^2 \phi (1 - \sin^2 \phi)$$

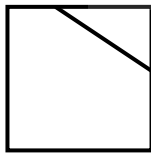
$$\sin^2 \phi = \frac{f_z^2 - \sin^2 \phi}{1 - \sin^2 \phi} = \frac{\sin^2 \phi}{1 - \sin^2 \phi}$$

$$\sin^2 \phi = \frac{f_z}{2}$$

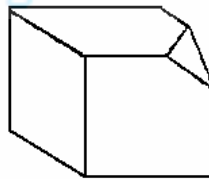
$$\phi = \sin^{-1} \left(\pm \frac{f_z}{\sqrt{2 - f_z^2}} \right)$$

$$\phi = \sin^{-1} \left(\pm \frac{f_z}{\sqrt{2}} \right)$$

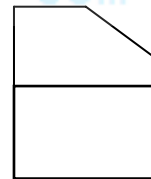
Ta thấy tỷ lệ co thuộc khoảng [0 1], với mỗi f_z ta có bốn khả năng của phép chiếu.



$$f_z = 0$$



$$f_z = 1/2$$



$$f_z = 1$$

Hình 5.9 Phép chiếu hình hộp với $f=0$, $f=1/2$ và $f=1$

Khi $f_z = 1/2$ thì:

$$\phi = \sin^{-1} \left(\pm \frac{1}{2\sqrt{2}} \right) \approx \sin^{-1} (\pm 0.35355) \approx \pm 20.705^\circ$$

$$\phi = \sin^{-1} \left(\pm \frac{1/2}{\sqrt{7/4}} \right) = \sin^{-1} (\pm 0.378) \approx \pm 22.208^\circ$$

3.2.3. Phép chiếu Isometric

Là phép chiếu trực lượng mà ở đó hệ số co cạnh trên 3 trục là bằng nhau.

Góc quay tương ứng là 35.26° và 45°

Được ứng dụng nhiều trong việc xây dựng các góc quan sát chuẩn cho đối tượng trong các hệ soạn thảo đồ họa.

$$\sin^2 \phi = \frac{1 - 2 \sin^2 \phi}{1 - \sin^2 \phi} \quad \sin^2 \phi = \frac{\sin^2 \phi}{1 - \sin^2 \phi}$$

$$\sin \varphi = \pm \frac{1}{\sqrt{3}}$$

$$\varphi = \pm 35.26^\circ$$

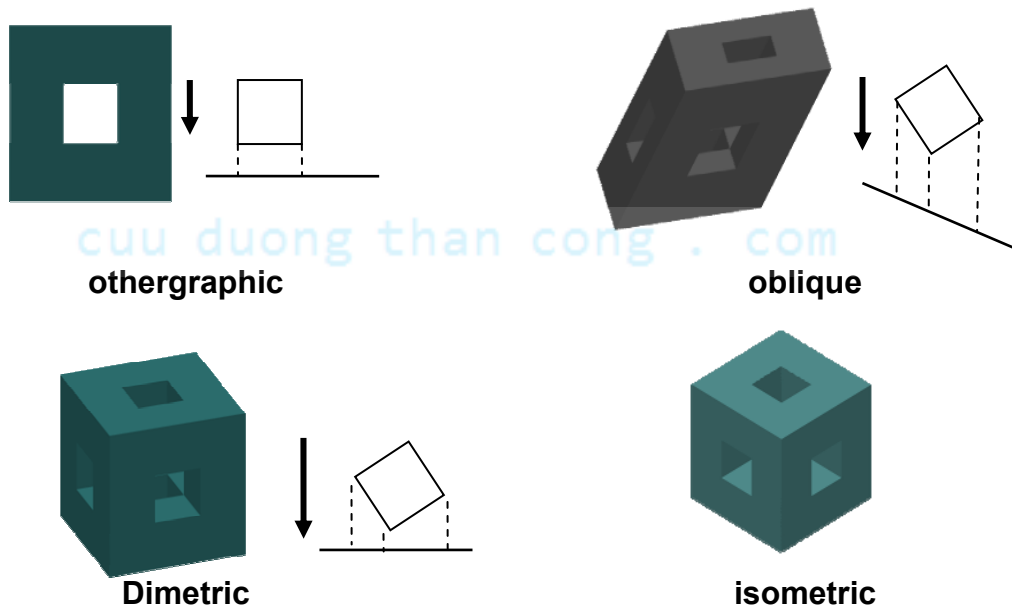
$$\sin^2 \phi = \frac{\sin^2 \varphi}{1 - \sin^2 \varphi} = \frac{1/3}{1 - 1/3} = 1/2$$

$$\sin \phi = \pm \frac{1}{\sqrt{2}}$$

$$\phi = \pm 45^\circ$$

$$\varphi = \pm 35.26^\circ \quad f = \sqrt{\cos^2 \varphi} = \sqrt{2/3} = 0.8165$$

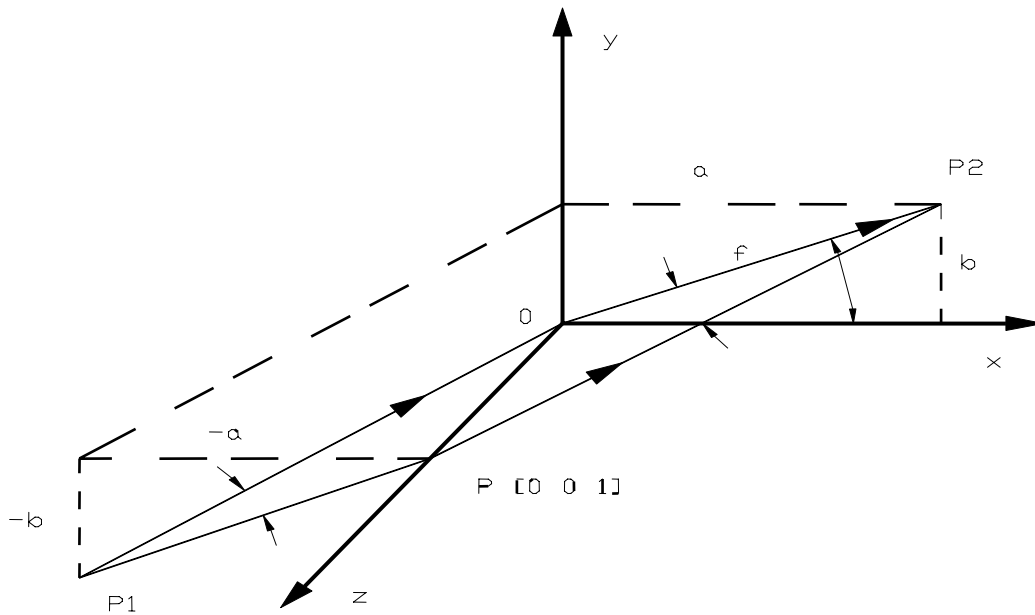
Ví dụ các phép chiếu song song:



Hình 5.10 Ví dụ các phép chiếu song song

3.3. Phép chiếu xiên - Oblique

- Phép chiếu Cavalier
- Phép chiếu Cabinet



Hình 5.11 Phép chiếu với tâm chiếu trên trục oz

3.3.1. Phép chiếu Cavalier

Phép chiếu cavalier là phép chiếu xiên được tạo thành khi các tia chiếu làm thành với mặt phẳng chiếu một góc 45°

Hệ số co trên các hệ trục tọa độ bằng nhau.

Ta có vector đơn vị theo trục z là $P=[0 \ 0 \ 1]$ tia chiếu làm với mặt phẳng một góc và p được chiếu lên mặt phẳng chiếu.

Ta có P1O và PP2 là 2 tia chiếu của phép chiếu xiên với góc β với mặt phẳng chiếu.

P2 là ảnh của P

P1 là điểm dịch chuyển của P tạo ra tia chiếu song song PP2

Tia chiếu P1O có thể thu được từ phép biến đổi P đến điểm $[-a-b]$

Trong không gian hai chiều dịch chuyển sẽ là:

$$T' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & -b & 1 \end{bmatrix}$$

Trong không gian ba chiều, f là chiều dài của hình chiếu vector đơn vị trục z trên mặt phẳng chiếu, đó chính là hệ số co.

$$[T''] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -a & -b & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} a = f \cos \alpha \\ b = f \sin \alpha \end{matrix} \quad \hookrightarrow \quad [T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -f \cos \alpha & -f \sin \alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$f = 0, b = 90^\circ$ phép chiếu sẽ trở thành phép chiếu trực giao.

Còn với $f = 1$ kích thước của hình chiếu bằng kích thước của đối tượng \Rightarrow cavalier

Phép chiếu Cavalier cho phép giá trị của a biến đổi một cách tự do $a = 30^\circ$ và 45°

3.3.2. Phép chiếu Cabinet

Phép chiếu xiên với hệ số co tỉ lệ $f = 1/2$.

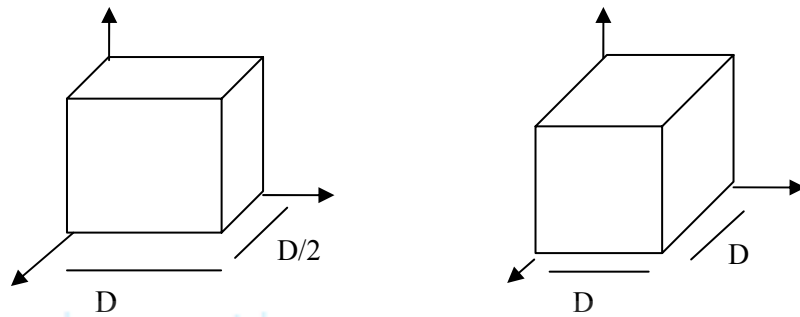
Với phép chiếu cabinet, giá trị của α có thể thay đổi tùy ý. Thông thường thì các giá trị hay được sử dụng là 30° và 45° .

Theo phép chiếu xiên một số mặt của đối tượng có thể được hiển thị như hình dạng thật cho nên rất phù hợp với việc mô tả các đối tượng có hình dạng tròn hay các bề mặt cong.

$$\beta = \cos^{-1}\left(\frac{f}{\sqrt{1^2 + f^2}}\right)$$

$$= \cos^{-1}\left(\frac{1/2}{\sqrt{1^2 + (1/2)^2}}\right) = 63.435^\circ$$

Ví dụ về phép chiếu xiên (Oblique Projections)



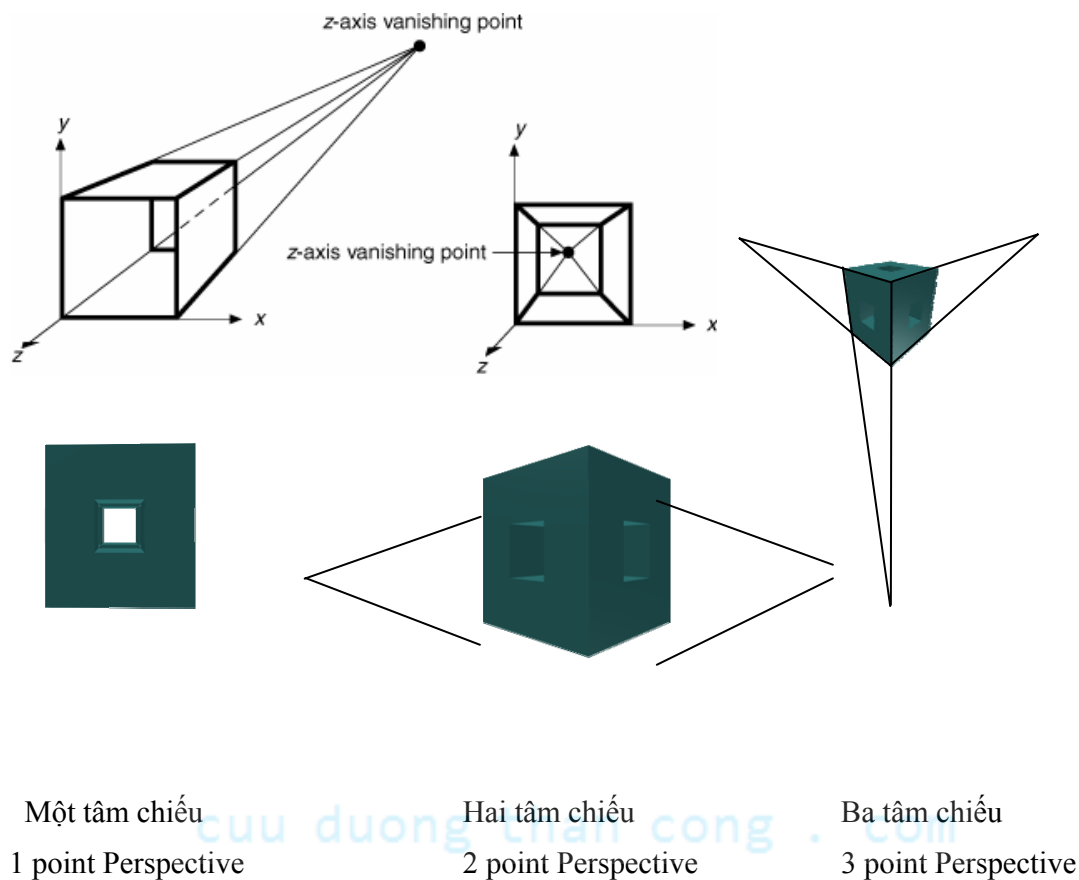
Hình 5.12 Ví dụ các phép chiếu xiên hình hộp

4. PHÉP CHIẾU PHỐI CẢNH (Perspective Projection)

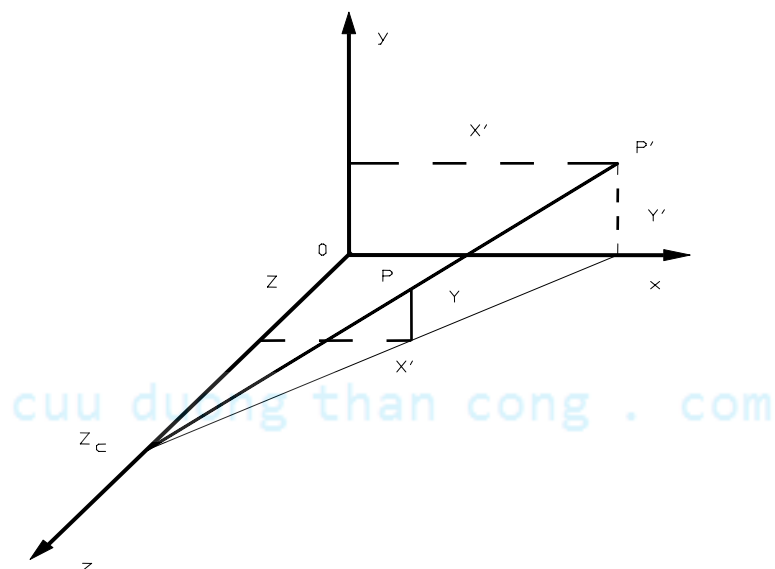
Phép chiếu phối cảnh là phép chiếu mà các tia chiếu không song song với nhau mà xuất phát từ một điểm gọi là tâm chiếu. Phép chiếu phối cảnh tạo ra hiệu ứng về luật xa gần tạo cảm giác về độ sâu của đối tượng trong thế giới thật mà phép chiếu song song không lột tả được.

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 5.13 Phép biến đổi phối cảnh



Hình 5.14 Phép chiếu với tâm chiếu trên trục z

Phép chiếu phối cảnh của các điểm trên đối tượng lên trên mặt phẳng 2D thu được từ phép chiếu trực giao và phép biến đổi phối cảnh.

4.1. Phép chiếu phối cảnh một tâm chiếu

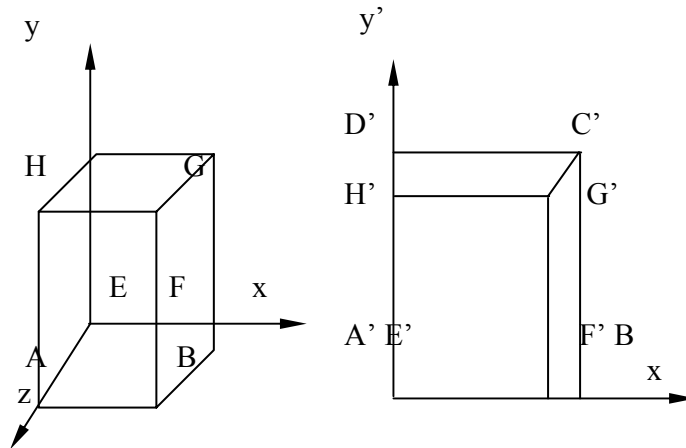
Giả sử khi mặt phẳng được đặt tại $z = 0$ và tâm phép chiếu nằm trên trục z , cách trục z một khoảng $zc = -1/r$.

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

Phương trình biến đổi:

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} [Tr] = \begin{bmatrix} x & y & z & rz+1 \end{bmatrix}$$

ma trận biến đổi một điểm phối cảnh $[Tr]$ có dạng:

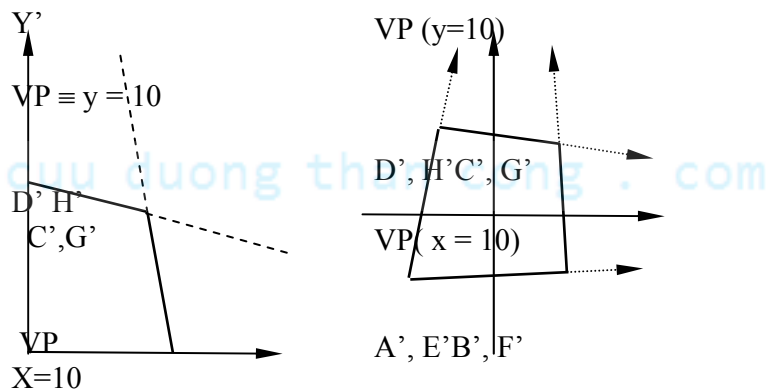


5.15 Phép chiếu phối cảnh một tâm chiếu

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

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

4.2. Phép chiếu phối cảnh hai tâm chiếu



5.16 Phép chiếu phối cảnh hai tâm chiếu

$$[T_c] = [T_{pq}][T_z]$$

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

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

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

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

Hai tâm chiếu: $\begin{bmatrix} -1/p & 0 & 0 & 1 \end{bmatrix}$ và $\begin{bmatrix} 0 & -1/q & 0 & 1 \end{bmatrix}$

Điểm triệt tiêu (VP -Vanishing point) tương ứng trên 2 trục x và y là điểm: $\begin{bmatrix} 1/p & 0 & 0 & 1 \end{bmatrix}$ và $\begin{bmatrix} 0 & 1/q & 0 & 1 \end{bmatrix}$.

4.3. Phép chiếu phối cảnh ba tâm chiếu

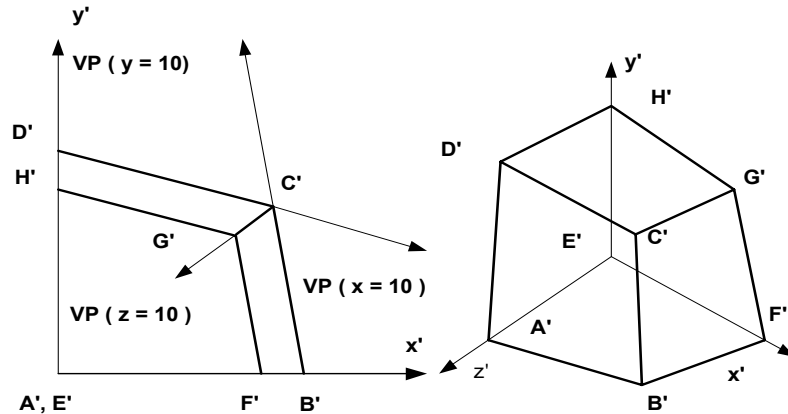
$$[T_{pqr}] = [T_p][T_q][T_r]$$

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

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

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

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



Hình 5.17 Phép chiếu phối cảnh ba tâm chiếu

Ba tâm chiếu: trên trục x tại điểm $\begin{bmatrix} -1/p & 0 & 0 & 1 \end{bmatrix}$,

y tại điểm $\begin{bmatrix} 0 & -1/q & 0 & 1 \end{bmatrix}$

z tại điểm $\begin{bmatrix} 0 & 0 & -1/r & 1 \end{bmatrix}$

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

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

Tóm tắt chương:

Chúng ta xét hai phép chiếu song song và phối cảnh, trong phép chiếu song song thì phân ra các loại: trục giao, trục lượng và phép chiếu xiên. Phép chiếu trục giao chỉ đơn giản là bỏ đi một trong ba tọa độ của điểm chiếu bằng cách cho các tia chiếu song song với một trong các trục tọa độ. Phép chiếu trục lượng thì quay đối tượng khi thấy được rõ nhất đối tượng rồi mới cho các tia chiếu song song với các trục tọa độ. Phép chiếu xiên thì tia chiếu không song song với trục tọa độ mà thay bằng nó làm với các mặt phẳng chiếu một góc không vuông.

Phép chiếu phối cảnh thì sử dụng một điểm cố định gọi là tâm chiếu và hình chiếu của các điểm được xác định bằng giao điểm của tia chiếu (nối điểm chiếu và tâm chiếu) với mặt phẳng quan sát. Phép chiếu phối cảnh hội tụ tại mắt nên đối tượng càng xa trông càng nhỏ và ngược lại.

Bài tập:

1. Cho Hình vuông ABCD có các tọa độ là: A(0,0,0), B(0,2,0), C(2,2,2) và D(2,0,2). Tính tọa độ mới của hình vuông sau khi chiếu nó bởi phép chiếu Isometric?

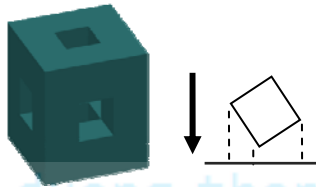
2. Cho Hình vuông ABCD có các tọa độ là: A(0,0,0), B(0,2,0), C(2,2,2) và D(2,0,2). Tính tọa độ mới của hình vuông sau khi chiếu nó bởi phép chiếu Dimetric với $fz=1/2$ (tỷ lệ co theo trục z)?

3. Cho tam giác ABC có các toạ độ là A(2,3,1), B(0,4,6) và C(5,2,7), Hãy tính toạ độ mới của hình tam giác đó sau khi chiếu phối cảnh sau:
 - Một tâm chiếu tại P(0,0,10)
 - Hai tâm chiếu tại M(5,0,0) và N(0,-8,0)
 - Ba tâm chiếu tại M(4,0,0), N(0,-6,0) và P(0,0,12)
4. Viết chương trình chiếu hình kim cương (ABCD) với phép chiếu trục giao.
5. Viết chương trình chiếu hình lập phương với phép chiếu
 - Trimetric
 - Dimetric
 - Isometric
6. Viết chương trình chiếu hình kim cương (ABCD) với phép chiếu xiên
 - Cavalier
 - Cabinet
7. Viết chương trình chiếu hình lập phương với phép chiếu phối cảnh.

Bài tập trắc nghiệm:

1. Phép chiếu là phép chuyển đổi những điểm của đối tượng trong hệ thống toạ độ n chiều thành những điểm trong hệ thống toạ độ có số chiều là Các phương án sau điền vào chỗ chấm thì phương án nào sai:
 - a. Có số chiều nhỏ hơn n
 - b. Thường là (n-1)
 - c. Tổng quát là số chiều nhỏ hơn n, thường (n-1)
 - d. Có số chiều (n+1)
2. Chương trình AutoCad đã sử dụng phép chiếu:
 - a. Trục giao
 - b. Trục lượng
 - c. Xiên
 - d. Phối cảnh
3. Phép chiếu Dimetric là phép chiếu song song có các tia chiếu vuông góc với màn chiếu, hình chiếu thu được sau khi quay đối tượng sao cho 3 mặt của đối tượng được trông thấy (thường mặt phẳng chiếu là $z=0$) và hệ số co Phương án nào điền vào chỗ chấm là đúng?
 - a. $f_x \neq f_y \neq f_z$
 - b. $f_x = f_y$
 - c. $f_x = f_y = f_z = \sqrt{\frac{2}{3}}$
 - d. $f_x = f_y = f_z = 1/2$
4. Phép chiếu trimetric là phép chiếu song song có các tia chiếu vuông góc với màn chiếu, hình chiếu thu được sau khi quay đối tượng sao cho 3 mặt của đối tượng được trông thấy (thường mặt phẳng chiếu là $z=0$) và hệ số co Phương án nào điền vào chỗ chấm là đúng?

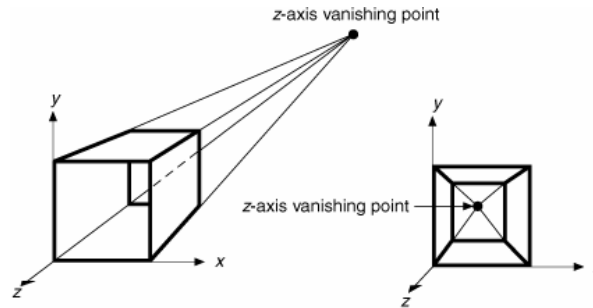
- a. $f_x \neq f_y \neq f_z$
 - b. $f_x = f_y$
 - c. $f_x = f_y = f_z = \sqrt{\frac{2}{3}}$
 - d. $f_x = f_y = f_z = 1/2$
5. Phép chiếu Isometric là phép chiếu song song có các tia chiếu vuông góc với màn chiếu, hình chiếu thu được sau khi quay đối tượng sao cho 3 mặt của đối tượng được trông thấy (thường mặt phẳng chiếu là $z=0$) và hệ số co Phương án nào điền vào chỗ trống là đúng?
- a. $f_x \neq f_y \neq f_z$
 - b. $f_x = f_y$
 - c. $f_x = f_y = f_z = \sqrt{\frac{2}{3}}$
 - d. $f_x = f_y = f_z = 1/2$
6. Hình sau là hình chiếu:



- a. Othergraphic (trực giao)
 - b. Trimetric
 - c. Dimetric
 - d. Isometric
7. Hình sau là hình chiếu:

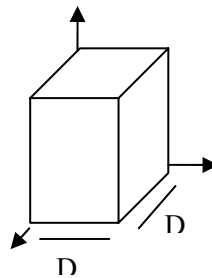


- a. Othergraphic (trực giao)
 - b. Trimetric
 - c. Dimetric
 - d. Isometric
8. Hình sau là phép chiếu:



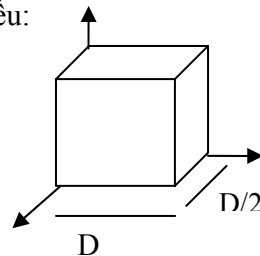
- a. Phối cảnh một tâm chiếu
- b. Phối cảnh hai tâm chiếu
- c. Cabinet
- d. Cavalier

9. Hình sau là phép chiếu:



- a. Phối cảnh một tâm chiếu
- b. Phối cảnh hai tâm chiếu
- c. Cabinet
- d. Cavalier

10. Hình sau là phép chiếu:



- a. Phối cảnh một tâm chiếu
- b. Phối cảnh hai tâm chiếu
- c. Cabinet
- d. Cavalier

11. Cho đoạn thẳng AB có tọa độ là A(2,6,1) và B(-1,2,-3) phép chiếu phối cảnh một tâm chiếu P(0,0,5) đoạn thẳng AB ta thu được tọa độ mới của AB là:

- a. (-1.5, 5.5) và (-0.25, 2.125)
- b. (2.5, 7.5) và (-0.625, 1.25)
- c. (3.5, -7.5) và (-0.65, 1.5)
- d. (-4.5, 1.25) và (0.25, 2.5)

12. Cho đoạn thẳng AB có tọa độ là A(1,4,-2) và B(3,-1,7) phép chiếu phối cảnh hai tâm chiếu M(10,0,0) và N(0,-5,0) đoạn thẳng AB ta thu được tọa độ mới của AB là:

a. $\left(\frac{17}{10}, \frac{17}{40}\right)$ và (3,-1)

b. $\left(\frac{12}{15}, \frac{43}{15}\right)$ và (-2,6)

c. $\left(\frac{10}{17}, \frac{40}{17}\right)$ và (5,2)

d. $\left(\frac{10}{17}, \frac{40}{17}\right)$ và (6,-2)

13. Cho đoạn thẳng AB có tọa độ là A(2,-5,3) và B(-1,4,0) phép chiếu phối cảnh ba tâm chiếu M(4,0,0), N(0,12,0) và P(0,0,-10) đoạn thẳng AB ta thu được tọa độ mới của AB là:

a. $\left(\frac{10}{9}, -\frac{25}{9}\right)$ $\left(-\frac{20}{9}, \frac{80}{9}\right)$

b. $\left(\frac{9}{10}, -\frac{9}{25}\right)$ $\left(-\frac{9}{20}, \frac{9}{80}\right)$

c. $\left(\frac{1}{9}, \frac{24}{9}\right)$ $\left(-\frac{22}{9}, -\frac{70}{9}\right)$

d. $\left(\frac{10}{11}, -\frac{25}{11}\right)$ $\left(-\frac{20}{11}, \frac{80}{11}\right)$

14. Đoạn mã sau là thuộc bài “...quay một đối tượng quanh trục tọa độ...” phép quay trong 3D. Hãy cho biết trục mà đối tượng quay quanh:

```
#define RADS 0.017453293// đổi độ ra radian
struct point{
    int x,y,z;
}
point quay(int &x, int &y, int &z, int goc , int chieu)
{
    point p;
    if(chieu==1)
    {
        p.x = x*cos(RADS*goc) + z*sin(RADS*goc);
        p.z = -x*sin(RADS*goc) + z * cos(RADS*goc);
        p.y = y;
    }
    .....
    return p;
}
```

- a. ox
b. oy
c. oz
d. Cả ba đều sai

15. Đoạn mã sau là thuộc bài “...quay một đối tượng quanh trục tọa độ...” phép quay trong 3D. Hãy cho biết trục mà đối tượng quay quanh:

```
#define RADS 0.017453293// đổi độ ra radian
struct point{
    int x,y,z;
}
point quay(int &x, int &y, int &z, int goc , int chieu)
{
    point p;
    //.....
    if(chieu==2)
    {
        p.y = y*cos(RADS*goc) - z*sin(RADS*goc);
        p.z = y*sin(RADS*goc) + z * cos(RADS*goc);
        p.x =x;
    }
    //.....
    return p;
}
```

- a. ox
- b. oy
- c. oz
- d. Cả ba đều sai

16. Đoạn mã sau là thuộc bài “...quay một đối tượng quanh trục tọa độ...” phép quay trong 3D. Hãy cho biết trục mà đối tượng quay quanh:

```
#define RADS 0.017453293// đổi độ ra radian
struct point{
    int x,y,z;
}
point quay(int &x, int &y, int &z, int goc , int chieu)
{
    point p;
    //.....
    if(chieu==3)
    {
        p.x = x*cos(RADS*goc)-y*sin(RADS*goc);
        p.y = x*sin(RADS*goc)+y*cos(RADS*goc);
        p.z = z;
    }
    return p;
}
```

- a. ox
- b. oy
- c. oz
- d. Cả ba đều sai

CHƯƠNG 6: MÀU SẮC TRONG ĐỒ HỌA

1. ÁNH SÁNG VÀ MÀU SẮC (light and color)

1.1. Quan niệm về ánh sáng

- Ánh sáng đem đến sự sống cho con người
- Ánh sáng đem đến màu sắc cho con người

Màu sắc là cảm giác mà nó xảy ra khi có năng lượng của ánh sáng, xuất hiện trên võng mạc và nhận biết được nhờ não.

- Hạnh phúc của con người là cảm nhận được màu sắc
- Nguyên tắc của ánh sáng dựa trên hai góc độ
 - Vật lý - physics
 - Sinh lý - physiology

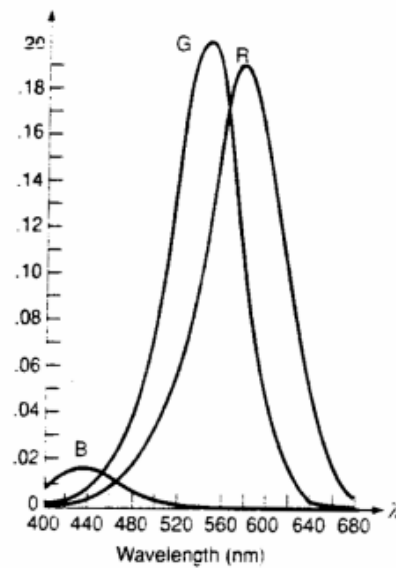
1.2. Yếu tố vật lý

Ánh sáng phụ thuộc vào mức năng lượng được truyền hay bước sóng của ánh sáng. Ánh sáng trắng hay dải sóng mà mắt người có thể cảm nhận được, sau khi phân tích qua lăng kính thành các phổ màu: tím, chàm, lam, lục, vàng, da cam, đỏ....Ánh sáng là sóng điện từ có bước sóng λ đi từ 400nm – 700nm.

Frequency $f(\text{hz})$	Color	Wavelength $\lambda(\text{nm})$
7×10^{14}	Violet	400
6×10^{14}	Blue	
	Green	500
5×10^{14}	Yellow	
	Orange	600
4×10^{14}	Red	700

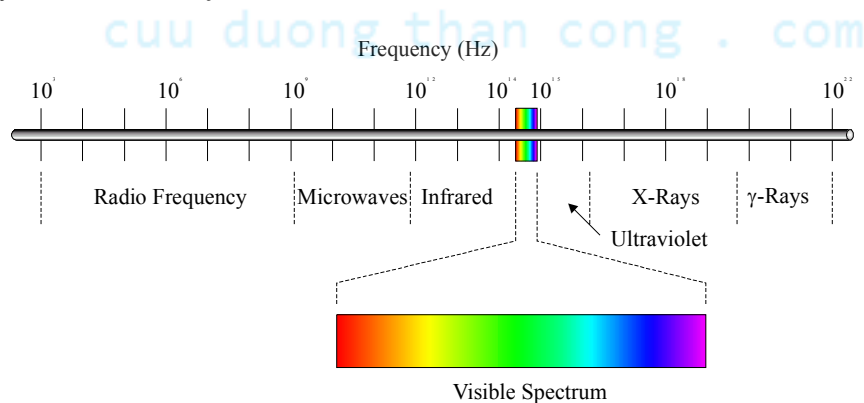
Hình 6.1 Tần số, màu sắc và bước sóng của ánh sáng nhìn thấy

Tổng năng lượng đặc trưng cho từng loại bước sóng được biểu diễn bằng hàm phân bố năng lượng phổ $P(\lambda)$.



Hình 6.2 Đồ thị phân bố ba màu

Nguyên lý pha màu với các sắc màu cơ bản là đỏ, lục, lam (Red, Green, Blue). Theo nguyên lý ba màu này, một màu bất kỳ đều có thể được tạo ra từ ba màu cơ bản.



Hình 6.3 Vùng ánh sáng thấy được

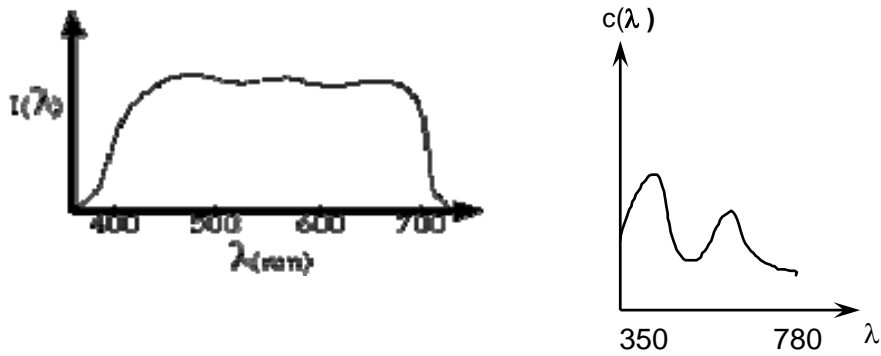
Phổ của ánh sáng (Spectrum)

- Ánh sáng xuất phát từ nguồn sáng được xác định bởi phổ $I(\lambda)$ của nó - *spectrum*, phổ $I(\lambda)$ này được đo bởi năng lượng của ánh sáng với bước sóng cho trước đi qua một đơn vị diện tích trong một khoảng thời gian.
- Thuật ngữ khác phổ công suất - *power spectrum*, với đơn vị là watts/m^2 .
- Phổ công suất được dùng để đo cường độ phát sáng của nguồn - *emission intensity*
- Hay còn gọi cường độ truyền dẫn - *transmission intensity* của ánh sáng theo luồng trong không gian, hay cường độ phát sáng- *illumination intensity* của ánh sáng đập lên bề mặt.

Màu sắc

- Isaac Newton - ánh sáng trắng đi qua thấu kính thủy tinh sẽ phát tán ra thành phổ các màu cầu vồng

- Ngược lại, thấu kính có thể kết hợp các phổ ánh sáng để tạo thành ánh sáng trắng.
- Chùm sáng khi phân tách thành phổ màu có liên quan đến phổ năng lượng $I(\lambda)$.
- Phổ điện từ đó có bước sóng từ 350 tới 780 nm và màu được đặc trưng bởi $c(\lambda)$



Hình 6.4 Phổ điện từ của ánh sáng

1.3. Cảm nhận màu sắc của con người (Physiology - Sinh lý - Human Vision)

Cấu tạo hệ quan sát của con người gồm 2 loại tế bào cảm thụ - sensors

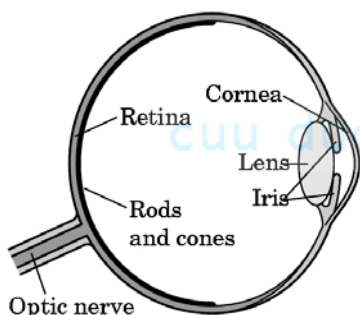
- Rods (tế bào que): cho cảm nhận cường độ ánh sáng thấp hay trong bóng tối
- Cones - tế bào hình nón

Nhạy cảm với ánh sáng màu sắc

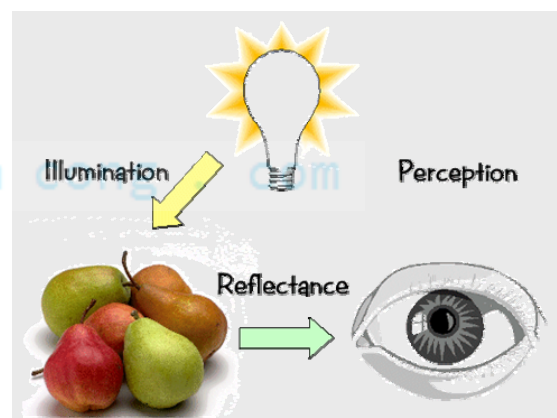
Chia làm 3 loại nón - cone

Ba loại sẽ có ba giá trị gọi là tristimulus values cảm nhận tương ứng trên 3 màu cơ bản và gửi đến não những tín hiệu tạo ra cảm nhận về màu sắc S-M-L.

Để đạt được một sự cảm nhận về một màu bất kỳ ta phải xác định giá trị của 3 đại lượng này



Hình 6.5 Cấu tạo mắt con người

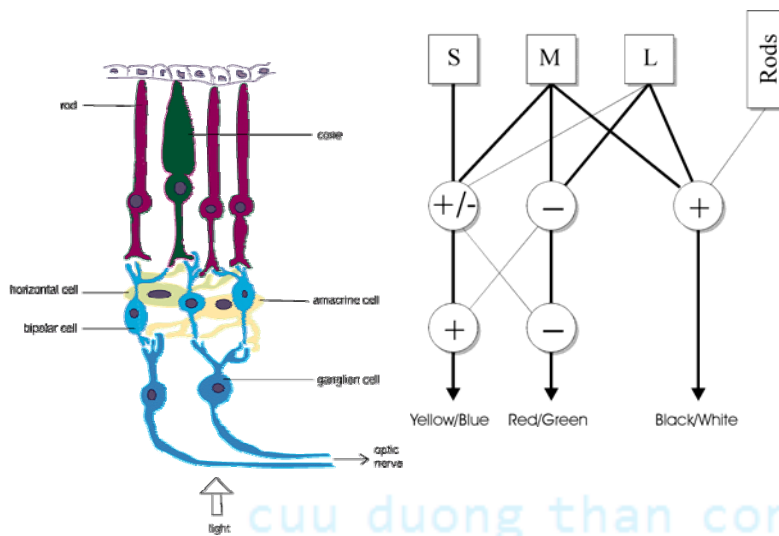


Hình 6.6 Con người cảm nhận màu sắc

Ba loại tế bào nón sẽ có độ nhạy cảm với 3 màu và các bước sóng khác nhau như:

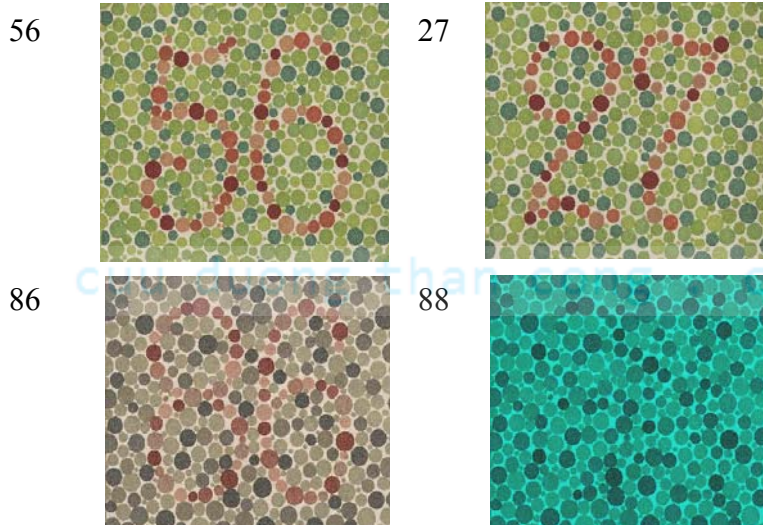
- L or R, hầu như nhạy cảm với ánh sáng đỏ (610 nm)
- M or G, nhạy cảm với ánh sáng lục (560 nm)
- S or B, nhạy cảm với ánh sáng lam (430 nm)
- Vậy ta có người bị mù màu chẳng qua là mất tế bào nón

S:M:L tỷ lệ = 1:20:40 \Rightarrow từ đó ta thấy con người nhạy cảm với màu đỏ hơn là màu xanh lam.



- Nó không chỉ đơn giản là RGB cộng với ánh sáng
- Kết hợp tế bào que và nón mang lại cảm nhận cả màu sắc và ánh sáng
- Tế bào đáp ứng thay đổi với cường độ:
 - Khi ánh sáng yếu: thích ứng với nhìn tối, tế bào que trội hơn cảm nhận màu sắc không đáng kể
 - Khi ánh sáng là trung bình: thì cả hai là mức trung bình
 - Ánh sáng cao: xử lý màu sắc, tế bào nón trội hơn

Hình 6.7 Cấu tạo và nguyên lý hoạt động các tế bào mắt



Hình 6.8 Cảm nhận màu sắc của con người

Ta thấy màu đỏ tươi (bão hoà) khác màu đỏ tái (chưa bão hoà).

Yếu tố cảm nhận sinh lý:

- Hue - sắc màu: dùng để phân biệt sự khác nhau giữa các màu như xanh, đỏ, vàng.....
- Saturation - độ bão hoà: chỉ ra mức độ thuần của một màu hay khoảng cách của màu tới điểm có cường độ cân bằng.
- Lightness - độ sáng: hiện thân về mô tả cường độ sáng từ ánh sáng phản xạ nhận được từ đối tượng.
- Brightness - độ phát sáng: cường độ ánh sáng tự đối tượng phát ra chứ không phải do phản xạ từ các nguồn sáng khác.
- Hệ thống màu được sử dụng rộng rãi đầu tiên do A.H.Munsell đưa ra vào những năm 1976 không gian ba chiều bao gồm ba yếu tố Hue, Lightness và Saturation.

Sắc màu trong hội họa: thường được xác định mẫu trên góc độ sắc thái (tints), sắc độ (shade), tông màu (tone) từ các màu nguyên chất hay bão hoà. Sắc thái là được hình thành từ việc thêm sắc tố trắng vào các màu nguyên chất để giảm độ bão hoà. Sắc độ hay còn gọi là độ giảm màu được tạo ra bằng cách thêm màu đen vào các màu nguyên chất để giảm đi độ sáng của màu. Còn tông màu là kết quả của cả hai quá trình trên khi thêm cả màu trắng lẫn màu đen vào các màu nguyên chất.

1.4. Các đặc trưng cơ bản của ánh sáng

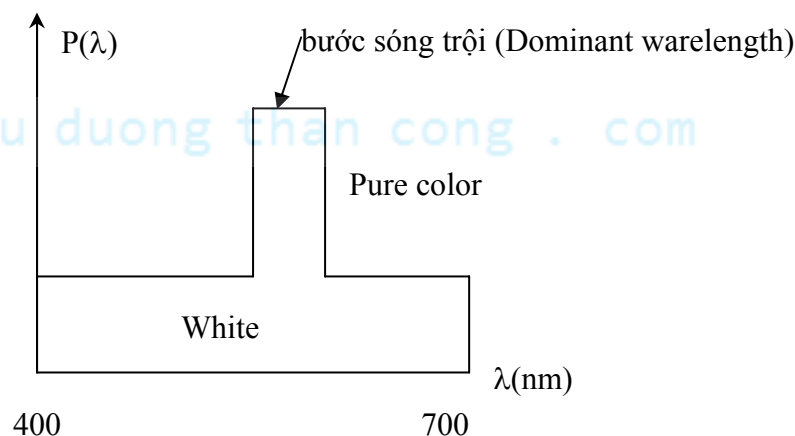
Ánh sáng có thể được mô tả bằng ba thuật ngữ:

- Độ sáng (Lightness): dựa vào tính chất vật lý của nó, hay còn gọi là tính phát sáng (brightness). Tính phát sáng đo lường năng lượng toàn phần trong ánh sáng. Nó tỷ lệ với diện tích giới hạn bởi $P(\lambda)$ và trục λ trong dãy 400 đến 700 nm. Diện tích này được tính như sau:

$$\int_{\lambda} P(\lambda) d\lambda$$

Tính phát sáng càng cao, thì độ sáng càng sáng hơn đối với người quan sát.

- Sắc độ (shade): để phân biệt ánh sáng trắng với ánh sáng đỏ với ánh sáng xanh. Đối với ánh sáng có sự phân bố quang phổ lý tưởng như hình dưới, sắc độ thích ứng với một tính chất vật lý khác được gọi là bước sóng trội của sự phân bố.



Hình 6.9 Phân bố quang phổ của ánh sáng

- Độ bão hoà (Saturation): mô tả mức độ chói lọi của ánh sáng. Ví dụ hai ánh sáng màu đỏ có thể khác nhau ở tính phát sáng/độ sáng và chúng có thể khác nhau ở mức độ chói lọi (ví dụ màu đỏ tươi/bão hoà khác với màu đỏ tái/ không bão hoà). Chúng ta có:

màu tươi

Độ bão hoà = _____

Màu tươi + màu trắng

2. ÁNH SÁNG ĐƠN SẮC

Không cảm nhận được các sắc màu khác như vàng, đỏ, tím... khi quan sát trên màn hình đen trắng

Định lượng là thuộc tính duy nhất của các tia sáng đơn sắc và về mặt vật lý nó được tính bằng năng lượng của tia sáng được mô tả cường độ (intensity) hay độ chiếu sáng (luminance).

Dưới góc độ cảm nhận về mặt tâm lý thì cường độ của tia sáng chính là độ sáng của vật (brighness)

Sử dụng phổ kế - photometer để đo độ sáng thấp nhất (min) và cao nhất (max) của màn hình. Và đó là khoảng động.

Khoảng cường độ nhận giá trị min là I_0 , đến max là 1.0. Làm thế nào để thể hiện được 256 mức xám khác nhau?

2.1. Cường độ sáng và cách tính

Cường độ của nguồn sáng sẽ thay đổi trong khoảng từ 0 đến 1: 0 qui ước cho màu đen và 1 cho màu trắng.

khoảng tăng của cường độ sáng sẽ phân chia theo hàm logarit

$$I_0 = I_0, I_1 = r I_0, I_2 = r I_1 = r^2 I_0, \dots, I_{255} = r^{255} I_0 = 1$$

$$r = (1/I_0)^{1/255}, I_j = r^j I_0 = I_0^{(255-j)/255}$$

$$I = k.N^\gamma$$

Với k và γ là các hằng số (có γ từ 2.2 -> 2.6), N số lượng hạt tại một thời điểm phát ra từ cathode trong một chùm tia điện tử.

$I_0 = I_0$
$I_1 = r I_0$
$I_2 = r I_1 = r^2 I_0$
...
$I_{255} = r I_{254} = r^{255} I_0 = 1$

2.2. Phép hiệu chỉnh gama

Ta có $I = K.V^\gamma$ hay $V = (I/K)^{1/\gamma}$

Trong đó V điện áp tỉ lệ với N trên mỗi điểm ảnh.

Giả sử chúng ta có một cường độ sáng I thì bước đầu tiên ta phải làm là tìm ra giá trị I_j gần nhất qua phép làm tròn. Giá trị j tìm được $I = r^j I_0$ vậy $r^j = I/I_0$ suy ra $j = \text{ROUND}(\log_r(I/I_0))$.

Thay j vào công thức ta có:

$$I_j = r^j \cdot I_0$$

Bước tiếp theo của tiến trình là xây dựng mức điện áp V_j cho điểm ảnh mà cường độ ánh sáng có giá trị tương ứng là I_j .

$$V_j = \text{ROUND}(I_j / K)^{1/\gamma}$$

Ta thấy để cường độ sáng là như nhau cho các màn hình (hay ảnh là như nhau), thì chỉ còn thay đổi giá trị gamma. Giá trị gamma là số mũ của hàm lũy thừa, giá trị đó đối với loại phim nhựa 35mm trong phòng tối là 1.5. Nhưng hệ số gamma của CRT là loại thiết bị độ sáng phụ thuộc vào ống phóng tia điện tử. Thực tế giá trị gamma của CRT dao động từ 2.3 đến 2.6.

Ta có sự phản hồi tuyến tính của CRT có thể được bù bởi phần cứng và phép bù này gọi là phép hiệu chỉnh gamma (Gamma correction). Việc sử dụng I_j làm chỉ số trong Lookup table (LUT) để tìm ra cường độ sáng cho các điểm ảnh trên màn hình gọi là phép hiệu chỉnh gamma với bảng LUT. Vậy bao nhiêu khoảng sẽ là đủ nhỏ cho việc thể hiện một điểm ảnh đen trắng là liên tục? Theo tính toán thì $r=1.01$ là mức ngưỡng phân biệt của mắt. Nếu $r < 1.01$ thì mắt sẽ không phân biệt được sự khác lệnh giữa hai cường độ lân cận nhau I_j và I_{j+1} .

$r = (1/I_0)^{1/n} = 1.01$ vậy $n = \log_{1.01}(1/I_0)$ với $(1/I_0)$ là khoảng biến động của thiết bị phần cứng.

2.3. Xấp xỉ bán tông - halftone

Sử dụng đen trắng để mô tả ảnh nhiều màu ?

Phương pháp trên dựa vào cấu tạo mắt của người cũng như nguyên lý thu nhận ảnh của mắt khi nhìn những vùng nhỏ ở khoảng cách xa. Lúc đó mắt không thể phân biệt được các vật một cách cụ thể mà chỉ ghi nhận cường độ trung bình của vùng ảnh đó. Phương pháp này được gọi là xấp xỉ bán tông.

Phương pháp này cho phép đạt được độ phân giải trong in ảnh báo vào khoảng từ 60->80 dpi, còn trong tạp chí và sách cao hơn là khoảng từ 110 -> 120 dpi.

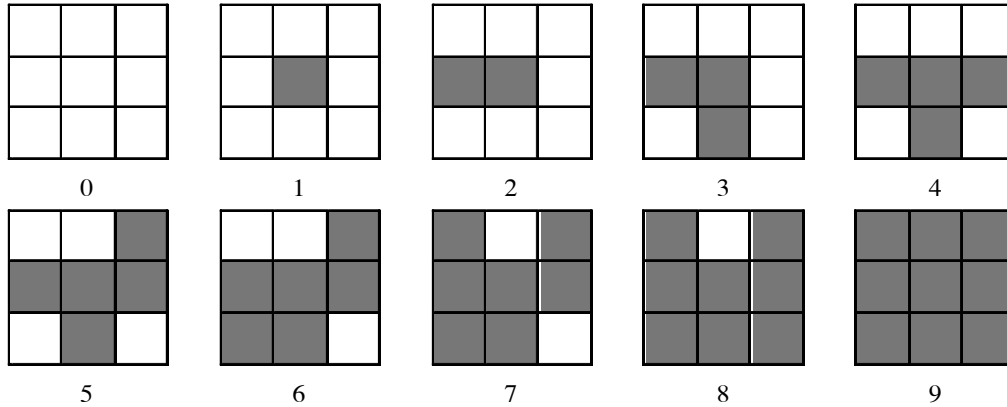


Hình 6.10 Dùng đen trắng để thể hiện ảnh màu

Ta có giải thuật phân ngưỡng (Threshold). Phân ngưỡng là lấy một giá trị trung bình của cả vùng ảnh làm ngưỡng và so sánh nó với mức sáng của từng điểm ảnh trong ô. Nếu giá trị của nó lớn hơn ngưỡng thì điểm đó được bật (on), nếu ngược lại thì tắt (off).

Ta thấy với phương pháp này mất đi nhiều thông tin của ảnh gây ra một số hiệu ứng phụ cho ảnh. Để giải quyết ta dùng phương pháp sau:

Mẫu tô: ta biểu diễn một điểm ảnh trên màn hình theo các mẫu tô. Đơn vị nhỏ nhất của ảnh lưới là 2×2 ta có 5 mức độ để thể hiện cường độ sáng của vùng đơn vị. Ma trận lưới kích thước $n \times n$ chúng ta có $n^2 + 1$ độ phân giải khác nhau. Hình dưới đây là ma trận 3×3 và các đơn vị mã là 0 đến 9.

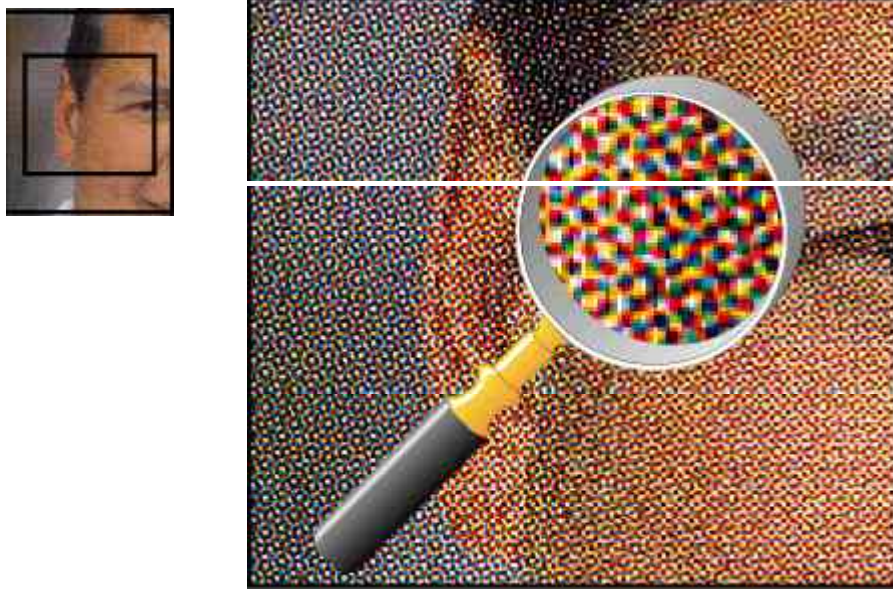


Hình 6.11 Phân bố các điểm trong vùng theo thứ tự tăng dần

Việc thể hiện cường độ vùng ảnh I bây giờ chỉ còn đơn thuần là bật tắt cả các vị trí $< I$

- Thứ nhất: Không dùng ma trận mẫu có dạng đường thẳng ngang
- Thứ hai: Các mẫu phải được hình thành theo chuỗi các bước liên tiếp nhau sao cho mọi điểm ảnh có mật độ thể hiện ngưỡng a đều phải có mặt để thể hiện mọi ngưỡng b với $b > a$.
- Thứ ba: Các mẫu phải được phát triển theo quy tắc từ tâm đi dần ra xung quanh. Nhờ đó sẽ gây được cho người sử dụng hiệu ứng tăng kích thước điểm.
- Thứ tư: Với một số các thiết bị in như máy in laser hay các thiết bị ghi hình, vấn đề về các điểm độc lập tuyệt đối là rất khó có khả năng đạt được. Khi mà đại đa phần các điểm ảnh được bật cho một cường độ sáng thì chúng sẽ gây ra các thay đổi cho các điểm còn lại.

Xấp xỉ bán tông với ảnh màu: Ta lấy mỗi cell không phải là một đơn vị nữa mà là ba đơn vị nhỏ đặc trưng cho ba màu (Red, Green, Blue).



Hình 6.12 Màu sắc trong ảnh màu

2.4. Ma trận Dither và phép lấy xấp xỉ bán tông

Bayer năm 1973 đã đưa ra dạng ma trận dither mà nhờ đó tăng được độ mịn của ảnh khi hiển thị.

Ma trận 2×2 ma trận dither có ký hiệu $D^{(2)}$:

Tính các ma trận $D^{(2n)}$ thông qua $D^{(n)}$:

$$D^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} \quad D^{(n)} = \begin{bmatrix} 4D^{(n/2)} + D_{00}^{(2)}U^{(n/2)} & 4D^{(n/2)} + D_{01}^{(2)}U^{(n/2)} \\ 4D^{(n/2)} + D_{10}^{(2)}U^{(n/2)} & 4D^{(n/2)} + D_{11}^{(2)}U^{(n/2)} \end{bmatrix}$$

$U^{(n)}$ là ma trận $n \times n$ với tất cả các phần tử = 1

Với $n = 4$ và kết quả từ $D^{(2)}$

$$D^{(4)} = \begin{bmatrix} 4 \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} + 0 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} & 4 \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} + 2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \\ 4 \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} + 3 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} & 4 \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} + 1 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$

Để xác định điểm (x,y) là bật hay tắt, ta cần xác định vị trí điểm tương ứng với vị trí ma trận Dither để so sánh cường độ sáng trung bình S với giá trị đó trong ma trận. Nếu $S > D_{ij}$ thì bật.

3. CÁC HỆ MÀU TRONG MÀN HÌNH ĐỒ HỌA

Mô hình màu - color model: là hệ thống có quy tắc cho việc tạo khoảng màu từ tập các màu cơ bản.

Có 2 loại mô hình màu là:

- Màu thêm (additive): mô hình màu thêm sử dụng ánh sáng - light để hiển thị màu. Màu sắc của mô hình này là kết quả của ánh sáng truyền dẫn - transmitted
- Màu trừ (subtractive): mô hình màu trừ sử dụng mực in - printing inks. Màu sắc cảm nhận được là từ ánh sáng phản xạ - reflected light (lấy màu trời).

Khoảng màu mà chúng ta tạo ra với tập các màu cơ bản gọi là gam màu hệ thống (system's color gamut)

Mỗi mô hình màu có khoảng màu hay gam màu riêng gamut (range) của những màu mà nó có thể hiển thị hay in.

Mỗi mô hình màu được giới hạn khoảng của phổ màu nhìn được. Gam màu hay khoảng còn được gọi là không gian màu "color space". Ảnh hay đồ họa vector có thể nói: sử dụng không gian màu RGB hay CMY hay bất cứ không gian màu nào khác.

Một số ứng dụng đồ họa cho phép người dùng sử dụng nhiều mô hình màu đồng thời để soạn thảo hay thể hiện đối tượng hình học. Điểm quan trọng là hiểu và để cho đúng mô hình cần thiết cho công việc.

Có ba hệ màu định hướng phân cứng:

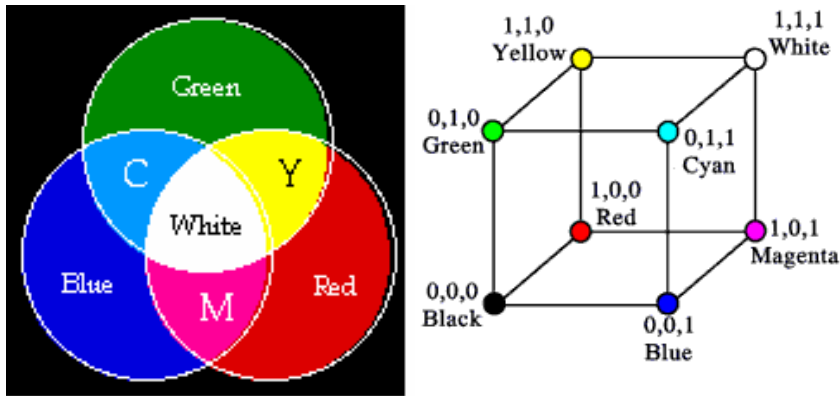
- RGB (Red, Green, Blue) dùng với CRT
- YIQ trong hệ thống tivi màu băng tần rộng
- CMY (Cyan, Magenta, Yellow) sử dụng một số thiết bị in màu

Không có một mô hình màu nào trong các mô hình thực tế trên có tính dễ sử dụng, vì chúng không có mối liên hệ trực tiếp với ý niệm màu trực giác của con người. Màu mà con người cảm nhận: Hue (sắc màu), Saturation (độ bão hoà), Lightness (độ sáng).

Các mô hình màu khác nhau được phát triển nhằm sử dụng cho một tiêu chí nhất định.

3.1. Mô hình màu RGB (Red, Green, Blue - đỏ, lục, lam)

Gam màu thể hiện trong màn hình CRT xác định bằng những đặc tính của hiện tượng phát quang các chất phát pho trong màn hình CRT. Mô hình không gian màu RGB được sắp xếp theo khối lập phương đơn vị. Đường chéo chính của khối lập phương với sự cân bằng về số lượng từng màu gốc tương ứng với các mức độ xám với đen là (0,0,0) và trắng (1,1,1).



Hình 6.13 Mô hình không gian màu RGB

$$C = rR + gG + bB$$

Trong đó C = màu hoặc ánh sáng kết quả. (r,g,b) = tọa độ màu trong miền $[0, 1]$, (R,G,B) = các màu cơ bản đỏ, lục và lam.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Nếu hai màu tạo ra cùng một giá trị kích thích thì chúng ta không thể phân biệt được hai màu. Không gian màu RGB dựa theo chuẩn ITU-R BT.709, với $\gamma = 2.2$ và điểm trắng của mô hình là 6500 degrees K.

3.2. Mô hình màu CMY (Cyan, Magenta, Yellow - xanh tím, Đỏ tươi, vàng)

Đây là mô hình màu bù (*Subtractive color models*) hiển thị ánh sáng và màu sắc phản xạ từ mực in. Bỏ xung thêm mực đồng nghĩa với ánh sáng phản xạ càng ít.

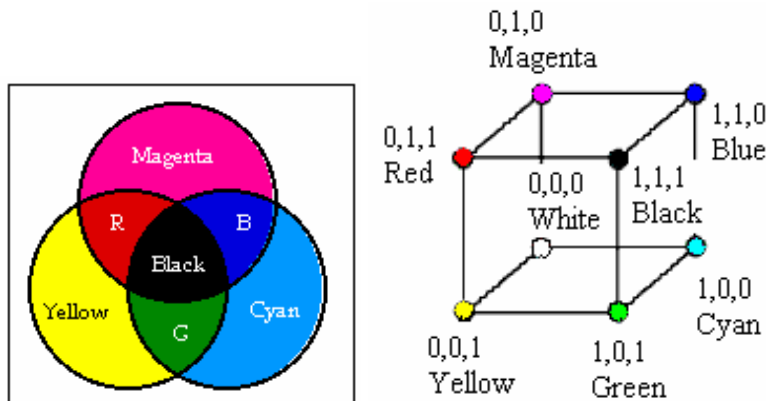
Khi bề mặt không phủ mực thì ánh sáng phản xạ là ánh sáng trắng - white.

Khi 3 màu có cùng giá trị cho ra màu xám. Khi các giá trị đạt max cho màu đen.

$$\text{Color} = cC + mM + yY$$

Ta có Red +Cyan = Black ; Green +Magenta = Black ; Blue + Yellow = Black

$$\text{Black} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



Hình 6.14 Mô hình không gian màu CMY

Mô hình màu CMY- K

Mô hình mở rộng của CMY ứng dụng trong máy in màu. Giá trị đen bổ xung vào thay thế cho hàm lượng màu bằng nhau của 3 màu cơ bản.

Công thức chuyển đổi:

$$K = \min(C, M, Y) ;$$

$$C = C - K ;$$

$$M = M - K ;$$

$$Y = Y - K ;$$

C-Cyan, M-Magenta, Y-Yellow; K-black

3.3. Mô hình màu YIQ

Mô hình màu YIQ là mô hình màu được ứng dụng trong truyền hình màu băng tần rộng tại Mỹ, và do đó nó có mối quan hệ chặt chẽ với màn hình đồ họa màu raster. YIQ là sự thay đổi của RGB cho khả năng truyền phát và tính tương thích với ti vi đen trắng thế hệ trước. Tín hiệu truyền sử dụng trong hệ thống NTSC (National Television System Committee).

Sự biến đổi RGB thành YIQ được xác định theo công thức sau:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Y độ chói, I & Q đại lượng về màu sắc

Chú ý: Y giống như Y trong mô hình CIE's

Nó hoàn toàn tương thích với đen/trắng (B/W) của TV

Những đại lượng trong ma trận biến đổi được tìm bằng cách sử dụng các phosphor NTSC RGB chuẩn có các tọa độ sắc phổ là R(0.67 0.33), G (0.21 0.71) và B(0.14 0.08). Người ta cũng giả định rằng điểm trắng nằm ở $x_w = 0.31$, $y_w = 0.316$ và $Y_w = 1.0$.

3.4. Mô hình màu HSV (Hue, Saturation, Value) - Mỹ thuật

Yếu tố cảm nhận màu sắc:

- Hue - sắc màu: dùng để phân biệt sự khác nhau giữa các màu như xanh, đỏ, vàng...

- Saturation - độ bão hoà: chỉ ra mức độ thuần của một màu hay khoảng cách của màu tới điểm có cường độ cân bằng(màu xám)
- Lightness - độ sáng: hiện thân về mô tả cường độ sáng từ ánh sáng phản xạ nhận được từ đối tượng.
- Brightness - độ phát sáng: cường độ ánh sáng mà tự đối tượng phát ra chứ không phải do phản xạ từ các nguồn sáng khác.

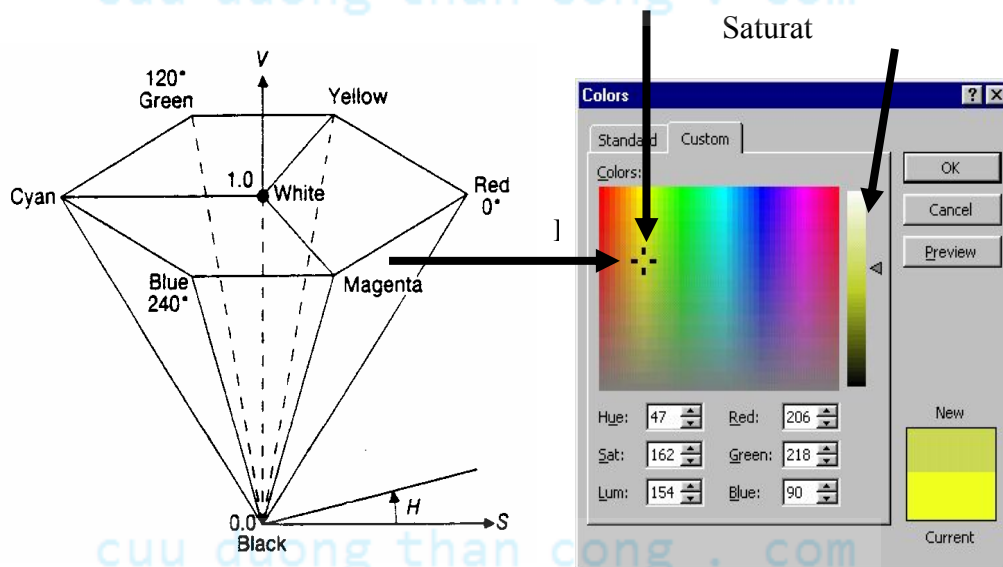
Mô hình màu RGB, CMY, YIQ được định hướng cho phần cứng

HSV (Hue, Saturation, Value)=HSB(Hue, Saturation, Brightness) định hướng người sử dụng dựa trên cơ sở về trực giác về tông màu, sắc độ và sắc thái mỹ thuật

Mô hình màu HSV được Alvey Ray Smith đưa ra 1978. Hue: màu sắc 0° - 360° đo bởi góc quay xung quanh trục đứng với màu đỏ là 0° , màu lục là 120° , màu lam là 240° . Các màu bổ sung cho hình chóp ở 180° đối diện với màu khác.

Value-Brightness:(độ sáng) 0-1 đường cao V với đỉnh là các điểm gốc tọa độ (0,0). Điểm ở đỉnh là màu đen và giá trị $V=0$, tại các điểm này giá trị của H và S không liên quan đến nhau. Khi điểm có $S=0$ và $V=1$ là điểm màu trắng, những giá trị trung gian của V đối với $S=0$ (trên đường thẳng qua tâm) là các màu xám. Khi $S=0$ giá trị của H phụ thuộc được gọi bởi các qui ước không xác định. Ngược lại khi S khác 0 giá trị H sẽ là phụ thuộc.

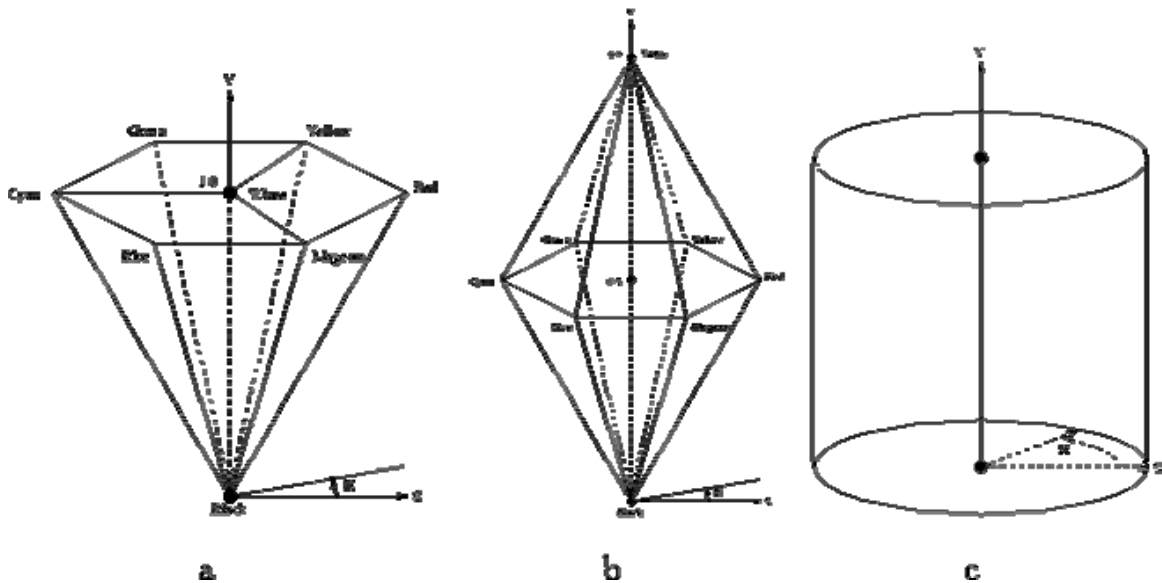
Saturation: Độ bão hoà 0-1, giá trị của S là tập các giá trị từ 0 trên đường trục tâm (trục V) đến 1 trên các mặt bên tại đỉnh của chóp 6 cạnh.



Hình 6.15 Mô hình màu HSV

Mô hình màu HLS (Hue, Lightness, Saturation Model) – không gian màu trực quan

Mô hình thường được sử dụng trong kỹ thuật đồ họa. Ưu điểm là rất trực giác ví dụ ta có thể chọn màu, thay đổi độ sáng và thay đổi độ bão hoà. Nhược điểm là khi chuyển đổi với không gian màu RGB sẽ có sai số (cube stood on end) thay đổi trên các loại màn hình khác nhau, rõ ràng không cảm nhận đều các màu.



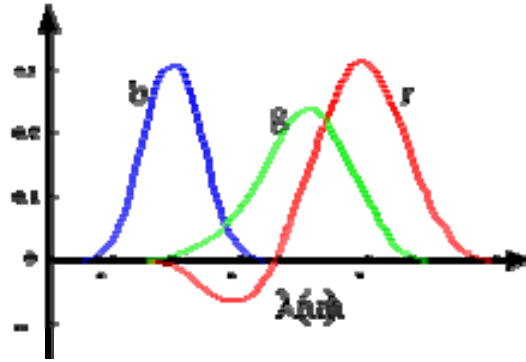
Hình 6.16 Mô hình màu hình chóp sáu cạnh đôi HLS

Chúng ta có thể coi mô hình HLS như một sự biến dạng của mô hình HLS mà trong đó mô hình này màu trắng được kéo hướng lên hình chóp sáu cạnh phía trên từ mặt $V=1$. Như với mô hình chóp sáu cạnh đơn, phần bổ sung của màu sắc được đặt ở vị trí 180° hơn là xung quanh hình chóp sáu cạnh đôi, sự bão hoà được đo xung quanh trục đứng, từ không trên trục tới 1 trên bề mặt. Độ sáng (Lightness)=0 cho màu đen (tại điểm mút thấp nhất của hình chóp sáu cạnh đôi) và bằng 1 cho màu trắng (tại đầu mút cao nhất).

3.5. Biểu đồ màu CIE (1931 – Commission Internationale de l'Eclairage)

Nhược điểm của RGB:

- Kết quả thực nghiệm cho thấy rất nhiều những ánh sáng mẫu không thể tạo thành từ 3 thành phần màu cơ sở với nguyên nhân do vỏ của võng mạc - retinal cortex.
- Với màu Cyan: cường độ của ánh sáng 2 màu green và blue kích thích cảm nhận màu đỏ trong mắt ngăn không cho thu được màu chính xác
- Cách duy nhất để thu được màu này là loại bớt phần màu đỏ bằng cách thêm ánh sáng đỏ vào mẫu ban đầu.
- Bằng cách thêm từ từ ánh sáng đỏ vào thu được (test + red) sẽ cho ra màu đúng bằng (blue + green)
- $C + rR = gG + bB \Leftrightarrow C = gG + bB - rR$
- Vấn đề đặt ra là việc phức tạp trong phân tích màu và chuyển đổi màu với đại lượng âm của ánh sáng đỏ độc lập thiết bị.



Hình 6.17 Hàm phân bố ba màu cơ sở (qua thực nghiệm - phụ thuộc vào mắt người)

CIE stands for *Comission Internationale de l'Eclairage* (International Commission on Illumination)

Commission thành lập 1913 tạo một diễn đàn quốc tế về trao đổi ý tưởng và thông tin cũng như tập chuẩn - set standards cho những vấn đề liên quan đến ánh sáng.

Mô hình màu CIE color phát triển trên cơ sở hoàn toàn độc lập thiết bị

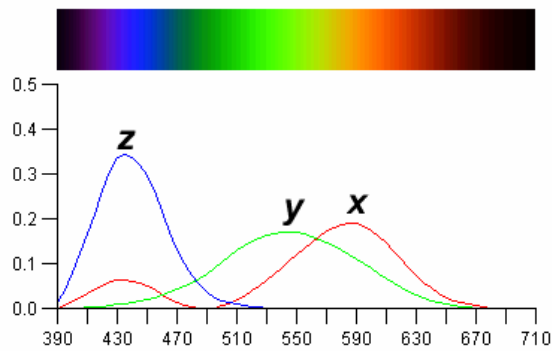
Dựa trên sự cảm nhận của của mắt người về màu sắc.

Yếu tố cơ bản của mô hình CIE định nghĩa trên chuẩn về nguồn sáng và chuẩn về người quan sát.

CIE XYZ - Color Space

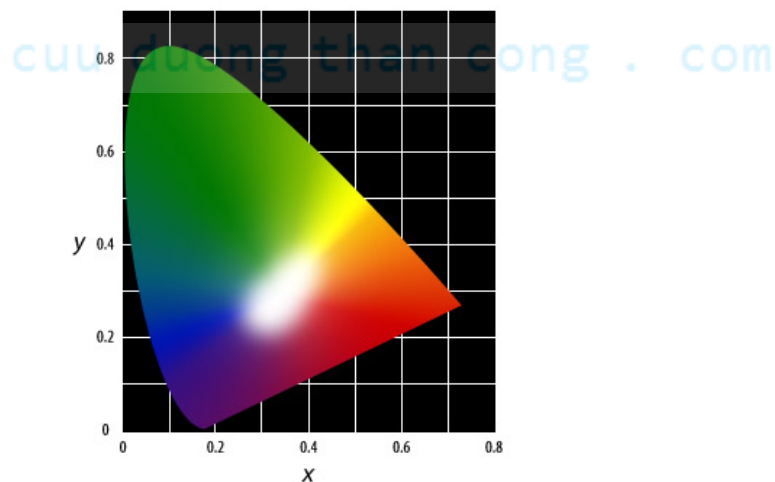
- CIE - Cambridge, England, 1931. Với ý tưởng 3 đại lượng ánh sáng - lights màu X, Y, Z cùng phổ tương ứng.
- Mỗi sóng ánh sáng λ có thể cảm nhận được bởi sự kết hợp của 3 đại lượng X,Y,Z
- Mô hình - là khối hình không gian 3D X,Y,Z gồm gam màu (gamut) của tất cả các màu có thể cảm nhận được.
- $\text{Color} = X'X + Y'Y + Z'Z$
- Các giá trị XYZ thay thế cho 3 đại lượng truyền thống RGB
- Màu được hiểu trên 2 thuật ngữ (Munsell's terms): màu sắc và sắc độ
- Ưu điểm của 3 loại màu nguyên lý cơ bản là có thể sinh ra các màu trên cơ sở tổng các đại lượng dương của màu mới thành phần.
- Việc chuyển đổi từ không gian màu 3D tọa độ (X,Y,Z) vào không gian 2D xác định bởi tọa độ (x,y), theo công thức dưới phân số của của tổng 3 thành phần cơ bản.
- $x = X/(X+Y+Z)$, $y = Y/(X+Y+Z)$, $z = Z/(X+Y+Z)$

Có: $x + y + z = 1$, ở đây tọa độ z không được sử dụng



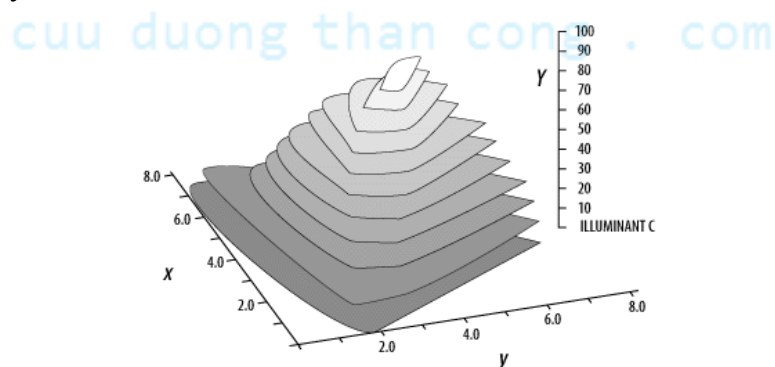
Hình 6.18 Hàm phân bố của các đại lượng CIE cơ sở

- Chuẩn CIE xác định 3 màu giả thuyết *hypothetical colors*, X, Y, and Z làm cơ sở cho phép trộn màu theo mô hình 3 thành phần kích thích.
- Không gian màu hình móng ngựa - horseshoe-shaped là kết hợp của không gian tọa độ 2D màu x, y và độ sáng.
- $\lambda_x = 700 \text{ nm}$; $\lambda_y = 543.1 \text{ nm}$; $\lambda_z = 435.8 \text{ nm}$
- Thành phần độ sáng hay độ chói được chỉ định chính bằng giá trị đại lượng Y trong tam kích tổ của màu sắc



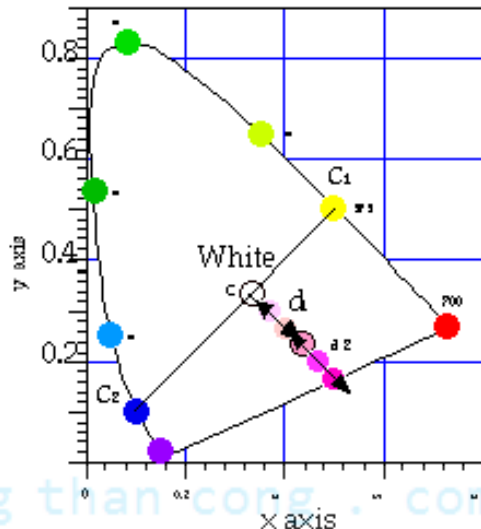
Hình 6.19 Không gian màu hình móng ngựa

Mô hình CIE xyY



Hình 6.20 Mô hình màu CIE xyY

- Thang đo của Y xuất phát từ điểm trắng trên đường thẳng vuông góc với mặt phẳng x,y với giá trị từ 0 tới 100.
- Khoảng màu lớn nhất khi Y=0 tại điểm trắng và bằng CIE Illuminant C. Đây là đáy của hình.
- Khi Y tăng màu trở nên sáng hơn và khoảng màu hay gam màu giảm diện tích trên tọa độ x,y cũng giảm theo.
- Tại điểm trên không gian với Y= 100 màu có sắc xám bạc và khoảng màu ở đây là bé nhất.



Hình 6.21 Không gian màu hình móng ngựa

4. CHUYỂN ĐỔI GIỮA CÁC HỆ MÀU

Việc xây dựng và thể hiện màu sắc của các đối tượng trên màn hình đồ họa chỉ thực hiện qua mô hình ba màu mà phần cứng hỗ trợ RGB. Vậy khi phần mềm ứng dụng có sử dụng đến các mô hình màu khác, ta phải chuyển đổi giữa chúng.

4.1. Chuyển đổi HSV - RGB

Hệ HSV có H (sắc màu) chạy từ 0^0 đến 360^0 với màu đỏ tại 0^0 .

S (độ bão hoà) và V (giá trị cường độ ánh sáng) thuộc khoảng $[0 \ 1]$

If (S==0)//H không tham gia - đen trắng

R = V;

G = V;

B = V;

Else // Khi đó $S < > 0$ trường hợp màu

// Màu của điểm ảnh được xác định thông qua 3 biến phụ M,N và K

if (H==360)

H=0;

Else

H = H/60;

I = (int)H; // lấy giá trị nguyên

F = H - I;

$$M = V*(1 - S);$$

$$N = V*(1 - S*F);$$

$$K = V*(1 - S*(1 - F));$$

if I == 0 then (R,G,B) = (V,K,M);

if I == 1 then (R, G, B) = (N, V, M);

if I == 2 then (R, G, B) = (M, V, K);

if I == 3 then (R, G, B) = (M, N, V);

if I == 4 then (R, G, B) = (K, M, V);

if I == 5 then (R, G, B) = (V, M, N);

4.2. Chuyển đổi RGB sang XYZ

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Các nhà sản xuất cung cấp các tọa độ XYZ cho phốt pho tương ứng với màu RGB mà màn hình hiển thị: (X_r, Y_r, Z_r) , (X_g, Y_g, Z_g) và (X_b, Y_b, Z_b)

Hay viết lại:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Có R=1

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \text{ suy ra } \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix}$$

Tương tự G=1

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ suy ra } \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix}$$

Tương tự B=1

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{ suy ra } \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix}$$

Thường $[X \ Y \ Z]$ cho mỗi phốt pho (phosphor) thì không được cung cấp, nên chúng ta tính với trường hợp điểm trắng (whitepoint) khi $R=G=B=1$.

Bây giờ chúng ta cần biết độ chói của điểm trắng được gửi bởi Y_w . Ta đặt:

$$E_r = X_r + Y_r + Z_r \text{ suy ra } x_r = X_r / E_r \text{ vậy } X_r = x_r \cdot E_r; Y_r = y_r \cdot E_r; Z_r = (1 - x_r - y_r) \cdot E_r$$

Tương tự:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_r E_r & x_g E_g & x_b E_b \\ y_r E_r & y_g E_g & y_b E_b \\ (1-x_r-y_r)E_r & (1-x_g-y_g)E_g & (1-x_b-y_b)E_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Ta có điểm trắng:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = [M] \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Mà ta có theo NTSC: RGB chuẩn (x_w, y_w, Y_w) như sau: $x_w = 0.31, y_w = 0.316$ và $Y_w = 1.0$

Có:

$$y_w = \frac{Y_w}{X_w + Y_w + Z_w} (X_w + Y_w + Z_w) = \frac{Y_w}{y_w}$$

$$x_w = \frac{X_w}{X_w + Y_w + Z_w} \quad X_w = x_w (X_w + Y_w + Z_w) \quad X_w = x_w \frac{Y_w}{y_w}$$

$$\text{Và: } Z_w = (1 - x_w - y_w) \frac{Y_w}{y_w}$$

Ta có $R + G + B = W$ nên:

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} + \begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix} + \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \quad \text{suy ra } X_w = X_r + X_g + X_b = x_r E_r + x_r E_g + x_b E_b$$

Từ đó:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = \begin{bmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ (1-x_r-y_r) & (1-x_g-y_g) & (1-x_b-y_b) \end{bmatrix} \begin{bmatrix} E_r \\ E_g \\ E_b \end{bmatrix}$$

Ta hoàn toàn tính được: E_r, E_g và E_b

Tóm tắt:

Chương này chúng ta nghiên cứu cấu tạo của ánh sáng (về mặt vật lý), rồi xét bộ phận cảm nhận ánh sáng của con người là mắt. Đưa ra các hệ màu định hướng cho phần cứng như: RGB dùng cho máy tính, CMYK dùng cho máy in và YIQ dùng cho truyền hình. Tất cả các hệ màu này con người đều không cảm nhận được, con người chỉ cảm nhận được hệ màu HSV hay HLS. Từ tất cả các ưu nhược điểm của các hệ màu trên, từ sự cảm nhận màu sắc của con người phụ thuộc vào cấu tạo của các tế bào mắt nên năm 1913 tổ chức quốc tế về ánh sáng đã đưa ra hệ màu chuẩn thuần nhất CIE. Hệ màu này có thể bao hàm tất cả các hệ màu trên, nó giải quyết được các nhược điểm của hệ màu RGB. Cuối cùng chúng ta đưa ra các công thức để chuyển đổi giữa các hệ màu với nhau.

Bài tập:

1. Giả sử rằng môi trường trọng tâm là không khí (hoặc chân không) hãy mô tả dải quang phổ hiển thị bằng một dải tần số.
2. Tại sao mọi thứ trông có vẻ màu xám hoặc đen trong một phòng tối nơi chúng ta hầu như không thể nhìn thấy được?
3. Hãy thiết lập một công thức đơn giản để tính diện tích bị giới hạn bởi hàm phân bố $P(\lambda)$. (Xem hình 6.9 sách Kỹ thuật đồ họa)
4. Từ hàm phân bố $P(\lambda)$ hãy lập công thức tính độ bão hoà từ hình 6.9 - sách kỹ thuật đồ họa.
5. Sự khác nhau giữa Y trong CMY và Y trong YIQ là gì?
6. Giả sử rằng một màn hình hiển thị tạo nên những gì được gọi là màu trắng chuẩn với $x_w = 0.313$, $y_w = 0.329$ và $Y_w = 1.0$ ($R=G=B=1$). Và các tọa độ sắc phổ của phát xạ giống như tọa độ được tìm thấy ở mô hình màu.

$$R(0.62 \ 0.34)G(0.290.59)B(0.150.06)$$

Hãy tìm ma trận biến đổi màu M cho màn hình hiển thị.

7. Hãy kiểm nghiệm rằng Y trong mô hình CIE XYZ tương tự Y trong mô hình màu NTSC YIQ. Ta có NTSC chuẩn thì: $x_w = 0.31$, $y_w = 0.316$ và $Y_w = 1.0$ và sử dụng các tọa độ sắc phổ của các phát xạ NTSC chuẩn: $R(0.67 \ 0.33)$, $G(0.21 \ 0.71)$ và $B(0.14 \ 0.08)$.

Bài tập trắc nghiệm:

1. Màn hình CRT thì giá trị gamma là:
 - a. 1.5
 - b. Từ 2.3 đến 2.6
 - c. 3.3 đến 4
 - d. 1.01
2. Người bị mù màu (chỉ thấy sáng và tối) là người:
 - a. Mắt bị mất tế bào que
 - b. Mắt bị mất tế bào nón
 - c. Mắt bị mất cả tế bào nón và que
 - d. Mắt có tỷ lệ ba tế bào nón không bình thường
3. Con người nhạy cảm với màu:
 - a. Lục
 - b. Lam
 - c. Vàng
 - d. Đỏ
4. Trong ảnh đen trắng, ta biểu diễn một điểm ảnh trên màn hình theo các mẫu tô. Nếu ma trận lưới của mẫu tô kích thước 4×4 , thì chúng ta có cả thấy số mẫu tô là:
 - a. 15
 - b. 16
 - c. 17

- d. 18
- 5. Hệ màu mà con người cảm nhận là:
 - a. Hue (sắc màu), Saturation (độ bão hoà) và Lightness (độ sáng)
 - b. RGB (Red - đỏ, Green - lục, Blue - lam)
 - c. CMY (Cyan - xanh tím, Magenta - đỏ tươi và Yellow - vàng)
 - d. CIE
- 6. Ta có ba màu nước đỏ (Red), lục (Green) và lam (Blue) đem trộn các màu bão hoà và cân bằng thì thu được màu:
 - a. Trắng
 - b. Đen
 - c. Đỏ
 - d. Cả ba đều sai
- 7. Máy in màu thường gồm số hộp mực màu:
 - a. 2 hộp
 - b. 3 hộp
 - c. 4 hộp
 - d. Càng nhiều hộp thì in càng được nhiều màu

cuu duong than cong . com

cuu duong than cong . com

CHƯƠNG 7: ĐƯỜNG CONG VÀ MẶT CONG TRONG 3D

1. ĐƯỜNG CONG - CURVE

Trong các ứng dụng của đồ họa máy tính, hầu như các thực thể là đường cong mềm và mặt cong, chúng dùng để mô tả thế giới thực: nhà cửa, xe cộ, núi non....hay xây dựng nên các thực thể đang được thiết kế. Nhưng ta thấy sử dụng các phương trình đường cong không thể hiện được hình ảnh thực hay ý tưởng của người thiết kế, còn nếu ta dùng tập hợp các điểm thì thường cần nhiều dung lượng nhớ để lưu trữ cũng như tốc độ tính toán.

Ta có quỹ đạo chuyển động của một điểm trong không gian thì tạo thành đường cong. Trong chương này sẽ đưa ra phương pháp tổng thể về những mô hình toán học để biểu diễn và xây dựng các loại đường và mặt cong trong không gian 3D trên máy tính.

1.1. Điểm biểu diễn đường cong (curve represents points)

Ta thấy qua hai điểm vẽ được một đường thẳng. Qua ba điểm vẽ được một đường cong trong mặt phẳng. Qua bốn điểm vẽ được một đường cong trong không gian. Dùng các phương trình đường cong như Hypebol, parabol... thì tính toán phức tạp và không thể hiện được hình ảnh thực hay ý tưởng của người thiết kế.

Chọn đường cong như thế nào để phù hợp với máy tính? Biểu diễn *Điểm* và kiểm soát đường cong -*Points represent-and control-the curve*. Đường cong là các đối tượng cơ bản thường là kết quả của tiến trình thiết kế và các điểm đóng vai trò là công cụ để kiểm soát và mô hình hoá đường cong. Cách tiếp cận này là cơ sở của lĩnh vực thiết kế mô hình hình học nhờ máy tính (*Computer Aided Geometric Design - CAGD*).

Các cách để biểu diễn đường cong:

- Tường minh (Explicit functions):

$$y = f(x), z = g(x)$$

- Không tường minh (Implicit equations):

$$f(x,y,z) = 0$$

- Biểu diễn các đường cong tham biến (Parametric representation)

$$x = x(t), y = y(t), z = z(t) \text{ trong đó } t \in [0 \ 1]$$

Hạn chế:

- Hệ đồ hoạ ứng dụng chỉ mô tả bó hẹp trong đoạn nào đấy
- Đường cong bậc cao với mỗi giá trị của x ta luôn có 2 tập giá trị của y (thực tế chỉ cần 1)
- Chúng ta cần biểu diễn đường cong mềm (chỉ biểu diễn đường “cong gãy”)

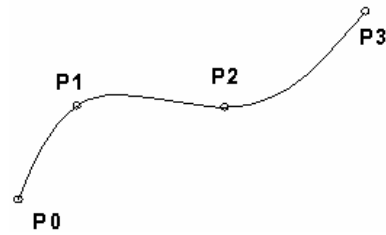
1.2. Đường cong đa thức bậc ba tham biến

Phải đảm bảo là đường cong không gian với 3 trục tọa độ x, y, z. Tránh được những tính toán phức tạp và những phần nhấp nhô ngoài ý muốn xuất hiện ở những đường đa thức bậc cao.

Công thức mô tả:

Tường minh : $y = f^2(x), z = g^3(x)$

Không tường minh: $f^3(x, y, z) = 0$



Hình 7.1 Đường cong đa thức bậc ba

Biểu diễn các đường cong tham biến (Parametric representation):

$$x = f^1(u), y = f^2(u), z = f^3(u) \text{ trong đó } u \in [0, 1]$$

Theo Lagrange:

$$x = a_1 + b_1u + c_1u^2 + d_1u^3$$

$$y = a_2 + b_2u + c_2u^2 + d_2u^3$$

$$z = a_3 + b_3u + c_3u^2 + d_3u^3$$

Ở đây ba phương trình với 12 ẩn số

Với 4 điểm p_0, p_1, p_2, p_3 phương trình xác định (vì 4 điểm thì xác định 1 đường cong trong không gian).

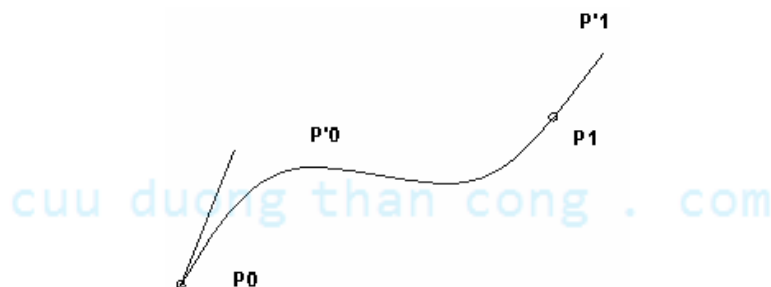
$$\text{Mỗi 1 điểm cho ta cặp 3 giá trị: } P_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \quad P_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad P_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \quad P_3 = \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix}$$

Cả thảy có 12 phương trình, thay vào 3 phương trình trên ta tính được 12 ẩn a_1, \dots, d_3

Ghi chú: rõ ràng có sự thay đổi một chút về đường cong thì ta lại phải giải lại hệ phương trình để tính các tham số cho đường cong, dẫn đến tính toán chậm.

1.3. Đường cong Hermite

Phương pháp Hermite dựa trên cơ sở của cách biểu diễn Ferguson hay Coons năm 60. Với phương pháp của Hermite đường bậc ba sẽ xác định bởi hai điểm đầu và cuối cùng với hai góc nghiêng tại hai điểm đó.



Hình 7.2 Đường cong Hermite

Theo công thức toán học hàm bậc ba được biểu diễn dưới dạng:

$$p = p(u) = k_0 + k_1u + k_2u^2 + k_3u^3$$

$$p(u) = \sum k_i u_i \in n \text{ (với } k_i \text{ là các tham số chưa biết)}$$

Độ dốc của đường cong được đo bằng $p'(u)$

$$p' = p'(u) = k_1 + 2k_2u + 3k_3u^2$$

p_0 và p_1 ta có hai độ dốc p'_0 và p'_1 với $u = 0$ và $u = 1$ tại hai điểm đầu cuối của đoạn $[0,1]$.

$$p_0(u=0)=k_0$$

$$p'_0(u=0)=k_1$$

$$p_1(u=1)=k_0+k_1+k_2+k_3$$

$$p'_1(u=1)=k_1+2k_2+3k_3$$

hay

$$k_0=p_0 \text{ và } k_1=p'_0$$

$$k_2=3(p_1 - p_0) - 2p'_0 - p'_1 \text{ và } k_3 = 2(p_0-p_1) + p'_0 + p'_1$$

Khi đã có k_0, k_1, k_2, k_3 thay vào:

$$p = p(u) = k_0 + k_1u + k_2u^2 + k_3u^3$$

$$p_0(1-3u^2+2u^3) + p_1(3u^2-2u^3) + p'_0(u-2u^2+u^3) + p'_1(-u^2+u^3)$$

$$p = p(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p'_0 \\ p'_1 \end{bmatrix}$$

Thay đổi của các điểm hay các góc nghiêng (thay đổi 2 vector) dẫn đến sự thay đổi hình dạng của đường.

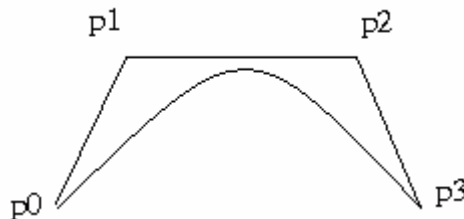
1.4. Đường cong Bezier

Việc sử dụng điểm với các vector kiểm soát được độ dốc của đường cong tại những điểm mà nó đi qua. Tuy nhiên không được thuận lợi cho việc thiết kế tương tác, không tiếp cận với các độ dốc của đường cong bằng các giá trị số (Hermite).

Paul Bezier, nhân viên hãng RENAULT vào năm 1970 đi đầu trong việc ứng dụng máy tính cho việc xây dựng các bề mặt. Hệ thống UNISURF của ông được áp dụng trong thực tế vào năm 1972 được thiết kế và kiểm xe Mezesez hay Renault.

Bezier đã sử dụng đa giác kiểm soát cho đường cong tại những đỉnh của đa giác và tiếp tuyến tại đó (p_0, p_1, p_2, p_3).

Ta có p_0, p_3 tương đương với p_0, p_1 trên đường Hermite, điểm trung gian p_1, p_2 được xác định bằng 1/3 theo độ dài của vector tiếp tuyến tại điểm p_0 và p_3



Hình 7.3 Đa giác kiểm soát Bezier

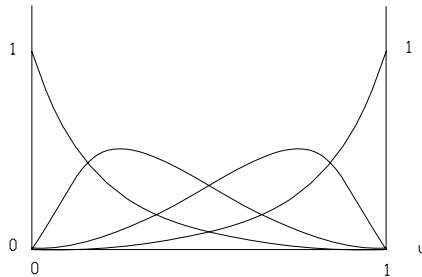
$$p'_0 = 3(p_1 - p_0)$$

$$p_1' = 3(p_3 - p_2)$$

$$p = p(u) = p_0(1-3u^2+2u^3) + p_1(3u^2-2u^3) + p_0'(u-2u^2+u^3) + p_1'(-u^2+u^3)$$

$$p = p(u) = p_0(1 - 3u + 3u^2 - u^3) + p_1(3u-6u^2+3u^3) + p_2(3u^2 - 3u^3) + p_3u^3$$

$$p = p(u) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$



Hình 7.4 Hàm hợp của đường cong Bezier

Ưu điểm:

Để dàng kiểm soát hình dạng của đường cong hơn vector tiếp tuyến tại p_0' và p_1' của Hermite.

Nằm trong đa giác kiểm soát với số điểm trung gian tùy ý (số bậc tùy ý), có số bậc = số điểm kiểm soát - 1.

Đi qua điểm đầu và điểm cuối của đa giác kiểm soát, tiếp xúc với cặp hai vector của đầu cuối đó.

Biểu thức Bezier-Bernstein

Đường Bezier cũng có thể được biết đến như biểu thức Bezier Bernstein bởi kỹ thuật mà Bezier sử dụng là áp dụng công thức hoá các vector trong phép tính đa giác xấp xỉ được Bernstein phát triển gần đây. Phép toán đại số được xác định như sau:

$$0 \leq u \leq 1$$

$$0! = 1, u_i = 1 \text{ khi } i = 0$$

$$P(u) = \sum_{i=0}^n B_{i,n}(u) P_i$$

$$B_{i,n}(u) = C(n, i) u^i (1-u)^{n-i}$$

$$C(n, i) = \frac{n!}{i!(n-i)!}$$

Trong đó P_0, \dots, P_n : vector vị trí của đa giác $(n+1)$ đỉnh.

1.5. Đường cong B-spline

1.5.1. Đường cong bậc ba Spline

Trong công thức của Bezier, chúng ta sử dụng hàm hợp liên tục để xác định điểm kiểm soát tương đối. Với các điểm nội suy thì mức độ tương đối sẽ khác nhau mà trong đó một chuỗi các phân tử nhỏ sẽ kết hợp với nhau tạo ra đường cong đa hợp. Theo tính toán thì đường bậc ba sẽ đa thức bậc thấp nhất có thể để biểu diễn một đường cong trong không gian và chuỗi điểm Hermite sẽ phù hợp nhất đối với việc xây dựng nên đường cong đa hợp này.

Việc yêu cầu người sử dụng đưa vào các vector tiếp tuyến tại mỗi điểm trong tập hợp các điểm là cực kỳ bất tiện cho nên thường trong các đường bậc ba đa hợp ta sử dụng các điều kiện biên liên tục trong phép đạo hàm bậc một và hai tại điểm nối giữa. và đường cong được xác định như trên gọi là đường spline bậc ba với phép đạo hàm liên tục bậc hai. Giá trị đạo hàm của đường cong sẽ xác định độ cong tại mỗi điểm nút và nó cũng đưa ra điều kiện biên cho mỗi đoạn trên đường cong.

Vậy đường bậc ba spline có ưu điểm là không phải xác định độ dốc của đường tại các nút nhưng nhược điểm của nó là chỉ tạo ra sự thay đổi toàn cục khi ta thay đổi vị trí của điểm.

Đường cong – Spline đi qua n điểm cho trước mà mỗi đoạn là các đường cong bậc ba độc lập có độ dốc và độ cong liên tục tại mỗi điểm kiểm soát hay điểm nút. Với n điểm ta có $(n-1)$ đoạn với mỗi đoạn gồm bốn vector hệ số hay $4(n-1)$ cho $n-1$ đoạn, và $2(n-1)$ điều kiện biên và $(n-2)$ điều kiện về độ dốc cùng $(n-2)$ về độ cong.

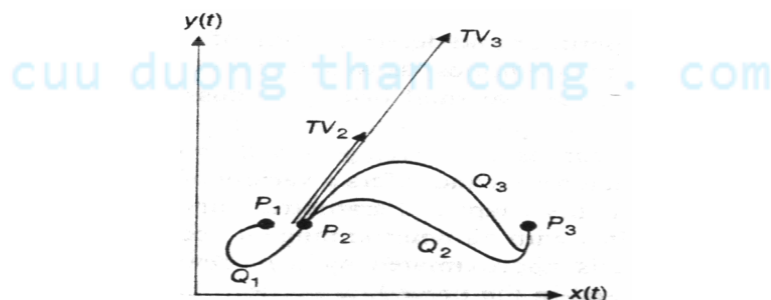
Để xây dựng nên đường spline có tham số với n điểm nút ta có một dãy các giá trị tham số mà ta gọi là vector nút.

$$u_0, \dots, u_{n-1} \text{ trong đó } u_{i+1} > u_i$$

Cần lựa chọn tại mỗi nút, cách lựa chọn đơn giản nhất là theo cách đơn điệu có nghĩa là với giá trị 0 tại điểm đầu và tăng lên 1 tại những điểm kế tiếp. tuy vậy phương pháp này dẫn đến độ cong không mong muốn tại các điểm vì vậy việc tham số hoá sẽ đưa vào chiều dài, nhưng phương pháp này cũng không được chính xác khi mà đường cong chưa xác định chiều dài. Tuy nhiên thông thường người ta sử dụng việc tích lũy của các dây cung với:

$$u_0 = 0 \text{ và } u_{i+1} = u_i + d_{i+1} \text{ trong đó } d_i: \text{ là khoảng cách giữa 2 điểm } p_{i-1} \text{ và } p_i$$

Trong các trường hợp đường cong có bậc lớn hơn ba có thể dùng cho đường spline. Thông thường đường spline bậc n sẽ được xây dựng trên các phần nhỏ liên tục của các biến độc lập.



Hình 7.5 Kết nối hai đường cong

Hình trên cho thấy hai đoạn cong có chung điểm nối mà đường cong liên tục tại điểm đó, việc biểu diễn tính liên tục của đường cong thông qua chữ cái C-Continue. C_0 để đảm bảo không

có sự gián đoạn giữa hai đoạn cong. C_1 tính liên tục bậc nhất hay đạo hàm bậc nhất tại điểm nối. C_2 đạo hàm bậc hai liên tục của đường cong tại điểm nối.

Giả sử khi biểu diễn đường cong mềm thông qua các đoạn cong q_1, q_2, q_3 (mỗi đoạn có 4 vector hệ số) cần thỏa mãn:

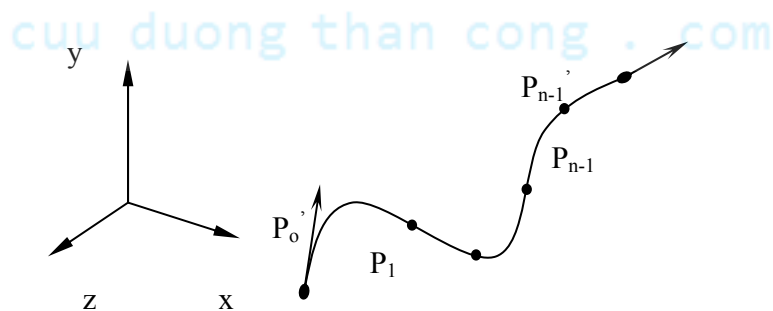
Liên tục tại điểm nối hay $C^1_0 = C^2_0$

Độ dốc (hay vector tiếp tuyến) tại điểm nối (điểm cuối của q_1 và đầu q_2) là như nhau: $C^1_1 = C^2_1$ (đạo hàm bậc nhất)

Thỏa mãn liên tục trên tại điểm nối (đạo hàm bậc 2 liên tục tại điểm nối) $C^1_2 = C^2_2$

Việc kết hợp các đoạn cong Hermite bậc ba để mô tả một đường cong mềm theo kiểu phân đoạn spline là phương pháp đơn giản nhất hay còn gọi là phương pháp Hermite nội suy. Với phương pháp này thì tham biến u_i cho mỗi đoạn cong i của tập các đoạn cong Hermite sẽ biến đổi trong khoảng từ 0 đến 1 và luôn tồn tại đạo hàm bậc nhất của các đoạn cong tại các điểm nối. Phương trình cho mỗi đoạn cong được sử dụng lúc này là phương trình đường cong bậc ba Hermite:

$$p = p(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p'_0 \\ p'_1 \end{bmatrix}$$



Hình 7.6 Phân đoạn của đường cong Spline - Hermite

Theo Hermite các đoạn là các đường cong, tính liên tục của đạo hàm bậc hai tại các điểm nối có thể dễ dàng đạt được bằng cách đặt $P''_{i-1}(u_{i-1}=1)$ là đạo hàm bậc hai tại điểm cuối của đoạn $(i-1)$ bằng với $P''_i(u_i=0)$ đạo hàm bậc hai tại điểm đầu của đoạn thứ i .

$$P''_{i-1}(1) = P''_i(0)$$

Có phương trình:

$$P_i(u) = k_{0i} + k_{1i}u + k_{2i}u^2 + k_{3i}u^3$$

Đạo hàm bậc hai sẽ là:

$$P''_i(u) = 2k_{2i} + 6k_{3i}u$$

$$P''_{i-1}(1) = P''_i(0) \text{ nên } 2k_{2(i-1)} + 6k_{3(i-1)} = 2k_{2i}$$

Vì điểm cuối của đoạn $i-1$ trùng với điểm đầu của đoạn thứ i ($P_i(0) = P_{i-1}(1)$)

Theo Hermite: $k_2 = 3(p_1 - p_0) - 2p'_0 - p'_1$ và

$$k_3 = 2(p'_0 - p'_1) + p_0'' + p_1''$$

$$2(3(P_i - P_{i-1}) - 2P'_{i-1} - P'_i) + 6(2(P_{i-1} - P_i) + P'_{i-1} + P'_i) = 2(2(P_{i-1} - P_i) + P'_{i-1} + P'_i)$$

$$\text{Hay: } P'_{i-1} + 4P'_i + P'_{i+1} = 3(P_{i+1} - P_i) \quad (*)$$

Với phương trình (*) này thì phương trình dạng tổng quát của đường cong Spline là tập của các đoạn cong Hermite sẽ xác định với điều kiện ban đầu cho là tập các điểm kiểm soát của đường cong và hai vector tiếp tuyến tại hai điểm đầu cuối của đường cong đó. Sử dụng (*) ta có thể tính được các giá trị của các vector tiếp tuyến tại từng điểm kiểm soát của đường cong.

$$\begin{bmatrix} 1 & 0 & . & . & . & . \\ 1 & 4 & 1 & 0 & . & . \\ 0 & 1 & 4 & 1 & 0 & . \\ . & . & . & . & . & . \\ . & . & 0 & 1 & 4 & 1 \\ . & . & . & . & 0 & 1 \end{bmatrix} \begin{bmatrix} P'_0 \\ P'_1 \\ . \\ . \\ P'_{n-2} \\ P'_{n-1} \end{bmatrix} = \begin{bmatrix} P'_0 \\ 3(P'_2 - P'_0) \\ . \\ . \\ 3(P'_{n-1} - P'_{n-3}) \\ P'_{n-1} \end{bmatrix}$$

Tương đương với:

$$\begin{bmatrix} P'_0 \\ P'_1 \\ . \\ . \\ P'_{n-2} \\ P'_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & . & . & . & . \\ 1 & 4 & 1 & 0 & . & . \\ 0 & 1 & 4 & 1 & 0 & . \\ . & . & . & . & . & . \\ . & . & 0 & 1 & 4 & 1 \\ . & . & . & . & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} P'_0 \\ 3(P'_2 - P'_0) \\ . \\ . \\ 3(P'_{n-1} - P'_{n-3}) \\ P'_{n-1} \end{bmatrix}$$

1.5.2. Đường B-spline

Với Bezier hay spline đều không cho ta thay đổi đường cong một cách cục bộ, việc thay đổi vị trí các điểm kiểm soát hay các vector tiếp tuyến không chỉ ảnh hưởng trực tiếp đến độ dốc của đường cong lân cận quanh điểm kiểm soát mà còn kéo theo ảnh hưởng đến các phần còn lại của đường cong. Đường Bezier thêm vào đó là khi tính xấp xỉ ở bậc cao sẽ rất phức tạp còn khi liên kết nhiều đoạn Bezier hay Hermite bậc thấp (bậc ba) có thể đem lại ích lợi khi tính toán nhưng yếu tố ràng buộc về tính liên tục của đạo hàm bậc cao tại các điểm nối không cho điều khiển cục bộ như mong muốn.

Việc kết hợp luôn phiên các đoạn cong tổng hợp, thông qua các đa thức tham số xác định riêng rẽ trên một số điểm kiểm soát lân cận với số bậc tùy ý không phụ thuộc vào số lượng các điểm kiểm soát, cho phép tạo nên đường cong trơn mềm B-spline. Đường cong này đã khắc phục được các nhược điểm mà các dạng đường cong trước chưa đạt được. Có nghĩa là khi dịch chuyển điểm kiểm soát của đường cong thì chỉ một vài phân đoạn lân cận của điểm kiểm soát đó bị ảnh hưởng chứ không phải toàn bộ đường cong.

Với $n+1$ số điểm kiểm soát P_i ta có:

$$P(u) = \sum_{i=0}^n N_{i,k}(u) \cdot P_i$$

Trong đó $N_{i,k}(u)$ là hàm hợp B-Spline bậc $k-1$ và sự khác biệt giữa B-spline và Bezier sẽ được thể hiện trên đó. Trong đường Bezier bậc của đa thức được xác định bởi số đoạn cong trên đường cong đó, còn với B-spline bậc được thỏa mãn độc lập với số điểm kiểm soát của đường.

Hơn nữa hàm hợp của Bezier khác 0 trên toàn bộ khoảng của tham số u còn B-spline chỉ khác 0 trên đoạn ngắn của các tham số. Mỗi đoạn trên hàm hợp chỉ tương ứng với một điểm thì chỉ dẫn tới sự thay đổi cục bộ trong khoảng mà trên đó tham số của hàm hợp khác 0.

Biểu diễn toán học của B-spline, với hàm B-spline có bậc $k-1$ xác định thì:

$$N_{i,k}(u) = \frac{(u - U_{i+1-k})}{(U_i - U_{i+1-k})} N_{i-1,k-1}(u) + \frac{(U_{i+1} - u)}{(U_{i+1} - U_{i+2-k})} N_{i,k-1}(u)$$

$$N_{i,1}(u) = \begin{cases} 1 & u \in [u_i, u_{i+1}] \\ 0 & \text{others} \end{cases}$$

Trong đó u_i là giá trị tại nút p_i với biến số là u được gọi là các vector nút.

Tất cả các giá trị nút đồng thời xác định trên vector nút và các nút nguyên thường sử dụng dễ dàng. Trong trường hợp này các hàm hợp bậc k sẽ khác 0 trong khoảng k của vector nút và toàn bộ các giá trị trên vector cho một tập hợp điểm bằng $n+1+k$.

Không như Bezier, đường B-spline không đi qua hai điểm đầu và cuối trừ khi hàm hợp được dùng là tuyến tính.

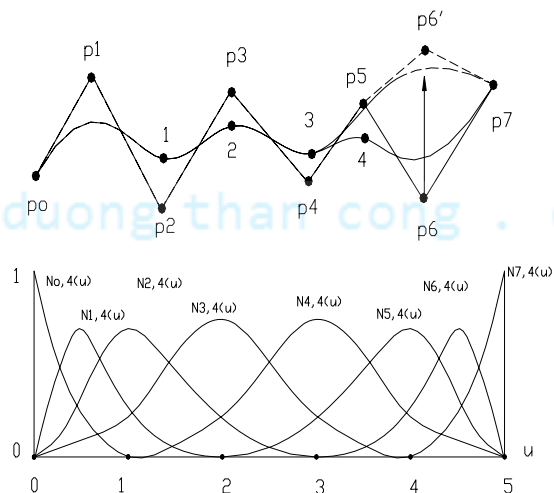
Đường B-spline có thể được tạo qua hai điểm đầu, cuối và tiếp xúc với vector đầu và cuối của đa giác kiểm soát. Bằng cách thêm vào các nút tại vị trí của các nút cuối của vector tuy nhiên các giá trị giống nhau không nhiều hơn bậc của đường cong.

Giống như đường cong Bezier, tính chất bao lồi của đa giác kiểm soát và tính chất chuẩn được thỏa mãn. Vậy có:

$$\sum_{i=0}^n N_{i,k}(u) = 1$$

Trong đường cong B-spline, số lượng các nút, bậc của đường cong và số điểm điều khiển luôn có các quan hệ ràng buộc:

$$0 \leq u \leq n - k + 2$$



Hình 7.7 Đường cong B-spline

Vậy việc xác định các vector nút sẽ phụ thuộc vào sự phân loại của chính bản thân chúng và điều đó sẽ ảnh hưởng đến hình dạng của đường cong được mô tả. Phân loại sẽ dựa trên loại của đường cong như sau:

Đều tuần hoàn (periodic)

Không tuần hoàn (open or unperiodic)

Không đều (non-uniform)

a. B Spline - Đều và tuần hoàn

Vector nút là đều khi giá trị của chúng cách đều nhau một khoảng ∇ xác định.

Ví dụ: [0 1 2 3 4 5] với ∇ xác định = 1

[-2-1/2 1 5/2 4] với ∇ xác định = 3/2

[-1-0.6 -0.2 0.2 0.6 1] với ∇ xác định = 0.4

Trong các bài toán thực tế, thông thường thì khoảng xác định của tham biến nằm trong khoảng từ 0 đến 1 hay từ 0^0 đến 360^0 thì việc chọn giá trị của các vector nút được chuẩn hoá trong khoảng [0 1] hay $[0^0 \ 360^0]$ đó.

[00.2 0.4 0.6 0.81] với ∇ xác định = 0.2

[$0^0 120^0 240^0 360^0$] với ∇ xác định = 120^0

Bậc (k-1)	Cấp (k)	Vector nút (m=n+k)	Khoảng tham số $(k-1) \leq t \leq (n+1)$
1	2	[0 1 2 3 4 5 6 7]	$1 \leq t \leq 6$
2	3	[0 1 2 3 4 5 6 7 8]	$2 \leq t \leq 6$
3	4	[0 1 2 3 4 5 6 7 8 9]	$3 \leq t \leq 6$

Các vector nút gọi là đều và tuần hoàn khi các hàm B-spline đối với mỗi phân đoạn có thể chuyển đổi lẫn nhau. Bảng trên chỉ ra sự thay đổi của miền tham số và vector nút khác nhau của các đường cong B-spline khi bậc của đường cong thay đổi. Số lượng của vector nút được qui định bởi biểu thức $m-n+k$ và số lượng các điểm kiểm soát tính qua biểu thức $(n+1)$ bằng 6.

Tính chất:

Ảnh hưởng của mỗi hàm cơ sở được giới hạn trong k đoạn là cấp của đường cong cần thể hiện. Vậy chúng ta sử dụng đường cong bậc ba thì ảnh hưởng của hàm cơ sở trải dài trên bốn đoạn của đường cong.

Đường B-spline tuần hoàn không đi qua các điểm đầu và cuối của đa giác kiểm soát ngoại trừ với đường bậc 1 ($k=2$) mà khi đó đường cong chuyển dạng thành đường thẳng.

Ví dụ về các đường B-spline tuần hoàn có các bậc khác nhau có cùng các điểm và đa giác kiểm soát. Khi $k=2$ đường cong bậc một trùng với các cạnh của đa giác kiểm soát.

Khi $k=3$, đường cong B-spline bậc 2, bắt đầu tại trung điểm của cạnh thứ nhất và kết thúc tại trung điểm của cạnh cuối cùng của đa giác kiểm soát.

b. Không tuần hoàn (Open – Non Uniform)

Một vector không tuần hoàn hoặc mở là vector nút có giá trị nút tại các điểm đầu cuối lặp lại với số lượng các giá trị lặp lại này bằng chính cấp k của đường cong và các giá trị nút trong mỗi điểm lặp này là bằng nhau

Nếu một trong hai điều kiện này hoặc cả hai điều kiện không được thoả mãn thì vectơ nút là không đều.

Ví dụ, xét một đa giác kiểm soát với bốn đỉnh. Các đường cong B-spline cấp 2,3,4 được xây dựng dựa trên đa giác kiểm soát có số lượng các nút $m=n+k$ sẽ có vector nút như sau:

Cấp (k)	số lượng nút ($m = n + k$)	Vector nút không tuần hoàn
2	6	[0 0 1 2 3 3]
3	7	[0 0 0 1 2 2 2]
4	8	[0 0 0 0 1 1 1 1]

Các biểu thức phải được thoả mãn đối với nút u_i trên vector nút không tuần hoàn bắt đầu tại u_0 .

Danh sách các vector nút không tuần hoàn đã đưa ra ở mục này đều thoả mãn các biểu thức sau:

$$u_i = 0 \quad 1 \leq i \leq k$$

$$u_i = i - k \quad k + 1 \leq i \leq n + 1$$

$$u_i = n - k + 2 \quad n + 1 \leq i \leq n + k + 1$$

Các vector nút không tuần hoàn cung cấp các hàm cơ sở được định nghĩa trong một miền tham số phức tạp và không có sự mất mát như với loại vector tuần hoàn và vì vậy đường cong B-spline loại này luôn đi qua các điểm đầu và cuối của đa giác kiểm soát.

Ví dụ: hàm hợp bậc ba tính xấp xỉ cho 8 khoảng sẽ xác định trên vector nút là 00001234555. Ở đây chúng ta còn thấy sự thay đổi cục bộ trên đường cong khi ta thay đổi vị trí mỗi điểm.

Đường cong Bezier là trường hợp đặc biệt của B-spline không tuần hoàn, trong đó số lượng các đỉnh sử dụng bằng với cấp của đường cong. Vector nút trong trường hợp này là:

$$[0 \ 0 \ \dots \ 0 \ 1 \ 1 \ \dots \ 1 \ 1]$$

kk

Đường cong B-spline bậc ba với bốn điểm kiểm soát và vector không tuần hoàn [0 0 0 0 1 1 1 1] cũng chính là đường cong Bezier.

c. Không đều

Trong vector nút không tuần hoàn, giá trị các nút xuất hiện tại các biên được lặp lại và các nút bên trong các bước nút bằng nhau. Nếu một trong hai điều kiện này hoặc cả hai điều kiện này không được thoả mãn thì vector nút là không đều.

Ví dụ các nút không đều có thể tạo ra bằng cách đặt các giá trị lặp lại đối với các nút ở khoảng giữa [0 1 2 3 3 4 5]

Hay tạo ra bước nhảy không bằng nhau giữa các nút [0.0 0.2 0.5 0.75 1.0]

Các vector nút loại đều cho phép người sử dụng dễ hình dung và xử lý trong các phép toán nhưng trong một số các trường hợp bước nút không đều lại có những ưu điểm đặc biệt. Ví dụ như trong việc điều khiển hình dạng của đường cong trong tiến trình thiết kế khi các sai lệch không

mong muốn có thể xuất hiện mà việc sử dụng đường cong B-spline đều với các dữ liệu điểm có các khoảng cách tương đối lớn mà không đều nhau.

Kết luận

- B-spline là một dòng của Bezier
- Thực tế khi ta chọn bậc k cho tập hợp k điểm thì B-spline chuyển thành Bezier
- Khi bậc của đa thức giảm sự ảnh hưởng cục bộ của mỗi điểm nút càng rõ ràng hơn.
- Khi tồn tại ảnh hưởng cục bộ càng lớn và đường cong phải đi qua điểm đó.
- Chúng ta có thể thay đổi hình dạng đường cong B-spline bằng cách:
 - Thay đổi kiểu vector nút: đều tuần hoàn, mở, không đều
 - Thay đổi cấp k của đường cong
 - Thay đổi số đỉnh và vị trí các đỉnh đa giác kiểm soát
 - Sử dụng các điểm kiểm soát trùng nhau

2. MÔ HÌNH BỀ MẶT (Surface) VÀ CÁC PHƯƠNG PHÁP XÂY DỰNG

2.1. Các khái niệm cơ bản

Mặt cong (surface): là quỹ đạo chuyển động của một đường cong tạo nên. Biểu diễn tham biến cho mặt cong:

- ✓ Dựa vào việc xây dựng và tạo bề mặt toán học trên những điểm dữ liệu
- ✓ Dựa trên việc xây dựng nên bề mặt phụ thuộc vào biến số có khả năng thay đổi một cách trực diện thông qua các tương tác đồ họa.

Ta có:

$$\begin{aligned}x &= x(u, v, w) & u, v, w &\in [0, 1] \\y &= y(u, v, w) & u + v + w &= 1 \\z &= z(u, v, w) \\Q(u, v, w) &= Q[x=x(u, v, w) \ y=y(u, v, w) \ z=z(u, v, w)]\end{aligned}$$

Biểu diễn theo mảnh

- ✓ Biểu diễn miếng tứ giác - quadrilatera Patches
- ✓ Biểu diễn miếng tam giác - Triangular Patches

2.2. Biểu diễn mảnh tứ giác

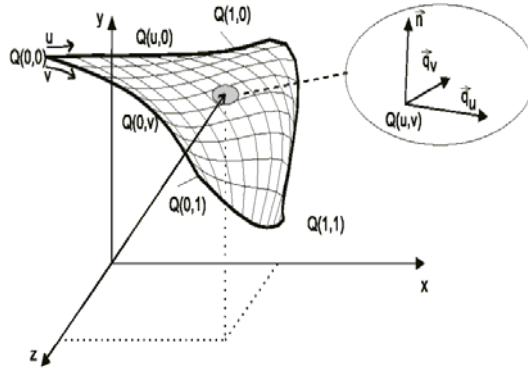
Phương trình:

$$\begin{aligned}x &= x(u, v) \\y &= y(u, v) & u, v &\in [0, 1] \\z &= z(u, v) \\Q(u, v) &= Q[x=x(u, v) \ y=y(u, v) \ z=z(u, v)]\end{aligned}$$

Thành phần

u, v là các tham biến

Các điểm $Q(0,0)$, $Q(0,1)$, $Q(1,0)$, $Q(1,1)$ là cận của mảnh, các đường cong $Q(1,v)$, $Q(0,v)$, $Q(u,0)$, $Q(u,1)$ là các biên của mảnh. Đạo hàm riêng tại điểm $Q(u,v)$ xác định vector tiếp tuyến theo hướng u , v .

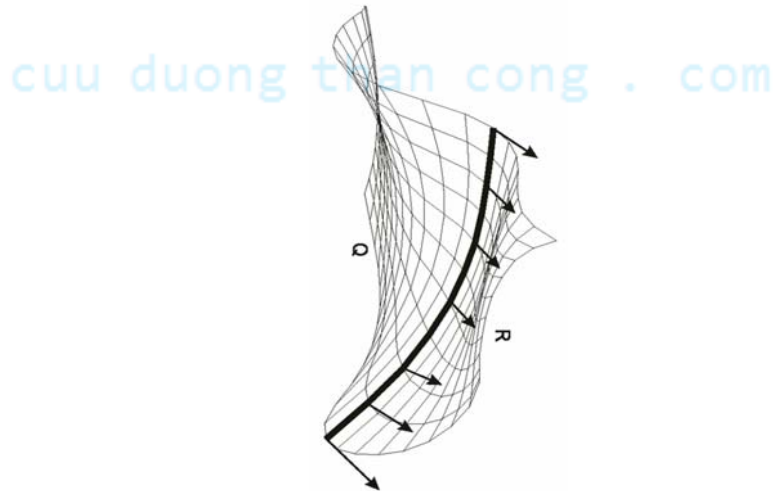


Hình 7.8 Biểu diễn mảnh tứ giác

$$\frac{\partial Q(u,v)}{\partial u} = Q\left[\frac{\partial x(u,v)}{\partial u}, \frac{\partial y(u,v)}{\partial u}, \frac{\partial z(u,v)}{\partial u}\right]$$

$$\frac{\partial Q(u,v)}{\partial v} = Q\left[\frac{\partial x(u,v)}{\partial v}, \frac{\partial y(u,v)}{\partial v}, \frac{\partial z(u,v)}{\partial v}\right]$$

2.2.1. Kết nối mảnh tứ giác



Hình 7.9 Kết nối mảnh tứ giác

Thực thể hình học biểu diễn thông qua các mảnh cùng dạng, các mảnh có thể nối với nhau theo các hướng u, v khi hai mảnh cùng hướng đó. Nếu mọi điểm trên biên của hai mảnh bằng nhau, hay hai biên bằng nhau. Hai mảnh liên tục bậc C_0 . Nếu hai biên bằng nhau và đạo hàm bằng nhau trên cùng một hướng thì hai mảnh gọi là kết nối bậc C_1 .

2.2.2. Hệ tọa độ Barycentric Coordinates

Tập các điểm $P_1, P_2 \dots P_n$, tập các tổ hợp của các điểm đó

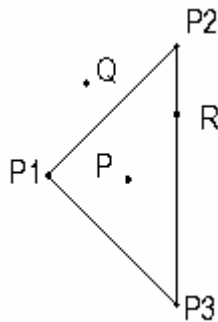
$$k_1 P_1 + k_2 P_2 + k_3 P_3 \dots + k_n P_n$$

Với $k_1 + k_2 + k_3 + \dots + k_n = 1$

Các điểm tạo thành không gian affine với các giá trị tọa độ nates

$k_1, k_2, k_3, \dots, k_n$ được gọi là hệ tọa độ barycentric.

2.2.3. Tam giác – *Triangular*



Hình 7.10 Mảnh tam giác

Trong tam giác các điểm có dạng P_1, P_2, P_3

Hệ số: $k_1, k_2, k_3 \in [0, 1]$

$$k_1 + k_2 + k_3 = 1$$

$$P = k_1 P_1 + k_2 P_2 + k_3 P_3$$

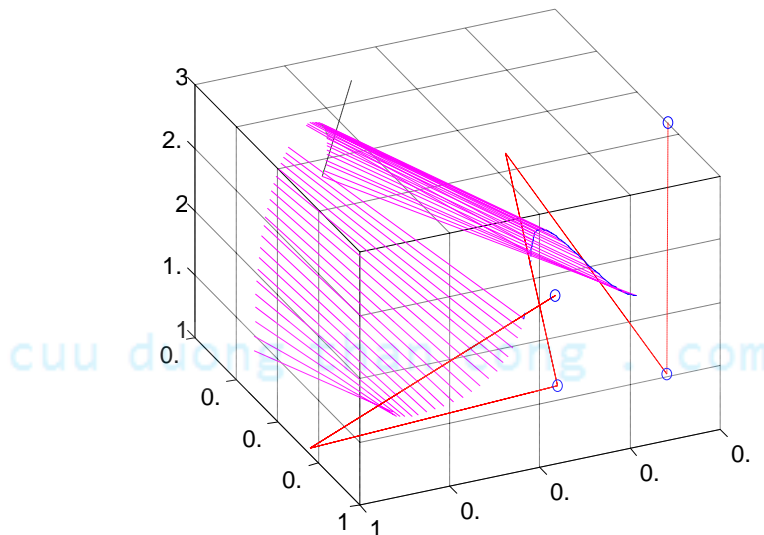
Nếu Hệ số $k_i > 1$ hoặc < 0 điểm P sẽ nằm ngoài tam giác (Q)

Nếu Hệ số $k_i = 1$ hoặc $= 0$ điểm P sẽ nằm trên cạnh tam giác (R)

2.3. Mô hình hoá các mặt cong (Surface Patches)

2.3.1. Mặt kẻ (*Ruled Surface*)

Bề mặt được xây dựng bằng cách cho trượt một đoạn thẳng trên hai đường cong. Các mặt kẻ nhận được bằng phép nội suy tuyến tính từ hai đường cong biên cho trước tương ứng với hai biên đối diện của mặt kẻ $P_1(u)$ và $P_2(u)$.



Hình 7.11 Mô hình bề mặt kẻ

Phương trình mặt kẻ:

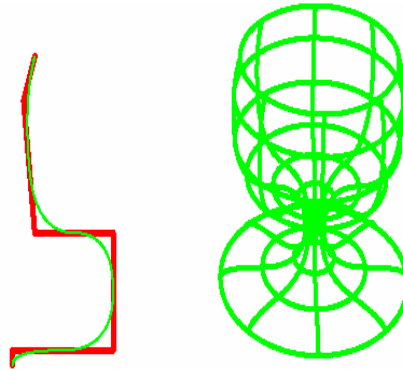
$$Q(u, v) = P_2(u)v + P_1(u)(1-v)$$

Nếu hai đường cong cho trước tương ứng là $P_1(v)$ và $P_2(v)$

Thì mặt kẻ có phương trình:

$$Q(u, v) = P_1(v)(1-u) + P_2(v)u = \begin{bmatrix} P_1(v) \\ P_2(v) \end{bmatrix}$$

2.3.2. Mặt tròn xoay (Revolution surface)



Hình 7.12 Mô hình mặt tròn xoay

Mặt được xây dựng bởi đường thẳng hay một đường cong phẳng, quanh một trục trong không gian.

Giả sử đường cong phẳng có dạng:

$$P(t) = [x(t) \ y(t) \ z(t)] \quad 0 \leq t \leq t_{\max}$$

Ví dụ: quay quanh trục x một thực thể nằm trên mặt phẳng xoy, phương trình bề mặt là

$$Q(t, f) = [x(t) \ y(t) \cos f \ z(t) \sin f]$$

$$0 \leq \phi \leq 2\pi$$

Ví dụ: Mặt tròn xoay

$P_1[1 \ 1 \ 0]$ và $P_2[6 \ 2 \ 0]$ nằm trong mặt phẳng xoy. Quay đường thẳng quanh trục ox sẽ được một mặt nón. Xác định điểm của mặt tại $t=0.5, f=\pi/3$.

Phương trình tham số cho đoạn thẳng từ P_1 tới P_2 là:

$$P(t) = [x(t) \ y(t) \ z(t)] = P_1 + (P_2 - P_1)t \quad 0 \leq t \leq 1$$

Với các thành phần Đề-các:

$$x(t) = x_1 + (x_2 - x_1)t = 1 + 5t$$

$$y(t) = y_1 + (y_2 - y_1)t = 1 + t$$

$$z(t) = z_1 + (z_2 - z_1)t = 0$$

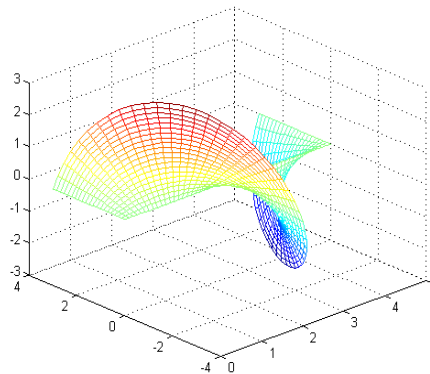
Dùng phương trình:

$$Q(1/2, \pi/3) = [1 + 5t(1+t)\cos f \ (1+t)\sin f]$$

$$= \begin{bmatrix} 7 \\ 2 \\ \frac{3}{2} \cos \frac{\pi}{3} \\ \frac{3}{2} \sin \frac{\pi}{3} \end{bmatrix}$$

$$= \begin{bmatrix} 7 \\ 2 \\ \frac{3}{4} \\ \frac{3\sqrt{3}}{4} \end{bmatrix}$$

2. 3.3. Mặt trượt (Swept Surface)



Hình 7.13 Mô hình mặt trượt

Sweep surface là mặt được tạo bởi bằng cách trượt một thực thể.

Ví dụ: một đường thẳng, đa giác, một đường cong, một hình... dọc theo một đường trong không gian.

$$Q(u,v) = P(u) * [T(v)]$$

$P(u)$ thực thể cần trượt

$[T(v)]$ là ma trận biến đổi ($[T(v)]$ có thể là ma trận tịnh tiến, quay, hay tỉ lệ ... hoặc là kết hợp của nhiều phép biến đổi đó).

Ví dụ:

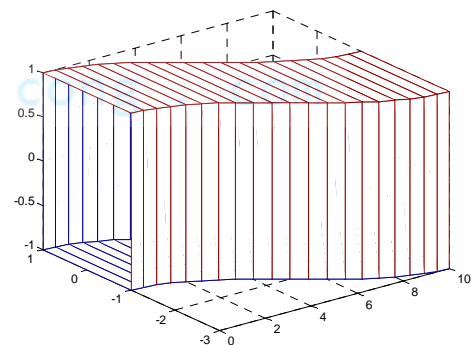
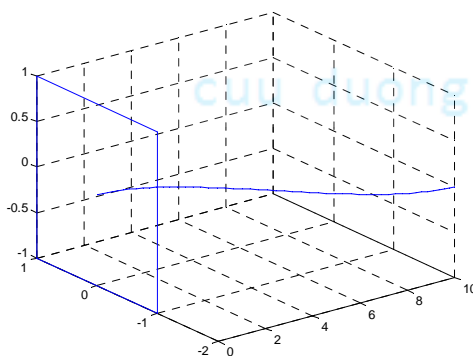
$$P_1[000], P_2[03\ 0]$$

$$P(t) = P_1 + (P_2 - P_1) * u = [0\ 3u\ 0]$$

$$0 \leq u, v \leq 1$$

$$[T(v)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(2\pi v) & \sin(2\pi v) & 0 \\ 0 & -\sin(2\pi v) & \cos(2\pi v) & 0 \\ 10v & 0 & 0 & 1 \end{bmatrix}$$

Ví dụ về mặt trượt (Swept Extrusion)



Hình 7.14 Hình thành mặt trượt

Hình vuông xác định bởi 4 đỉnh:

$$P_1[0 \ -10], P_2[0-1-1]$$

$$P_3[01 \ -1], P_4[01 \ 1]$$

Đường cong trượt:

$$x = 10v = \cos(\Pi v) - 1$$

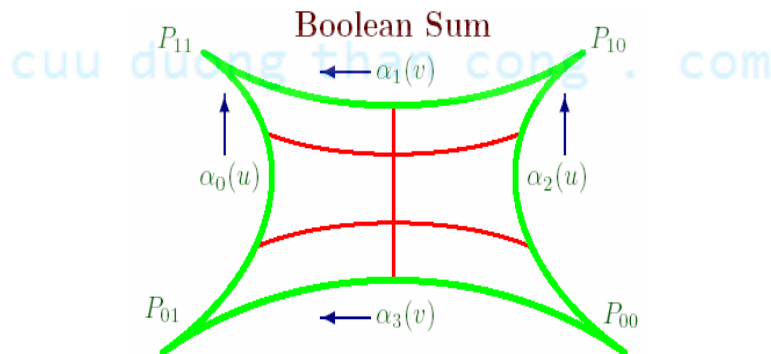
$$|P(u)| = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 & 1 \\ 0 & -1 & -1 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & -1 & 1 & 1 \end{bmatrix} \quad |T(v)| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 10v & \cos(\Pi v) - 1 & 0 & 1 \end{bmatrix}$$

Quay 1 góc khi trượt:

$$\begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 10v & \cos(\Pi v) - 1 & 0 & 1 \end{bmatrix}$$

2.3.4. Mặt nội suy trên bốn đường biên (Boolean sum surface)

Mặt được xây dựng trên 4 điểm và các đường cong biên, $S(u,v)$ mặt nội suy trên 4 đường biên.



Hình 7.15 Mô hình mặt cong Boolean Sum

$$S(u, v) = S_1(u, v) + S_2(u, v) - P(u, v)$$

Với:

$$P(u, v) = (1-u)(1-v)P_{00} + (1-u)vP_{01} + u(1-v)P_{10} + uvP_{11}$$

$$S_1(u, v) = va_0(u) + (1-v)a_2(u)$$

$$S_2(u, v) = ua_1(v) + (1-u)a_3(v);$$

P là các đỉnh của mảnh 4

$a_i(u)$ là các phương trình đường biên

Ví dụ về mặt Boolean Sum:

Với $u = 0$

$$S(0, v) = S_1(0, v) + S_2(0, v) - P(0, v)$$

$$= v a_0(0) + (1 - v)a_2(0) + 0a_1(v) + 1 a_3(v) - (1 - v)P_{00} - v P_{01}$$

$$= v P_{01} + (1 - v)P_{00} + a_3(v) -(1 - v)P_{00} - v P_{01}$$

$$= a_3(v)$$

2.4. Mặt từ các đường cong

2.4.1. Mặt cong bậc ba Hermite

Mặt cong tham biến được tạo bởi bề mặt qua tại 4 điểm dữ liệu tại 4 góc và các đường cong có phương trình bậc ba qua chúng, như vậy 16 vector điều kiện hay tương đương với 48 giá trị đại số cần thiết để xác định các hệ số phương trình.

Khi những hệ số là 4 điểm dữ liệu góc và 8 vector tiếp tuyến tại các điểm đó theo các hướng u, v tương ứng cùng 4 vector xoắn thì mặt cong tạo thành là mặt cong Hermite. Phương trình có dạng:

$$Q(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 C_{ij} u^i v^j \quad 0 \leq u, v \leq 1$$

$$Q(u, v) = [U] [C] [V]^T \quad 0 \leq u, v \leq 1$$

$$\text{Với: } u = [u^3 \ u^2 \ u \ 1], v = [v^3 \ v^2 \ v \ 1]$$

Và ma trận hệ số $[C]$ là:

$$[C] = \begin{bmatrix} C_{00} & C_{01} & C_{02} & C_{03} \\ C_{10} & C_{11} & C_{12} & C_{13} \\ C_{20} & C_{21} & C_{22} & C_{23} \\ C_{30} & C_{31} & C_{32} & C_{33} \end{bmatrix}$$

Cuối cùng ta thu được các hệ số theo phương trình mới có dạng:

$$Q(u, v) = [U] [M_H] [B] [M_H]^T [V]^T \quad u, v \in [0, 1]$$

$$[M_H] = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Và

Và B là ma trận điều kiện biên:

$$[B] = \left[\begin{array}{cc|cc} P_{00} & P_{01} & P_{v00} & P_{v01} \\ P_{10} & P_{11} & P_{v10} & P_{v11} \\ \hline P_{u00} & P_{u01} & P_{uv00} & P_{uv01} \\ P_{u10} & P_{u11} & P_{uv10} & P_{uv11} \end{array} \right]$$

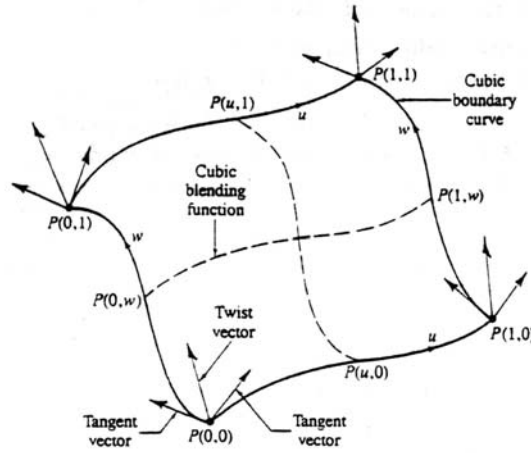
Hay với dạng thức rút gọn của ma trận $[B]$ theo các ma trận điều kiện biên tương ứng:

$[B]$ ma trận các giá trị tham số

$[P_u], [P_v]$ các vector tiếp tuyến theo u, v tương ứng.

$[P_{uv}]$ ma trận xoắn trên u, v

$$[B] = \begin{bmatrix} [P] & [P_v] \\ [P_u] & [P_{uv}] \end{bmatrix}$$



Hình 7.16 Mặt cong Hermite và các điểm dữ liệu

Các vector tiếp tuyến và vector xoắn của bề mặt cong được biểu diễn qua phương trình sau:

$$Q_u(u,v) = [U] [M_H]^u [B] [M_H]^T [V]^T$$

$$Q_v(u,v) = [U] [M_H]^v [B] [M_H]^T [V]^T$$

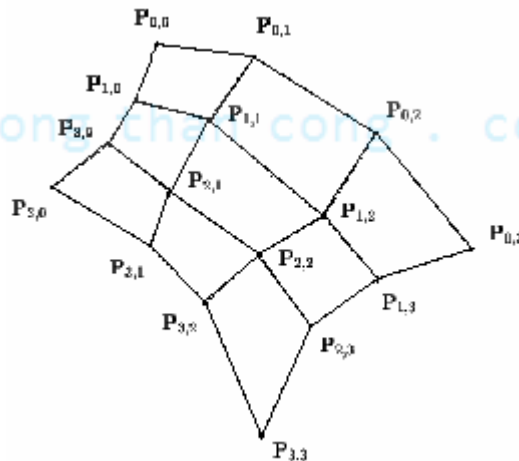
$$Q_{uv}(u,v) = [U] [M_H]^{uv} [B] [M_H]^T [V]^T$$

Với:

$$[M_H]^u \text{ & } [M_H]^v = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & -6 & 3 & 3 \\ -6 & 6 & -4 & -2 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.4.2. Mặt cong Bezier

Mảnh-patch Bézier



Hình 7.17 Mặt cong Bezier

Mảnh Bezier được hình thành trên phép trượt của đường cong Bezier. Việc xây dựng nên mảnh Bezier dưới các điểm kiểm soát, tạo nên đa diện kiểm soát.

$$\{P_{i,j} : 0 \leq i \leq n, 0 \leq j \leq m\}$$

Phương trình tổng quát của mặt cong tham biến Bezier có dạng:

$$P(u, v) = \sum_{j=0}^m \sum_{i=0}^n P_{i,j} B_{i,n}(u) \cdot B_{j,m}(v) \text{ trong đó } u, v \in [0, 1]$$

Mảnh Bezier bậc ba:

Mặt cong Bezier bậc ba là mặt phổ biến nhất trong CG, vì độ đơn giản của nó. Nó hình thành trên 4x4 điểm kiểm soát, công thức có dạng:

$$Q(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 B_{n,i}(u) B_{m,j}(v) P_{ij}$$

Đa thức Bernstein có dạng:

$$B_0(t) = (1-t)^3$$

$$B_1(t) = 3t(1-t)^2$$

$$B_2(t) = 3t^2(1-t)$$

$$B_3(t) = t^3$$

Tính chất của mảnh Bézier

- Tính bao lồi: Mặt cong Bezier luôn nằm trong đa diện lồi của các điểm kiểm soát
- Mặt cong đi qua 4 điểm cận $P_{00}, P_{01}, P_{10}, P_{11}$ hay chính xác
 $Q(0,0)=P_{00}, Q(0,1)=P_{01}, Q(1,0)=P_{10}, Q(1,1)=P_{11}$
- Đường cong biên của Mặt Bezier là đường cong Bezier
- Mặt cong là liên tục và đạo hàm riêng các bậc tồn tại của nó cũng liên tục.
- Đạo hàm riêng của mặt cong có dạng:

$$Q(u, v) = [U] [N] [B] [M]^T [V]^T$$

$$\partial Q(0,0) / \partial u = 3(P_{01} - P_{00})$$

$$\partial Q(0,0) / \partial v = 3(P_{01} - P_{00})$$

$$\partial Q(0,1) / \partial u = 3(P_{03} - P_{02})$$

$$\partial Q(1,0) / \partial v = 3(P_{13} - P_{03})$$

$Q(u, v)$ là mọi điểm nằm trên mặt cong và

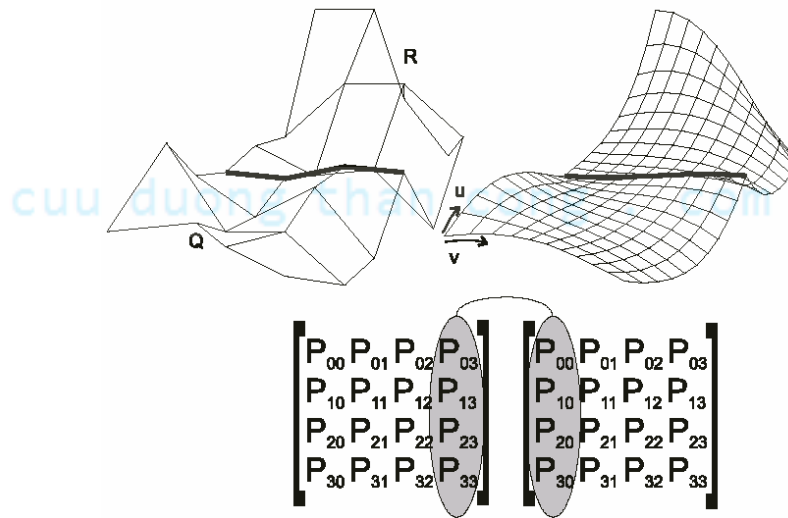
$$[V] = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}$$

$$[U] = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$[N] \text{ và } [M] \text{ được biểu diễn} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Nối 2 miếng Bezier Bậc 3(Bi-cubic)

$$Q(u, v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_{00} & B_{01} & B_{02} & B_{03} \\ B_{10} & B_{11} & B_{12} & B_{13} \\ B_{20} & B_{21} & B_{22} & B_{23} \\ B_{30} & B_{31} & B_{32} & B_{33} \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$



Hình 7.18 Nối hai mảnh Bezier bậc ba

Hai mảnh Q và R cùng chung tham biến tại biên (Giả sử u), hai đường cong biên phải bằng nhau $Q(1, v) = R(0, v)$. Hệ số của cột cuối ma trận Q = cột đầu ma trận R , tương tự: nếu theo hướng của v thì hàng sẽ thay cột ma trận.

Bậc của mặt cong theo mỗi hướng của tham biến bằng số điểm kiểm soát trừ 1. Tính liên tục hay đạo hàm của mặt theo mỗi tham biến bằng số điểm kiểm soát trừ 2. Hình dạng của mặt biến đổi theo các cạnh của đa giác kiểm soát. Mặt lưới chỉ đi qua các điểm góc cạnh của đa giác kiểm soát, nó chỉ nằm trong phần giới hạn bởi lưới của đa giác lõi kiểm soát và không thay đổi dưới tác động của các phép biến đổi affine. Mỗi đường biên của mặt Bezier là một đường cong Bezier với mặt cong bậc ba Bezier các đường cong biên luôn đảm bảo là các đường Bezier bậc 3. Như vậy lưới đa giác cho bề mặt sẽ là 4×4 .

2.4.3. Mặt cong B-Spline

Phương trình mặt B-spline:

$$Q(u, w) = \sum_{i=1}^n \sum_{j=1}^m N_{i,k}(u) \cdot M_{j,h}(w) \cdot P_{i,j}$$

$$N_{i,k}(u) = \begin{cases} 1 & x_i \leq u < x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,k}(u) = \frac{(u - x_i)N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u)N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}}$$

$$\begin{cases} x_i = 0 (1 \leq i \leq k) \\ x_i = i - k (k + 1 \leq i \leq n) \\ x_i = n - k + 1 (n + 1 \leq i \leq n + k) \end{cases}$$

P_{ij} là điểm kiểm soát

N và M là đa thức B-spline

Với các mặt cong mở mặt cong phụ thuộc vào các nút vector

Đặc điểm của mặt cong B-Spline

- Số bậc cao nhất của bề mặt theo mỗi hướng thì bằng số điểm kiểm soát -1 theo hướng đó.
- Đạo hàm riêng của phương trình bề mặt theo mỗi tham biến có bậc bằng số điểm kiểm soát theo tham biến đó trừ 2.
- Bề mặt B-spline thì không chịu ảnh hưởng của phép biến đổi affine. Bề mặt sẽ thay đổi nếu ta thay đổi đa giác kiểm soát.
- Ảnh hưởng của một điểm kiểm soát đơn được giới hạn bởi $\pm k/2 h/2$ khoảng đối với mỗi tham số.
- Nếu số đỉnh của đa giác kiểm soát bằng số bậc theo mỗi tham biến và không có điểm kép nào thì mặt B-spline sẽ chuyển thành mặt Bezier.
- Nếu các đa giác kiểm soát có dạng tam giác thì lưới đa giác kiểm soát sẽ có hình dáng gần giống với bề mặt cong.
- Mỗi mặt B-Spline luôn nằm trong bao lồi của đa giác kiểm soát.
- Mỗi mặt B-Spline có dáng điệu luôn bám theo hình dáng của đa giác kiểm soát.

Tóm tắt:

Việc tạo ra các đường cong theo ý muốn cũng là vấn đề thường gặp khi làm việc với đồ họa máy tính. Chúng ta khảo sát cách tiếp cận vẽ đường cong bằng Hermite, Bezier và B-spline. Các cách tiếp cận này dựa trên cơ sở vẽ đường cong bằng một tập điểm mô tả hình dáng của đường cong gọi là tập điểm kiểm soát. Khi thay đổi tập điểm này, hình dáng của đường cong sẽ thay đổi theo. Cách tiếp cận này cho thấy sự thuận lợi và linh hoạt khi cần phải vẽ các đường cong phức tạp và do đó nó được dùng nhiều trong thiết kế.

Một nhược điểm trong cách vẽ đường cong bằng Bezier là khi một phần của đường cong đã đạt yêu cầu, nhưng khi hiệu chỉnh phần còn lại sẽ mất đi phần đã đạt yêu cầu, hay việc nối tron

các đường cong sẵn có. Để khắc phục các vấn đề này ta có cách tiếp cận cải tiến vẽ đường cong bằng B-spline.

Tương tự như vậy việc biểu diễn các mặt cong trong đồ họa máy tính cũng là một vấn đề cần thiết để mô tả đối tượng trong thế giới thực. Chúng ta khảo sát về các phương pháp biểu diễn mặt cong thông qua phương trình tham số. Trong đó, phương trình tham số của một mặt có dạng là một phương trình tham số hai biến $p(u,v)$ và một điểm bất kỳ trên mặt sẽ được biểu diễn dưới dạng $p(u,v) = (x(u,v), y(u,v), z(u,v))$. Chúng ta khảo sát một số mặt đơn giản như: mặt kẻ, mặt tròn xoay, mặt trượt và mặt Boolean Sum.

Trên cơ sở các đường cong bằng Hermite, Bezier và B-spline chúng ta cũng xây dựng được các mặt Hermite, Bezier và B-spline.

Bài tập:

1. Cho 4 điểm $P_0(1,3,6)$, $P_1(6,0,3)$, $P_2(-1,3,-2)$ và $P_3(5,4,1)$ dùng 4 điểm trên làm điểm kiểm soát đưa ra phương trình của đường cong Bezier.
2. Cho hai điểm $P_1[-2 \ 4 \ 5]$ và $P_2[0 \ 1 \ -6]$ nằm trong mặt phẳng xoy. Quay đường thẳng quanh trục ox sẽ được một mặt nón. Xác định điểm của mặt tại $t=0.8$, $\phi=\pi/4$.
3. Cài đặt thuật toán vẽ đường cong bằng Bezier cho phép người dùng định nghĩa tập điểm kiểm soát mô tả hình dạng đường cong và cho phép người dùng hiệu chỉnh một số điểm kiểm soát mô tả hình dạng đường cong và cho phép người dùng hiệu chỉnh một số điểm kiểm soát để hiệu chỉnh đường cong theo ý muốn.
4. Viết phương trình vẽ đường Spline.
5. Viết chương trình sinh đường C_curve (Fractal).
6. Viết chương trình sinh đường Dragons (Fractal).
7. Viết chương trình sinh đường Kon (Fractal).
8. Viết chương trình sinh đường Mandelbrot (Fractal).
9. Viết chương trình sinh đường Pythagoras (Fractal).

cuu duong than cong . com

PHỤ LỤC 1

Hướng dẫn sử dụng thư viện đồ hoạ trong C/C++ hay BC

1. Yêu cầu

Phải có tập tin điều khiển màn hình EGA/VGA.BGI (thông thường tệp này thường nằm trong thư mục \BC\BGI hay TC\BGI khi cài đặt).

Nếu chúng ta có sử dụng tới font chữ thì cần phải có thêm các file (*.CHR) như: GOTH.CHR (chữ Gothic), LITT.CHR (chữ Small Font), SANS.CHR (chữ Sans Serif), TRIP.CHR (chữ cao gấp 3).

Để dùng được thư viện các hàm đồ hoạ cần có dòng lệnh:

#include <graphics.h> và đặt mục chọn Graphics library là ON ([x] trong menu Options/Linker/Libraries.

Khi cần tham khảo cú pháp, cách sử dụng của bất kỳ một hàm đồ hoạ nào, đưa con trỏ về tên hàm trong chương trình sau đó nhấn tổ hợp phím CTRL+F1. Muốn tham khảo danh sách toàn bộ các hàm của thư viện đồ hoạ nhấn tổ hợp phím CTRL+F1 ngay tại dòng chữ <graphics.h>.

2. Khởi tạo và đóng chế độ đồ hoạ

Độ phân giải của màn hình được đo bằng số điểm theo chiều ngang nhân với số điểm theo chiều dọc của màn hình đồ hoạ. Toạ độ gốc của màn hình đồ hoạ (0,0) là điểm nằm tại góc trên cùng phía bên trái. Mỗi kiểu đồ hoạ dùng một hệ toạ độ riêng. Hệ toạ độ cho màn hình VGA là 640x480.

Khởi động đồ hoạ với màn hình ngầm định:

```
#include <graphics.h>
void main(void){
int    gdriver, gmode, errorcode;
    gdriver = DETECT;//ngầm định
    initgraph(&gdriver,&gmode,"C:\\TC\\BGI"); //tìm mode màn hình trong thư mục BGI
    errorcode = graphresult();
    if (errorcode !=grOk) {
        printf("\n Không khởi tạo được");
        getch();exit(1);
    }
    .....// Các thao tác đồ hoạ tiếp theo
    closegraph();
}
```

Ví dụ: viết chương trình hai đường thẳng cắt nhau

```
#include <conio.h>
#include <graphics.h>
#include <math.h>
// hệ số đổi từ độ sang radian
```

```

#define RADS 0.017453293
void giaodiem(double x1, double y1, double x2, double y2,
double a1, double b1, double a2, double b2) {
    double dx, dy, da, db, x, y, t, tich;
    dx = x2 - x1;
    dy = y2 - y1;
    da = a2 - a1;
    db = b2 - b1;
    tich = db * dx - da * dy;
    if (tich != 0) {
        t = ((a1 - x1) * dy - (b1 - y1) * dx) / tich;
        if (t >= 0.0 && t <= 1.0) {
            x = t * (a2 - a1) + a1;
            y = t * (b2 - b1) + b1;
            line(x1, y1, x2, y2);
            line(a1, b1, a2, b2);
            setfillstyle(SOLID_FILL, RED);
            fillellipse(x, y, 3, 3);
        }
    }
}

void main() {
    int gr_drive = DETECT, gr_mode;
    double x1, y1, x2, y2, a1, b1, a2, b2;
    printf("\nNhap vao toa do doan thang thu nhât: ");
    scanf("%lf%lf%lf%lf", &x1, &y1, &x2, &y2);
    printf("\nNhap vao toa do doan thang thu hai: ");
    scanf("%lf%lf%lf%lf", &a1, &b1, &a2, &b2);
    initgraph(&gr_drive, &gr_mode, "");
    giaodiem(x1, y1, x2, y2, a1, b1, a2, b2);
    getch();
    closegraph(); //đóng chế độ đồ hoạ
}

```

3. Các hàm cơ bản

3.1. Bảng màu của màn hình đồ hoạ.

Setbkcolor(int color)	thiết lập màu nền cho màn hình đồ hoạ
setcolor(int color)	đặt màu vẽ cho nét vẽ hiện tại
getmaxcolor()	lấy số màu cao nhất được sử dụng trong chế độ đồ hoạ hiện tại
setallpalette(struct palettetype far *palette)	sẽ làm thay đổi toàn bộ màu trong bảng màu palette
setpalette(int colox, int colory)	sẽ làm thay đổi màu thứ colox thành màu colory
getpalette(struct palettetype far *palette)	sẽ lấy lại giá trị của palette đang dùng
textheight("W")	lấy chiều cao của dòng văn bản

outtextxy(int x, int y, char * msg)	Chỉ thị đưa xâu kí tự msg ra màn hình đồ hoạ tại vị trí (x,y)
-------------------------------------	---

3.2. Nguyên sơ điểm

Nguyên sơ đơn giản nhất chính là điểm, mọi nguyên sơ khác đều được xây dựng nên từ điểm.

int getmaxx()	Số chấm điểm lớn nhất theo chiều ngang trong chế độ đồ hoạ hiện tại
int getmaxy()	Số chấm điểm lớn nhất theo chiều dọc trong chế độ đồ hoạ hiện tại
putpixel(x, y, color)	Đề vẽ một điểm sáng lên màn hình tại điểm (x, y) có màu color
getpixel(x,y)	nhận biết màu hiện tại của điểm (x, y)

3.3. Nguyên sơ đường

Các hàm cho đường:

moveto(int x, int y)	di chuyển vị trí con trỏ hiện tại của màn hình đồ hoạ tới toạ độ (x,y)
getx(), gety()	lấy toạ độ của con trỏ hiện tại theo chiều ngang và chiều dọc màn hình đồ hoạ
lineto(int x, int y)	vẽ một đường thẳng từ vị trí con trỏ hiện tại tới vị trí có toạ độ (x, y) trên màn hình đồ hoạ
line(int x1, int y1, int x2, int y2)	Vẽ một đường thẳng từ toạ độ (x1,y1) đến toạ độ (x2, y2) trên màn hình đồ hoạ. Đường thẳng mới được vẽ không phụ thuộc vào vị trí hiện thời của con trỏ màn hình
linereel(int dx, int dy)	Vẽ một đoạn thẳng từ vị trí con trỏ hiện tại tới vị trí (x +dx, y+dy). Sau khi vẽ con trỏ chuyển tới vị trí mới (x+dx, y+dy)

3.4. Nguyên sơ hình chữ nhật

rectangle(int x1, int y1, int x2, int y2) vẽ hình chữ nhật có toạ độ phía góc trên bên trái là (x1, y1) và góc dưới bên phải có toạ độ (x2,y2).

bar(int x1, int y1, int x2, int y2) vẽ hình chữ nhật có tô màu phía trong. Hai chỉ thị rectangle và bar khác nhau ở chỗ rectangle chỉ tạo nên một hình chữ nhật với đường viền bao quanh.

bar3d(int x1,int y1,int x2,int y2,int depth, int top) vẽ khối hộp chữ nhật, mặt ngoài của nó là hình chữ nhật xác định bởi các toạ độ (x1,y1,x2,y2) hình chữ nhật này được tô màu, depth là chiều sâu của khối 3 chiều, top nhận giá trị 1 hoặc 0 để khối 3 chiều có nắp hay không có nắp.

3.5. Nguyên sơ hình tròn

Tất cả các hàm dưới đây, góc tính theo độ và giá trị từ 0^0 đến 360^0 .

arc(int x, int y, int gd,int gc,int r) vẽ cung tròn có (x,y) toạ độ tâm tròn, r bán kính, gd góc đầu, gc góc cuối.

circle(int x, int y, int r) là hàm vẽ hình tròn có tâm tại điểm (x,y) với bán kính r.

ellipse(int x, int y,int gd, int gc,int xr, int yr) vẽ 1 hình ellipse có (x,y) toạ độ tâm cung, gd góc đầu, gc góc cuối, xr là bán trục ngang, yr là bán trục đứng.

pieslice(int x, int y, int gd, int gc, int r) vẽ và tô màu hình quạt có (x,y) là toạ độ tâm quạt, gd là góc đầu, gc là góc cuối, r là bán kính.

3.6. Nguyên sơ đa giác

drawpoly(int numpoints, int far *polypoints) sẽ vẽ nên một đường gấp khúc bất kỳ với numpoints là số điểm mà đường gấp khúc đi qua, polypoints (mảng) toạ độ điểm (x1,y1,x2,y2....). Khi điểm cuối (xn,yn) trùng với điểm đầu (x1,y1) thì được một đa giác.

fillpoly(int numpoints, int *polypoints) sẽ tô màu đa giác bằng màu hiện thời.

setfillstyle(int pattern, int color) dùng để xác định mẫu tô cho đa giác, trong đó mẫu tô là các hằng số nguyên được định nghĩa như sau:

Tên hằng mẫu	Giá trị	Mô tả
EMPTY_FILL	0	Tô bằng màu nền
SOLID_FILL	1	Tô bằng nét liền
LINE_FILL	2	Tô -----
LTSLASH_FILL	3	Tô ////
SLASH_FILL	4	Tô ///// in đậm
BKSLASH_FILL	5	Tô \\\\\ in đậm
LTBKSLASH_FILL	6	Tô \\\\ . com
HATCH_FILL	7	Tô đường gạch bóng nhật
XHATCH_FILL	8	Tô đường gạch bóng chữ thập
INTERLEAVE_FILL	9	Tô đường đứt quãng
WIDE_DOT_FILL	10	Tô bằng dấu chấm thưa
CLOSE_DOT_FILL	11	Tô bằng dấu chấm dày

3.7. Nguyên sơ văn bản

outtext(char far *textstring) sẽ hiển thị nội dung xâu textstring tại vị trí hiện thời của màn hình đồ hoạ.

outtextxy(int x, int y, char far *textstring) hiển thị nội dung xâu textstring tại toạ độ (x, y) trong màn hình đồ hoạ.

settextstyle(int font, int direction, int charsize) dùng để xác lập kiểu chữ với các font chữ khác nhau.

Trong đó int font được xác lập thông qua các hằng sau:

TÊN FONT	Giá trị	Ý nghĩa
DEFAULT_FONT	0	Font 8x8 bit-mapped
TRIPLEX_FONT	1	Stroked triplex font

SMALL_FONT	2	Stroked small font
SANS_SERIF_FONT	3	Stroked sans-serif font
GOTHIC_FONT	4	Stroked gothic font

int direction được xác định nếu `HORIZ_DIR = 0` là nằm ngang từ trái qua phải, `VERT_DIR = 1` là thẳng đứng từ dưới lên trên.

int charsize nhận giá trị từ 1 đến 10 là hệ số phóng đại chữ.

`settextjustification(int hoz, int vert)` xác định vị trí dòng văn bản được đưa ra màn hình đồ hoạ bởi `outtext()` và `outtextxy()`.

Trong đó int hoz có thể nhận một trong các hằng sau:

`LEFT_TEXT = 0` văn bản xuất hiện phía bên trái con trỏ màn hình đồ hoạ

`CENTER_TEXT = 1` văn bản xuất hiện ở giữa với tâm là con trỏ màn hình đồ hoạ

`RIGHT_TEXT = 2` văn bản xuất hiện phía bên phải con trỏ màn hình đồ hoạ

Còn int vert là tham số có thể nhận các giá trị sau:

`BOOTTOM_TEXT = 0` văn bản xuất hiện ở phía trên con trỏ

`CENTER_TEXT = 1` văn bản xuất hiện ở quanh con trỏ

`TOP_TEXT = 2` văn bản xuất hiện ở phía dưới con trỏ

`textheight(char *s)` trả về chiều cao (theo pixel) của chuỗi do s trỏ tới. Với 8x8 bit map Font và hệ số khuếch đại chữ là 1 thì `textheight("H")=8`

`textwidth(char *s)` trả về chiều dài của chuỗi tính theo pixel.

3.8. Cửa sổ (viewport)

viewport là 1 vùng hình chữ nhật trên màn hình đồ hoạ giống như window trong textmode.

`setviewport(int x1, int y1, int x2, int y2, int clip)` trong đó (x1,y1,x2,y2) là góc trái trên và góc phải dưới thoả mãn điều kiện như sau:

$0 \leq x1 \leq x2$ và $0 \leq y1 \leq y2$

Tham số clip: clip=1 không cho phép vẽ ra ngoài viewport

clip=0 cho phép vẽ ra ngoài viewport

`getviewsettings(struct viewporttype *vp)` nhận viewport hiện hành,

```
struct viewporttype {
```

```
    int left,top,right,bottom;
```

```
    int clip;
```

```
};
```

`clearviewport(void)` xoá viewport

`cleardevice(void)` xoá mọi thứ trên màn hình và đưa con trỏ về toạ độ (0,0) của màn hình

Chú ý: nhờ sử dụng viewport ta có thể viết các chương trình đồ hoạ có trục toạ độ, bằng cách thiết lập viewport với clip=0 (cho phép vẽ ra ngoài giới hạn)

3.9. Tạo hình ảnh chuyển động

Nguyên sơ mới mà chúng ta sẽ xây dựng là tạo hình ảnh chuyển động trên màn hình đồ hoạ. Về nguyên tắc, để có thể tạo nên những hình ảnh chuyển động chúng ta cần có một hình mẫu, sau

đó lưu hình mẫu trên màn hình đồ hoạ lại bằng chỉ thị `getimage(int x1, int y1, int x2, int y2, void far *bitmap)`; trong đó `bitmap` là miền nhớ dùng để lưu hình ảnh của hình chữ nhật có toạ độ $(x1, y1)$ và $(x2, y2)$ trên màn hình đồ hoạ.

Chỉ thị `getimagesize(int x1, int y1, int x2, int y2)` dùng để xác định kích thước bộ nhớ dùng để cất hình ảnh giới hạn trong hình chữ nhật có toạ độ $(x1, y1)$, $(x2, y2)$.

Chỉ thị `putimage(int x, int y, void far * bitmap, int copymode)` dùng để khôi phục lại hình ảnh đã được cất giữ bằng `getimage()`. Chỉ thị `putimage()` kết hợp với chỉ thị làm trễ (`delay()`) sao cho số các hình ảnh hiển thị trên màn hình đồ hoạ khoảng 24 hình/giây chúng ta sẽ nhận được một hình ảnh chuyển động.

Ví dụ 1: Tạo hình ảnh chuyển động là một mũi tên.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#define ARROW_SIZE 10
void draw_arrow(int x, int y);
int main(void) {
    /* request autodetection */
    int gdriver = DETECT, gmode, errorcode;
    void *arrow;
    int x, y, maxx;
    unsigned int size;
    /* initialize graphics and local variables */
    initgraph(&gdriver, &gmode, "\\TC\\BGI");
    /* read result of initialization */
    errorcode = graphresult();
    if (errorcode != grOk) { /* an error occurred */
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1); /* terminate with an error code */
    }
    maxx = getmaxx(); x = 0;
    y = getmaxy() / 2;
    /* draw the image to be grabbed */
    draw_arrow(x, y);
    /* calculate the size of the image */
    size = imagesize(x, y-ARROW_SIZE, x+(4*ARROW_SIZE), y+ARROW_SIZE);
    /* allocate memory to hold the image */
    arrow = malloc(size);
    /* grab the image */
    getimage(x, y-ARROW_SIZE, x+(4*ARROW_SIZE), y+ARROW_SIZE, arrow);
    /* repeat until a key is pressed */
```

```
while (!kbhit()){
    /* erase old image */
    putimage(x, y-ARROW_SIZE, arrow, XOR_PUT);
    x += ARROW_SIZE;
    if (x >= maxx)
        x = 0;
    /* plot new image */
    putimage(x, y-ARROW_SIZE, arrow, XOR_PUT);
}
/* clean up */
free(arrow);
closegraph();
return 0;
}

void draw_arrow(int x, int y) {
    /* draw an arrow on the screen */
    moveto(x, y);
    linerel(4*ARROW_SIZE, 0);
    linerel(-2*ARROW_SIZE, -1*ARROW_SIZE);
    linerel(0, 2*ARROW_SIZE);
    linerel(2*ARROW_SIZE, -1*ARROW_SIZE);
}
}
```

Các code chương trình ví dụ cho bài tập lập trình

Bài 1: quay đối tượng

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<math.h>
#include<alloc.h>
#define RADS 0.017453293
struct point{
    int x,y,z;
}
a[4]=
{
    {110,180,80},
    {10,200,80},
    {120,50,80},
    {110,180,250}
};
int noi[4][4];
void Bresenham_Line(int x1,int y1, int x2, int y2, int c)
{

```

```
if(x1 == x2)
{
    int i;
    if(y1 < y2)
        for(i = y1; i <= y2; i++)
        {
            putpixel(x1, i, c);    delay(10);
        }
    else
        for(i = y2; i <= y1; i++)
        {
            putpixel(x1, i, c);    delay(10);
        }
}
if(y1 == y2)
{
    int i;
    if(x1 < x2)
        for(i = x1; i <= x2; i++)
        {
            putpixel(i, y1, c); delay(10);
        }
    else
        for(i = x2; i <= x1; i++)
        {
            putpixel(i, y1, c);    delay(10);
        }
}
if(x1 < x2)
{
    if(y1 < y2)
    {
        if((y2 - y1)/(x2 - x1) < 1)
        {
            int i;
            int Dx = x2 - x1;
            int Dy = y2 - y1;
            int p = 2*Dy - Dx;
            putpixel(x1, y1, c);
            for(i = x1; i < x2; i++)
            {
                if(p < 0)
                    p += 2*Dy;
                else
                {
                    p += 2*(Dy - Dx);
                    y1++;
                }
                x1++;
                putpixel(x1, y1, c);    delay(10);
            }
        }
    }
}
```

```
    }
}
else
{
    int i;
    int Dx = x2 - x1;
    int Dy = y2 - y1;
    int p = 2*Dx - Dy;
    putpixel(x1,y1,c);
    for(i = y1; i < y2; i ++)
    {
        if(p < 0)
            p += 2*Dx;
        else
        {
            p += 2*(Dx - Dy);
            x1 ++;
        }
        y1 ++;
        putpixel(x1,y1,c);    delay(10);
    }
}
}
if(y1 > y2)
{ if((y2 - y1)/(x2 - x1) > -1 )
{
    int i;
    int Dx = x2 - x1;
    int Dy = y2 - y1;
    int p = 2*Dy + Dx;
    putpixel(x1,y1,c);
    for(i = x1; i <= x2; i ++)
    {
        if(p > 0)
            p += 2*Dy;
        else
        {
            p += 2*(Dy + Dx);
            y1--;
        }
        x1 ++;
        putpixel(x1,y1,c);    delay(10);
    }
}
}
else
{
    int i;
    int Dx = x2 - x1;
    int Dy = y2 - y1;
    int p = 2*Dx + Dy;
```

```
        putpixel(x1,y1,c);
        for(i = y1; i >= y2; i --)
        {
            if(p < 0)
                p += 2*Dx;
            else
            {
                p += 2*(Dx + Dy);
                x1++;
            }
            y1 --;
            putpixel(x1,y1,c);    delay(10);
        }
    }
}
if(x1 > x2)
{
    if(y1 < y2)
    {
        if((y2-y1)/(x2-x1) > -1)
        {
            int i;
            int Dx = x2 - x1;
            int Dy = y2 - y1;
            int p = -2*Dy - Dx;
            putpixel(x1,y1,c);
            for(i = x1; i > x2; i --)
            {
                if(p < 0)
                {
                    p=p - 2*Dy - 2*Dx;
                    y1 ++;
                }
                else
                    p =p - 2*Dy;
                x1 --;
                putpixel(x1,y1,c);delay(10);
            }
        }
        else
        {
            int i;
            int Dx = x2 - x1;
            int Dy = y2 - y1;
            int p = -2*Dx - Dy;
            putpixel(x1,y1,c);
            for(i = y1; i < y2; i ++ )
            {
                if(p < 0)
                    p -= 2*Dx;
```

```
        else
        {
            p -= 2*(Dx + Dy);
            x1 --;
        }
        y1 ++;
        putpixel(x1,y1,c); delay(10);
    }
}
}
if(y1 > y2)
{
    if((y2-y1)/(x2-x1) < 1)
    {
        int i;
        int Dx = x2 - x1;
        int Dy = y2 - y1;
        int p = 2*Dy - Dx;
        putpixel(x1,y1,c);
        for(i = x1; i > x2; i --)
        {
            if(p > 0)
            {
                p = p - 2*Dy + 2*Dx ;
                y1--;
            }
            else
            {
                p -= 2*Dy;
            }

            x1 --;
            putpixel(x1,y1,c); delay(10);
        }
    }
    else
    {
        int i;
        int Dx = x1 - x2;
        int Dy = y1 - y2;
        int p = 2*Dx + Dy;
        putpixel(x1,y1,c);
        for(i = y1; i > y2; i --)
        {
            if(p < 0)
            {
                p = p - 2*Dx + 2*Dy ;
                x1 --;
            }
            else
            {
                p += 2*Dx;
            }

            y1 --;
            putpixel(x1,y1,c); delay(10);
        }
    }
}
```



```

        p = p - 2*Dx ;
        y1 --;
        putpixel(x1,y1,c);        delay(10);
    }
}

}

}

}

void Dothi()
{
    Bresenham_Line(getmaxx()/2,getmaxy()/2,getmaxx()-10,getmaxy()/2,15);
    Bresenham_Line(getmaxx()/2,10,getmaxx()/2,getmaxy()/2,15);
    Bresenham_Line(getmaxx()/2,getmaxy()/2,getmaxx()/2-170,getmaxy()/2+170,15);
    Bresenham_Line(getmaxx()/2-170,getmaxy()/2+170,getmaxx()/2-170,getmaxy()/2+166,15);
    Bresenham_Line(getmaxx()/2-170,getmaxy()/2+170,getmaxx()/2-163,getmaxy()/2+167,15);
    settxtjustify(CENTER_TEXT,CENTER_TEXT);
    outtextxy(getmaxx()-9,getmaxy()/2+1,"□");
    outtextxy(getmaxx()-9,getmaxy()/2+10,"y");
    outtextxy(getmaxx()/2,10,"-");
    outtextxy(getmaxx()/2 - 10,10,"z");
    outtextxy(getmaxx()/2-155,getmaxy()/2+170,"x");
}

point Diem3d(int x , int y , int z)
{
    point p;
    if(x>=0&&y>=0&&z>=0)
    {
        p.x = int(getmaxx()/2+y - x*cos(RADS*45));
        p.y = int(getmaxy()/2-z + x*cos(RADS*45));
    }
    if(y>=0&&x<0&&z>=0)
    {
        p.x = int(y+getmaxx()/2-x*cos(RADS*45));
        p.y = int(getmaxy()/2-z+x*cos(RADS*45));
    }
    if(x>=0&&y<0&&z>=0)
    {
        p.x = int(getmaxx()/2+y-x*cos(RADS*45));
        p.y = int(getmaxy()/2-(z-x*cos(RADS*45)));
    }
    if(x>=0&&y>=0&&z<0)
    {
        p.x = int(getmaxx()/2+y-x*cos(RADS*45));
        p.y = getmaxy()/2-z+x*cos(RADS*45);
    }
    if(y>=0&&x<0&&z<0)
    {
        p.x = int(getmaxx()/2+y-x*cos(RADS*45));
        p.y = int(getmaxy()/2+(-z-x*cos(RADS*45)));
    }
}

```

```
}
if(x>=0&&y<0&&z<0)
{
    p.x = int(getmaxx()/2+y-x*cos(RADS*45));
    p.y = int(getmaxy()/2-z+x*cos(RADS*45));
}
if(z>=0&&y<0&&x<0)
{
    p.x = int(getmaxx()/2-(-y+x*cos(RADS*45)));
    p.y = int(getmaxy()/2-(z-x*cos(RADS*45)));
}
if(x<0&&y<0&&z<0)
{
    p.x = int(getmaxx()/2+y-x*cos(RADS*45));
    p.y = int(getmaxy()/2-z+x*cos(RADS*45));
}
return(p);
}
point hinhchieu(int &x, int &y , int &z , int chieu)
{
    point p;
    if(chieu==1)
    {
        p.x =int( getmaxx()/2+y - x*cos(RADS*45));
        p.y =int( getmaxy()/2 + x*cos(RADS*45));
    }
    if(chieu==2)
    {
        p.x =int ( getmaxx()/2+y-x*cos(RADS*45)-y);
        p.y =int ( getmaxy()/2-z+x*cos(RADS*45));
    }
    if(chieu == 3)
    {
        p.x =int( getmaxx()/2+y-x*cos(RADS*45)+x*sin(RADS*45));
        p.y =int( getmaxy()/2-z);
    }
    return(p);
}
point quay(int &x, int &y, int &z, int goc , int chieu)
{
    point p;
    if(chieu==1)
    {
        p.x = x*cos(RADS*goc) + z*sin(RADS*goc);
        p.z = -x*sin(RADS*goc) + z * cos(RADS*goc);
        p.y = y;
    }
    if(chieu==2)
    {
        p.y = y*cos(RADS*goc) - z*sin(RADS*goc);
```

```
p.z = y*sin(RADS*goc) + z * cos(RADS*goc);
p.x =x;
}
if(chieu==3)
{
p.x = x*cos(RADS*goc)-y*sin(RADS*goc);
p.y = x*sin(RADS*goc)+y*cos(RADS*goc);
p.z = z;
}
return p;
}
void chieumat(point k[],int t, int m )
{
int i,j=0;
int n[4][4];
for(i = 0 ; i<4 ; i++)
for(j = 0 ; j<4 ; j++)
{
n[i][j]=1;
n[j][i]=1;
}
for(i = 0 ; i< 4 ; i++)
{
k[i] = hinhchieu(k[i].x,k[i].y,k[i].z,t);
}
for( i = 0 ; i< 4 ; i++)
for( j = 0 ; j< 4 ; j++)
{
if(i!=j&& n[i][j]==1&&n[j][i]==1)
{
Bresenham_Line(k[i].x,k[i].y,k[j].x,k[j].y,m);
n[i][j]=0;
n[j][i]=0;
}
}
}
void main()
{
clrscr();
point b[4],c[4],l[4];
int j=0;
int goc[2]={45,35};
int driver = DETECT, mode;
initgraph(&driver,&mode,"c:\\tc\\bgi");
Dothi();
for(int i = 0; i < 4; i++)
b[i] = Diem3d(a[i].x,a[i].y,a[i].z);
for( i = 1; i < 4; i++)
{
Bresenham_Line(b[j].x,b[j].y,b[i].x,b[i].y,10);
```

```
}
Bresenham_Line(b[1].x,b[1].y,b[2].x,b[2].y,10);
Bresenham_Line(b[1].x,b[1].y,b[3].x,b[3].y,10);
Bresenham_Line(b[2].x,b[2].y,b[3].x,b[3].y,10);
settextjustify(LEFT_TEXT,RIGHT_TEXT);
printf("\nQuay quanh trục : oy góc 45");
printf("\nQuay quanh trục : ox góc 35.5");
for( i = 0 ; i < 4 ; i++)
{
c[i] = quay(a[i].x , a[i].y , a[i].z , goc[1] ,1);
}
for(i = 0 ; i < 4 ; i++)
{
c[i] = quay(c[i].x,c[i].y,c[i].z,goc[2],2);
}
for(i = 0; i < 4; i++)
l[i] = c[i];
for( i = 0; i < 4; i++)
c[i] = Diem3d(c[i].x,c[i].y,c[i].z);
outtextxy(10,110,"Chon mat chieu: ");
outtextxy(10,120,"xoy: Bam so 1.");
outtextxy(10,130,"xoz: Bam so 2.");
outtextxy(10,140,"yoz: Bam so 3.");
outtextxy(10,150,"Bam so 4 de ket thuc");
char m;
do{
m = getch();
for(i = 0;i < 4; i++)
c[i] = l[i];
switch (m)
{
case '1' :{
chieumat(c,1,5);
break;
}
case '2' :{
chieumat(c,2,7);
break;
}
case '3' :
{
chieumat(c,3,9) ;
break;
}
}
}while( m !='4');
closegraph();
}
```

Bài 2: xén tĩa

```
#include <conio.h>
```

```
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#define ROUND(a) ((double)(a+0.5))
#define TRUE 1
#define FALSE 0
int cliptest(double p, double q, double *u1, double *u2)
{
    double r;
    int retVal = TRUE;
    if (p < 0.0)
    {
        r = q / p;
        if (r > *u2)
            retVal = FALSE;
        else
            if (r > *u1)
                *u1 = r;
    }
    else
        if (p > 0.0)
        {
            r = q / p;
            if (r < *u1)
                retVal = FALSE;
            else
                if (r < *u2)
                    *u2 = r;
        }
        else
        {
            if (q < 0.0)
                retVal = FALSE;
        }
    return (retVal);
}

void clipline(double x1, double y1, double x2, double y2,
              double xmin, double ymin, double xmax, double ymax)
{
    double u1 = 0.0, u2 = 1.0, dx, dy;
    double oldx1, oldy1, oldx2, oldy2;
    oldx1 = x1;
    oldy1 = y1;
    oldx2 = x2;
    oldy2 = y2;
    dx = x2 - x1;
    if (cliptest(-dx, x1 - xmin, &u1, &u2))
        if (cliptest(dx, xmax-x1, &u1, &u2))
        {
            dy = y2 - y1;
```

```
if (cliptest(-dy, y1 - ymin, &u1, &u2))
    if (cliptest(dy, ymax - y1, &u1, &u2))
    {
        if (u2 < 1.0)
        {
            x2 = x1 + u2 * dx;
            y2 = y1 + u2 * dy;
        }
        if (u1 > 0.0)
        {
            x1 += u1 * dx;
            y1 += u1 * dy;
        }
        setcolor(RED);
        line(oldx1, oldy1, x1, y1);
        line(x2, y2, oldx2, oldy2);
        setcolor(YELLOW);
        line(x1, y1, x2, y2);
    }
}
}
void main()
{
    int gr_drive = DETECT, gr_mode;
    char c;
    double x1, y1, x2, y2;
    initgraph(&gr_drive, &gr_mode, "");
    rectangle(100,50, getmaxx()-100, getmaxy()-50);
    randomize();
    outtextxy(100, getmaxy()-10, "Nhấn phím bất kỳ để sinh đường khác; ESC để thoát");
    do {
        x1 = random(getmaxx() / 2);
        y1 = random(getmaxy());
        x2 = getmaxx()/2 + random(getmaxx() / 2);
        y2 = random(getmaxy());
        clipline(x1, y1, x2, y2, 100, 50, getmaxx()-100, getmaxy()-50);
        c = getch();
    } while (c != 27);
    closegraph();
}
```

Bài 3: Phép chiếu

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
#include<math.h>
#include<alloc.h>
#define RADS 0.017453293
struct point{
```

```
int x,y,z;
}
a[8]=
{
    {30,100,50},
    {30,230,50},
    {30,230,90},
    {30,100,90},
    {120,100,50},
    {120,230,50},
    {120,230,90},
    {120,100,90}
};

int n = 3,*d;
int noi[20][20];
void Dothi()
{
    line (getmaxx()/2,getmaxy()/2,getmaxx()-10,getmaxy()/2);
    line (getmaxx()/2,10,getmaxx()/2,getmaxy()/2);
    line (getmaxx()/2,getmaxy()/2,getmaxx()/2-170,getmaxy()/2+170);
    line (getmaxx()/2-170,getmaxy()/2+170,getmaxx()/2-170,getmaxy()/2+166);
    line (getmaxx()/2-170,getmaxy()/2+170,getmaxx()/2-163,getmaxy()/2+167);
    settextrjust(CENTER_TEXT,CENTER_TEXT);
    outtextxy(getmaxx()-9,getmaxy()/2+1,"0");
    outtextxy(getmaxx()-9,getmaxy()/2+10,"y");
    outtextxy(getmaxx()/2,10,"-");
    outtextxy(getmaxx()/2 - 10,10,"z");
    outtextxy(getmaxx()/2-155,getmaxy()/2+170,"x");
}
point Diem3d(int &x, int &y, int &z)
{
    point p;
    p.x = int(getmaxx()/2+ y - x*cos(RADS*45));
    p.y = int(getmaxy()/2 - z + x*cos(RADS*45));
    return p;
}
point chance(int &x , int &y , int &z)
{
    point p;
    if(x>=0&&y>=0&&z>=0)
    {
        p.x = getmaxx()/2+y - x*cos(RADS*45);
        p.y = getmaxy()/2-z + x*cos(RADS*45);
    }
    if(y>=0&&x<0&&z>=0)
    {
        p.x = y+getmaxx()/2-x*cos(RADS*45);
        p.y = getmaxy()/2+z-x*cos(RADS*45);
    }
}
```

```
if(x>=0&&y<0&&z>=0)
{
    p.x = getmaxx()/2+y-x*cos(RADS*45);
    p.y = getmaxy()/2-(z-x*cos(RADS*45));
}
if(x>=0&&y>=0&&z<0)
{
    p.x = getmaxx()/2+(sqrt(pow(x,2)+pow(y,2)))*cos(RADS*45);
    p.y = getmaxy()/2+(sqrt(pow(x,2)+pow(y,2)))*cos(RADS*45)-z;
}
if(y>=0&&x<0&&z<0)
{
    p.x = getmaxx()/2+y-x*cos(RADS*45);
    p.y = getmaxy()/2+(-z-x*cos(RADS*45));
}
return(p);
}
point hinhchieu(int &x, int &y , int &z , int chieu)
{
    point p;
    if(chieu==1)
    {
        p.x =int( getmaxx()/2+y - x*cos(RADS*45));
        p.y =int( getmaxy()/2+x*cos(RADS*45));
    }
    if(chieu==2)
    {
        p.x =int ( getmaxx()/2+y-x*cos(RADS*45)-y);
        p.y =int ( getmaxy()/2-z+x*cos(RADS*45));
    }
    if(chieu == 3)
    {
        p.x =int( getmaxx()/2+y-x*cos(RADS*45)+x*sin(RADS*45));
        p.y =int( getmaxy()/2-z);
    }
    return(p);
}
point quay(int &x, int &y, int &z, int goc , int chieu)
{
    point p;
    if(chieu==1)
    {
        p.x = x*cos(RADS*goc) + z*sin(RADS*goc);
        p.z = -x*sin(RADS*goc) + z * cos(RADS*goc);
        p.y = y;
    }
    if(chieu==2)
    {
        p.y = y*cos(RADS*goc) - z*sin(RADS*goc);
        p.z = y*sin(RADS*goc) + z * cos(RADS*goc);
    }
}
```



```
        p.x =x;
    }
    if(chieu==3)
    {
        p.x = x*cos(RADS*goc)-y*sin(RADS*goc);
        p.y = x*sin(RADS*goc)+y*cos(RADS*goc);
        p.z = z;
    }
    return p;
}
void chieumat(point k[],int t, int &m )
{
    int i,j;
    for(i = 0 ; i< 8 ; i++)
        k[i] = hinhchieu(k[i].x,k[i].y,k[i].z,t);
    for( i = 0 ; i< 8 ; i++)
        for( j = 0 ; j< 8 ; j++)
        {
            if(noi[i][j]==1&&noi[j][i]==1)
            {
                Bresenham_Line(k[i].x,k[i].y,k[j].x,k[j].y,m);
            }
        }
    noi[0][3]=1,noi[3][0]=1;
    noi[4][7]=1,noi[7][4]=1;
    noi[3][7]=1,noi[7][3]=1;
}
void main()
{
    clrscr();
    d = &n;
    point b[8],g[3][8],c[8],l[8];
    int t, goc ;
    int driver = DETECT, mode;
    initgraph(&driver,&mode,"c:\\tc\\bgi");
    Dothi();
    for(int i = 0; i < 8; i++)
        b[i] = Diem3d(a[i].x,a[i].y,a[i].z);
    for( i = 0; i < 3; i++)
    {
        line(b[i].x,b[i].y,b[i+4].x,b[i+4].y);
        line(b[i].x,b[i].y,b[i+1].x,b[i+1].y);
        line(b[i+4].x,b[i+4].y,b[i+5].x,b[i+5].y);
        noi[i][i+4]=1 , noi[i+4][i] =1;
        noi[i][i+1] = 1, noi[i+1][i]=1;
        noi[i+4][i+5]=1, noi[i+5][i+4]=1;
    }
    noi[0][3]=1,noi[3][0]=1;
    noi[4][7]=1,noi[7][4]=1;
    noi[3][7]=1,noi[7][3]=1;
```

```
line(b[0].x,b[0].y,b[3].x,b[3].y);
line(b[4].x,b[4].y,b[7].x,b[7].y);
line(b[3].x,b[3].y,b[7].x,b[7].y);
settextjustify(LEFT_TEXT,RIGHT_TEXT);
printf("\nQuay quanh trục :");
printf("\ny.Bam so 1");
printf("\nx.Bam so 2.");
printf("\nz.Bam so 3.");
scanf("%d",&t);
printf("\nQuay goc bao nhieu do:");
scanf("%d",&goc);
for( i = 0 ; i<= 7 ; i++)
{
c[i] = quay(a[i].x , a[i].y , a[i].z , goc ,t);
l[i] = c[i];
}
for( i = 0; i < 8; i ++ )
c[i] = Diem3d(c[i].x,c[i].y,c[i].z);
for( i = 0; i < 3; i ++ )
{
line(c[i].x,c[i].y,c[i + 4].x,c[i + 4].y);
line(c[i].x,c[i].y,c[i+1].x,c[i+1].y);
line(c[i+4].x,c[i+4].y,c[i+5].x,c[i+5].y);
}
line(c[0].x,c[0].y,c[3].x,c[3].y);
line(c[4].x,c[4].y,c[7].x,c[7].y);
line(c[3].x,c[3].y,c[7].x,c[7].y);
outtextxy(10,410,"Chon mat chieu: ");
outtextxy(10,420,"xoy: Bam so 1.");
outtextxy(10,430,"xoz: Bam so 2.");
outtextxy(10,440,"yoz: Bam so 3.");
outtextxy(10,450,"Bam so 4 de ket thuc");
char m;
do{
m = getch();
for(i = 0;i < 8; i ++ )
c[i]= l[i];
switch (m)
{
case '1' :{
chieumat(c,1,*(&d));
break;
}
case '2' :{
chieumat(c,2,*(&d));
break;
}
case '3' :
{
chieumat(c,3,*(&d)) ;
}
```

```
                break;
            }
        }while( m !='4');
    closegraph();
}
```

cuu duong than cong . com

cuu duong than cong . com

PHỤ LỤC 2

Đề bài điều kiện 1

MÔN: KỸ THUẬT ĐỒ HOẠ

Họ và tên:

Lớp:

1. Dùng giải thuật Bresenham viết hàm sinh đoạn thẳng (xét tất cả trường hợp của hệ số k).
`void Bre_line(int x1, int y1, int x2, int y2, int mau);`
 2. Xây dựng thủ tục vẽ tam giác (trong hệ toạ độ có gốc (0,0) ở giữa màn hình) có các toạ độ đỉnh sau:
 $A(x_1, y_1) \dots (40, 50)$, $B(x_2, y_2) \dots (90, 80)$, $C(x_3, y_3) \dots (80, 10)$
 3. Quay tam giác trên một góc α quanh điểm có toạ độ:
 $S(x_4, y_4) \dots (-40, 50)$, $\alpha = 60^\circ$
 4. Chuyển động tam giác trên (xoá hình cũ khi chuyển động) khi dùng các phím mũi tên \leftarrow , \rightarrow , \uparrow , \downarrow
 5. Cho hình tứ diện có các toạ độ đỉnh:
 $A(x_1, y_1, z_1) \dots (40, 50, 10)$ $B(x_2, y_2, z_2) \dots (90, 80, 0)$
 $C(x_3, y_3, z_3) \dots (80, 10, 90)$ $D(x_4, y_4, z_4) \dots (100, 50, 60)$
 - a. Vẽ hình chiếu vuông góc của nó trên mặt phẳng $z=0$ (xoy)
 - b. Vẽ hình chiếu vuông góc sau khi quay nó xung quanh trục oy một góc $\varphi = -60^\circ$
 - c. Vẽ hình chiếu Isometric của nó lên mặt phẳng $z=0$ (xoy)
- Chú ý: Có thể nộp bài theo các phương án sau:
- a. Nộp các câu thành các file *.CPP
 - b. Gộp thành một bài, trong bài đó chọn các phương án để chạy các câu của bài thi
 - c. Có thể dán các câu (*.CPP) hay một bài lớn dạng menu chọn vào soạn thảo word, rồi gửi file đó.

Đề bài điều kiện 2

MÔN: KỸ THUẬT ĐỒ HOẠ

Họ và tên:

Lớp:

1. Dùng giải thuật Midpoint viết hàm sinh đoạn thẳng (xét tất cả các trường hợp của hệ số k).
`void Mid_line(int x1, int y1, int x2, int y2, int mau);`
2. Xây dựng thủ tục vẽ hình chữ nhật (trong hệ tọa độ có gốc (0,0) ở giữa màn hình) có các tọa độ đỉnh sau:
 $A(x_1, y_1) \dots (50, 50)$, $B(x_2, y_2) \dots (100, 50)$, $C(x_3, y_3) \dots (100, 80)$, $D(x_4, y_4) \dots (80, 50)$
3. Quay hình chữ nhật trên một góc α quanh điểm có tọa độ:
 $S(x_4, y_4) \dots (-40, 50)$, $\alpha = 45^\circ$
4. Chuyển động hình chữ nhật trên (xoá hình cũ khi chuyển động) khi dùng các phím mũi tên $\leftarrow, \rightarrow, \uparrow, \downarrow$
5. Cho hình tứ diện có các tọa độ đỉnh:
 $A(x_1, y_1, z_1) \dots (40, 50, 10)$ $B(x_2, y_2, z_2) \dots (90, 80, 0)$
 $C(x_3, y_3, z_3) \dots (80, 10, 90)$ $D(x_4, y_4, z_4) \dots (100, 50, 60)$
 - a. Vẽ hình chiếu vuông góc của nó trên mặt phẳng $z=0$ (xoy)
 - b. Vẽ hình chiếu vuông góc sau khi quay nó xung quanh trục oy một góc $\varphi = -60^\circ$
 - c. Vẽ hình chiếu Isometric của nó lên mặt phẳng $z=0$ (xoy)

cuu duong than cong . com

Đề bài điều kiện 3

MÔN: KỸ THUẬT ĐỒ HOẠ

Họ và tên:

Lớp:

1. Dùng giải thuật Midpoint sinh đường tròn.

`void Mid_circle(int xc, int yc, int r, int mau);`

2. Xây dựng thủ tục vẽ hình 3 đường tròn cắt nhau (trong hệ toạ độ có gốc (0,0) ở giữa màn hình).

3. Quay 3 đường tròn trên một góc α quanh điểm có toạ độ:

$S(x_4, y_4) \dots (60, -70), \alpha = 30^\circ$

4. Chuyển động 3 đường tròn trên (xoá hình cũ khi chuyển động) khi dùng các phím mũi tên $\leftarrow, \rightarrow, \uparrow, \downarrow$

5. Cho hình tứ diện có các toạ độ đỉnh:

$A(x_1, y_1, z_1) \dots (40, 50, 10)$

$B(x_2, y_2, z_2) \dots (90, 80, 0)$

$C(x_3, y_3, z_3) \dots (80, 10, 90)$

$D(x_4, y_4, z_4) \dots (100, 50, 60)$

- Vẽ hình chiếu vuông góc của nó trên mặt phẳng $z=0$ (xoy)
- Vẽ hình chiếu vuông góc sau khi quay nó xung quanh trục oy một góc $\varphi = -60^\circ$
- Vẽ hình chiếu Isometric của nó lên mặt phẳng $z=0$ (xoy)

Đề bài điều kiện 4

MÔN: KỸ THUẬT ĐỒ HOẠ

Họ và tên:

Lớp:

1. Dùng giải thuật Midpoint sinh đường ellipse.

`void Mid_ellip(int xc, int yc, int rx, int ry, int mau);`

2. Xây dựng thủ tục vẽ chữ H (trong hệ tọa độ có gốc (0,0) ở giữa màn hình)

3. Quay chữ H trên một góc alfa quanh điểm có tọa độ:

`S(x4,y4)(120,-60), alfa = 900`

4. Chuyển động chữ H trên (xóa hình cũ khi chuyển động) khi dùng các phím mũi tên ←, →, ↑, ↓

5. Cho hình tứ diện có các tọa độ đỉnh:

`A(x1,y1,z1)....(40,50,10)`

`B(x2,y2,z2)(90,80,0)`

`C(x3,y3,z3)(80,10,90)`

`D(x4,y4,z4)(100,50,60)`

a. Vẽ hình chiếu vuông góc của nó trên mặt phẳng $z=0$ (xoy)

b. Vẽ hình chiếu vuông góc sau khi quay nó xung quanh trục oy một góc $\phi=-60^0$

c. Vẽ hình chiếu Isometric của nó lên mặt phẳng $z=0$ (xoy)

cuu duong than cong . com

TÀI LIỆU THAM KHẢO

- [1] James D.Foley, Andrie van Dam, Steven K.Feiner, Jonhn F. Hughes, Computer Graphics Principles and Practice, Addison Wesley, 1994.
- [2] Hoàng Kiếm, Dương Anh Đức, Lê Đình Duy, Vũ Hải Quân. Giáo trình cơ sở Đồ hoạ Máy tính, NXB Giáo dục, 2000.
- [3] Lê Tấn Hùng, Huỳnh Quyết Thắng. Kỹ thuật đồ hoạ máy tính, NXB khoa học và kỹ thuật, 2002.
- [4] Steven Harrington, Computer Graphics A Programming Approach, McGraw Hill International Edition, 1987.
- [5] Gerald Farin, Curves and Surfaces for Computer Aided Geometric Design A Practical Guide, Academic Press Inc, 1990.

cuu duong than cong . com

cuu duong than cong . com

MỤC LỤC

LỜI NÓI ĐẦU.....	3
CHƯƠNG 1: TỔNG QUAN VỀ KỸ THUẬT ĐỒ HOẠ.....	4
1. CÁC KHÁI NIỆM TỔNG QUAN CỦA KỸ THUẬT ĐỒ HOẠ MÁY TÍNH (COMPUTER GRAPHICS).....	4
1.1. Lịch sử phát triển.....	4
1.2. Kỹ thuật đồ họa vi tính.....	5
2. CÁC KỸ THUẬT ĐỒ HOẠ.....	5
2.1. Kỹ thuật đồ họa điểm (Sample based-Graphics).....	5
2.2. Kỹ thuật đồ họa vector.....	6
2.3. Phân loại của đồ họa máy tính.....	8
2.4. Các ứng dụng tiêu biểu của kỹ thuật đồ họa.....	9
2.5. Các chuẩn giao diện của hệ đồ họa.....	11
3. PHẦN CỨNG ĐỒ HOẠ (GRAPHICS HARDWARE).....	11
3.1. Các thành phần phần cứng của hệ đồ họa tương tác.....	11
3.2. Máy in.....	12
3.3. Màn hình CRT.....	12
3.4. Màn hình tinh thể lỏng (Liquid Crystal Display – LCD).....	15
Tóm tắt chương:	16
Bài tập:.....	16
Bài tập trắc nghiệm:.....	17
CHƯƠNG 2: CÁC GIẢI THUẬT SINH THỰC THỂ CƠ SỞ.....	18
1. CÁC ĐỐI TƯỢNG ĐỒ HOẠ CƠ SỞ.....	18
1.1. Hệ tọa độ thế giới thực và hệ tọa độ thiết bị.....	18
1.2. Điểm và đoạn thẳng.....	18
2. CÁC GIẢI THUẬT XÂY DỰNG THỰC THỂ CƠ SỞ.....	19
2.1. Giải thuật vẽ đoạn thẳng thông thường.....	19
2.2. Thuật toán DDA (Digital Differential Analyzer).....	19
2.3. Giải thuật Bresenham.....	20
2.4. Giải thuật trung điểm-Midpoint.....	22
2.5. Giải thuật sinh đường tròn (Scan Converting Circles)(Bresenham).....	24
2.7. Giải thuật sinh đường tròn Midpoint.....	26
2.8. Giải thuật sinh đường ellipse.....	27
2.9. Giải thuật sinh ký tự.....	30

2.10. Giải thuật sinh đa giác (Polygon)	32
Tóm tắt chương:	36
Bài tập:	36
Bài tập trắc nghiệm:.....	37
CHƯƠNG 3: CÁC PHÉP BIẾN ĐỔI ĐỒ HOẠ	41
1. CÁC PHÉP BIẾN ĐỔI HÌNH HỌC HAI CHIỀU	41
1.1. Phép biến đổi Affine (Affine Transformations)	41
1.2. Các phép biến đổi đối tượng.....	41
2. TỌA ĐỘ ĐỒNG NHẤT VÀ CÁC PHÉP BIẾN ĐỔI	45
2.1. Tọa độ đồng nhất.....	45
2.2. Phép biến đổi với tọa độ đồng nhất	46
3. CÁC PHÉP BIẾN ĐỔI HÌNH HỌC BA CHIỀU	47
3.1. Biểu diễn điểm trong không gian 3 chiều.....	47
3.2. Phép tịnh tiến.....	47
3.3. Phép tỉ lệ.....	48
3.4. Phép biến dạng	48
3.5. Phép lấy đối xứng.....	48
3.6. Phép quay 3 chiều.....	49
Tóm tắt:	53
Bài tập:	54
Bài tập trắc nghiệm:.....	55
CHƯƠNG 4: CÁC GIẢI THUẬT ĐỒ HOẠ CƠ SỞ	59
1. HỆ TỌA ĐỘ VÀ MÔ HÌNH CHUYỂN ĐỔI.....	59
1.1. Các hệ thống tọa độ trong đồ họa.....	59
1.2. Phép ánh xạ từ cửa sổ vào cổng xem.....	61
2. CÁC GIẢI THUẬT XÉN TỈA (CLIPPING)	62
2.1. Khái niệm	62
2.2. Clipping điểm	63
2.3. Xén tỉa đoạn thẳng.....	63
2.4. Giải thuật xén tỉa đa giác (Sutherland Hodgman)	69
Tóm tắt chương:	73
Bài tập:	74
Bài tập trắc nghiệm:.....	74
CHƯƠNG 5: PHÉP CHIẾU –PROJECTION	79
1. KHÁI NIỆM CHUNG.....	79
1.1. Nguyên lý về 3D (three-Dimension)	79
1.2. Đặc điểm của kỹ thuật đồ họa 3D	79

1.3.Các phương pháp hiển thị 3D	79
2. PHÉP CHIẾU	81
3. PHÉP CHIẾU SONG SONG (Parallel Projections).....	83
3.1. Phép chiếu trực giao(Orthographic projection.....	83
3.2. Phép chiếu trục lượng (Axonometric)	84
3.3. Phép chiếu xiên - Oblique	87
4. PHÉP CHIẾU PHỐI CẢNH (Perspective Projection)	89
4.1. Phép chiếu phối cảnh một tâm chiếu	91
4.2. Phép chiếu phối cảnh hai tâm chiếu	91
4.3. Phép chiếu phối cảnh ba tâm chiếu	92
Tóm tắt chương:	93
Bài tập:.....	93
Bài tập trắc nghiệm:.....	94
CHƯƠNG 6: MÀU SẮC TRONG ĐỒ HOẠ	99
1. ÁNH SÁNG VÀ MÀU SẮC (light and color).....	99
1.1. Quan niệm về ánh sáng.....	99
1.2. Yếu tố vật lý	99
1.3. Cảm nhận màu sắc của con người (Physiology - Sinh lý - Human Vision)	101
1.4. Các đặc trưng cơ bản của ánh sáng.....	103
2. ÁNH SÁNG ĐƠN SẮC	104
2.1. Cường độ sáng và cách tính.....	104
2.2. Phép hiệu chỉnh gama.....	104
2.3. Xấp xỉ bán tông - halftone	105
2.4. Ma trận Dither và phép lấy xấp xỉ bán tông	107
3. CÁC HỆ MÀU TRONG MÀN HÌNH ĐỒ HOẠ.....	108
3.1. Mô hình màu RGB (Red, Green, Blue - đỏ, lục, lam).....	108
3.2. Mô hình màu CMY (Cyan, Magenta, Yellow - xanh tím, Đỏ tươi, vàng)	109
3.3. Mô hình màu YIQ.....	110
3.4. Mô hình màu HSV (Hue, Saturation, Value) - Mỹ thuật.....	110
3.5. Biểu đồ màu CIE (1931 – Commission Internationale de l'Eclairage)	112
4. CHUYỂN ĐỔI GIỮA CÁC HỆ MÀU	115
4.1. Chuyển đổi HSV - RGB	115
4.2. Chuyển đổi RGB sang XYZ.....	116
Tóm tắt:	117
Bài tập:.....	118
Bài tập trắc nghiệm:.....	118
CHƯƠNG 7: ĐƯỜNG CONG VÀ MẶT CONG TRONG 3D	120
	171

1. ĐƯỜNG CONG - CURVE	120
1.1. Điểm biểu diễn đường cong (curve represents points)	120
1.2. Đường cong đa thức bậc ba tham biến	120
1.3. Đường cong Hermite	121
1.4. Đường cong Bezier	122
1.5. Đường cong B-spline	124
2. MÔ HÌNH BỀ MẶT (Surface) VÀ CÁC PHƯƠNG PHÁP XÂY DỰNG	130
2.1. Các khái niệm cơ bản	130
2.2. Biểu diễn mảnh tứ giác	130
2.3. Mô hình hoá các mặt cong (Surface Patches)	132
2.4. Mặt từ các đường cong	136
Tóm tắt:	140
Bài tập:	141
PHỤ LỤC 1	142
1. Yêu cầu	142
2. Khởi tạo và đóng chế độ đồ hoạ	142
3. Các hàm cơ bản	143
3.1. Bảng màu của màn hình đồ hoạ	143
3.2. Nguyên sơ điểm	144
3.3. Nguyên sơ đường	144
3.4. Nguyên sơ hình chữ nhật	144
3.5. Nguyên sơ hình tròn	144
3.6. Nguyên sơ đa giác	145
3.7. Nguyên sơ văn bản	145
3.8. Cửa sổ (viewport)	146
3.9. Tạo hình ảnh chuyển động	146
PHỤ LỤC 2	165
TÀI LIỆU THAM KHẢO	169
MỤC LỤC	170

KỸ THUẬT ĐỒ HỌA

Mã số: 492KTH350

Chịu trách nhiệm bản thảo

TRUNG TÂM ĐÀO TẠO BƯU CHÍNH VIỄN THÔNG 1