

Trường Đại Học Bách Khoa TP Hồ Chí Minh
Khoa Khoa học & Kỹ thuật Máy tính



ĐỒ HỌA MÁY TÍNH

CHƯƠNG 1:

GIỚI THIỆU ĐỒ HỌA MÁY TÍNH

NỘI DUNG TRÌNH BÀY

- ❑ Giới thiệu tổng quan về đồ họa máy tính
- ❑ Đối tượng cơ bản trong đồ họa máy tính
- ❑ Thiết bị hiển thị đồ họa

cuu duong than cong. com

cuu duong than cong. com

CHƯƠNG TRÌNH

Nội dung

- Chương 1-Giới thiệu đồ họa máy tính
- Chương 2-Bước đầu tạo dựng hình ảnh
- Chương 3-Xây dựng công cụ vẽ hình ảnh
- Chương 4-Vector trong đồ họa máy tính
- Chương 5-Biến đổi hình *độ* *đường* *than* *công*. *com*
- Chương 6-Mô hình hóa đối tượng 3 D bằng lưới đa giác
- Chương 7-Phép nhìn trong không gian ba chiều
- Chương 8-Tô màu vật thể ba chiều
- Chương 9-Kỹ thuật lắp đẽo quy, ứng dụng tạo hoa văn
- Chương 10-Đồ họa raster *độ* *đường* *than* *công*. *com*

CHƯƠNG TRÌNH

□ Cách tính điểm

– Thi giữa kỳ:	20%	
– Thi cuối kỳ:	50%	
– Thực hành:		
– Bài tập :	30%	
		

Bài tập: Tuần: 2, 4, 6, 10, 12, 14, 16

Thực hành: Tuần: 1, 3, 5, 7, 11, 13, 15 (Tại C6)

TÀI LIỆU MÔN HỌC

- [1] **Francis S. Hill, Jr**, *Computer Graphics*, Macmillan Publishing Company, 1990.
- [2] **Foley, van Dam, Feiner, Hughes**, *Computer Graphics principles and practice*, Addison-Wesley Publishing Company, 1996.
- [3] **Nguyễn Hữu Lộc**, *Đồ họa máy tính và mô hình hóa hình học*, Nhà xuất bản thành phố Hồ Chí Minh, 2000.
- [4] **Hoàng Kiếm, Dương Anh Đức, Lê Đình Huy, Vũ Hải Quân**, *Cơ sở đồ họa máy tính*, Nhà xuất bản giáo dục, 2000.
- [5] **Nguyễn Quốc Cường, Hoàng Đức Hải**, *Đồ họa vi tính*, Nhà xuất bản giáo dục, 1998.
- [6] **Tống Nghĩa, Hoàng Đức Hải**, *Đồ họa máy tính trong ngôn ngữ C*, Nhà xuất bản giáo dục, 1999.
- [7] **Nguyễn Tiến, Ngô Quốc Việt**, *Giáo trình đồ họa máy tính*, Nhà xuất bản thống kê, 2001.
- [8] **Lê Tấn Hùng, Huỳnh Quyết Thắng**, *Kỹ thuật đồ họa*, Nhà xuất bản Khoa học và Kỹ thuật, 2004
- [http://www.cse.hcmut.edu.vn/~tgson/Computer Graphics](http://www.cse.hcmut.edu.vn/~tgson/Computer%20Graphics)

ĐỊNH NGHĨA ĐỒ HỌA MÁY TÍNH

- ❑ Dùng máy tính để tạo ra hình ảnh
- ❑ Đồ họa máy tính và xử lý ảnh
 - Đồ họa máy tính: tạo hình ảnh dựa trên đặc tả hoặc mô hình
 - Xử lý ảnh: nâng cao chất lượng hoặc chỉnh sửa hình ảnh.

cuu duong than cong. com

cuu duong than cong. com

ỨNG DỤNG CỦA ĐỒ HỌA MÁY TÍNH

- ❑ Ứng dụng trong giải trí, xuất bản và nghệ thuật
 - Sản xuất phim hoạt hình, tạo hiệu ứng cho phim nhựa
 - Trò chơi máy tính
 - Duyệt Web
 - Chế bản điện tử
- ❑ Xử lý ảnh
- ❑ Ứng dụng trong tự động hóa và điều khiển
- ❑ Mô phỏng
- ❑ Máy tính hỗ trợ thiết kế
 - Hỗ trợ thiết kế kiến trúc
 - Hỗ trợ thiết kế mạch điện tử
- ❑ Hình ảnh hóa số liệu khoa học

ĐỐI TƯỢNG CƠ BẢN TRONG ĐHMT

- Đường gấp khúc (polyline)
- Văn bản (text)
- Vùng tô (filled region)
- Ảnh ma trận điểm hay ảnh raster (raster image)

cuu duong than cong. com

cuu duong than cong. com

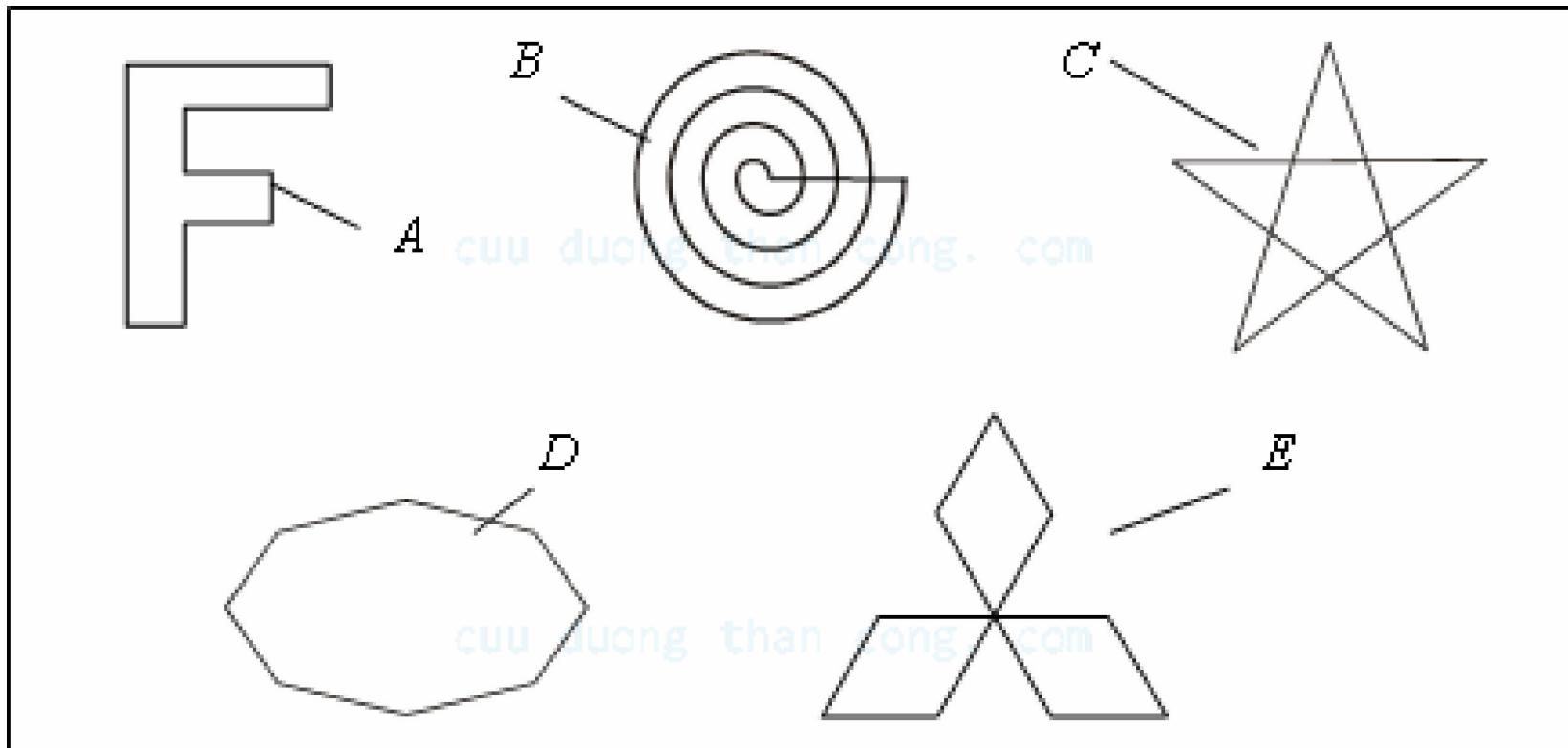
ĐƯỜNG GẤP KHÚC

- ❑ Đường gấp khúc là một tập các đoạn thẳng được nối với nhau.
- ❑ Đường gấp khúc được dùng mô phỏng đường cong.
- ❑ Các hàm liên quan
 - Vẽ điểm: **drawDot(x1, y1)**
 - Vẽ đoạn thẳng: **drawLine(x1, y1, x2, y2)**
 - Vẽ đường gấp khúc: **drawPolyline(poly)**

cuuduongthancong.com

ĐƯỜNG GẤP KHÚC

- ☐ Khi đỉnh đầu và đỉnh cuối được nối bằng một đoạn thẳng thì đường gấp khúc trở thành đa giác



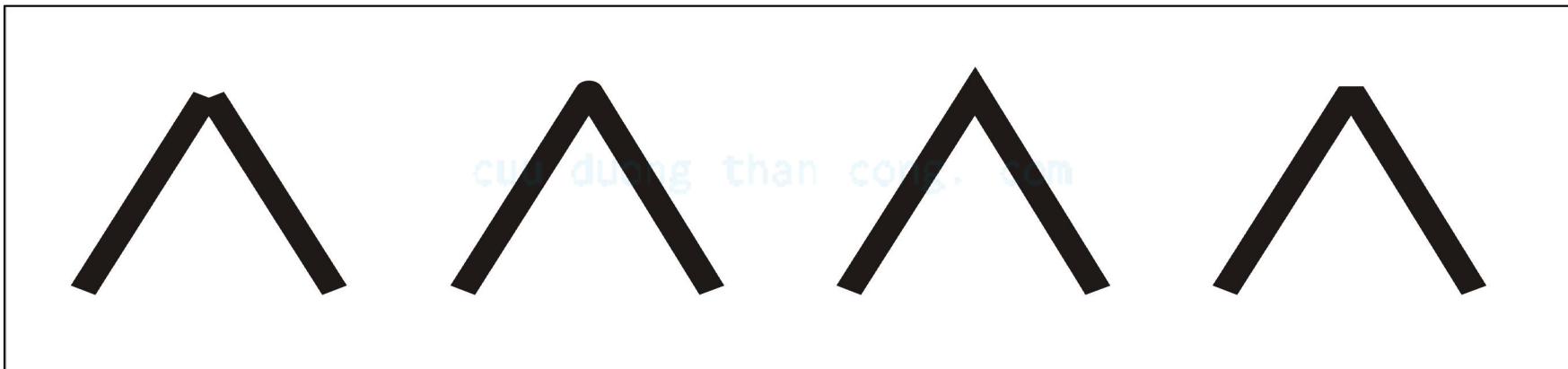
ĐƯỜNG GẤP KHÚC

❑ Thuộc tính

- Màu sắc
- Độ dày
- Kiểu đường (liền nét, đứt nét)
- Cách nối hai cạnh dày

❑ Thiết lập thuộc tính: *đường thanh công. com*

setDash (dash7) hoặc setLineThickness(thickness).



VĂN BẢN

- ❑ Một số thiết bị có hai chế độ hiển thị
 - Chế độ hiển thị văn bản
 - Chế độ hiển thị đồ họa
- ❑ Thủ tục hiển thị chuỗi
 - **drawString(x, y, string);**
- ❑ Thuộc tính
 - Font
 - Màu sắc
 - Kích thước
 - Hướng
 - Khoảng cách giữa các ký tự trong chuỗi

VĂN BẢN

Arial

Arial bold

Arial italic

Times

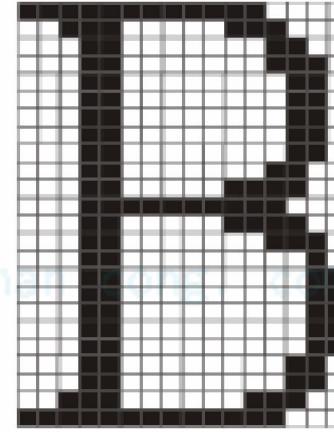
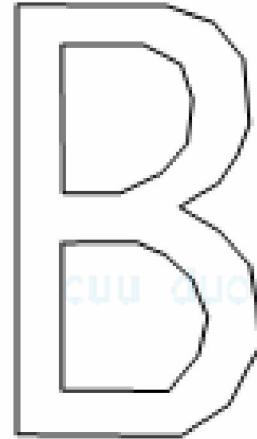
Times Bold

Times italic

Courier

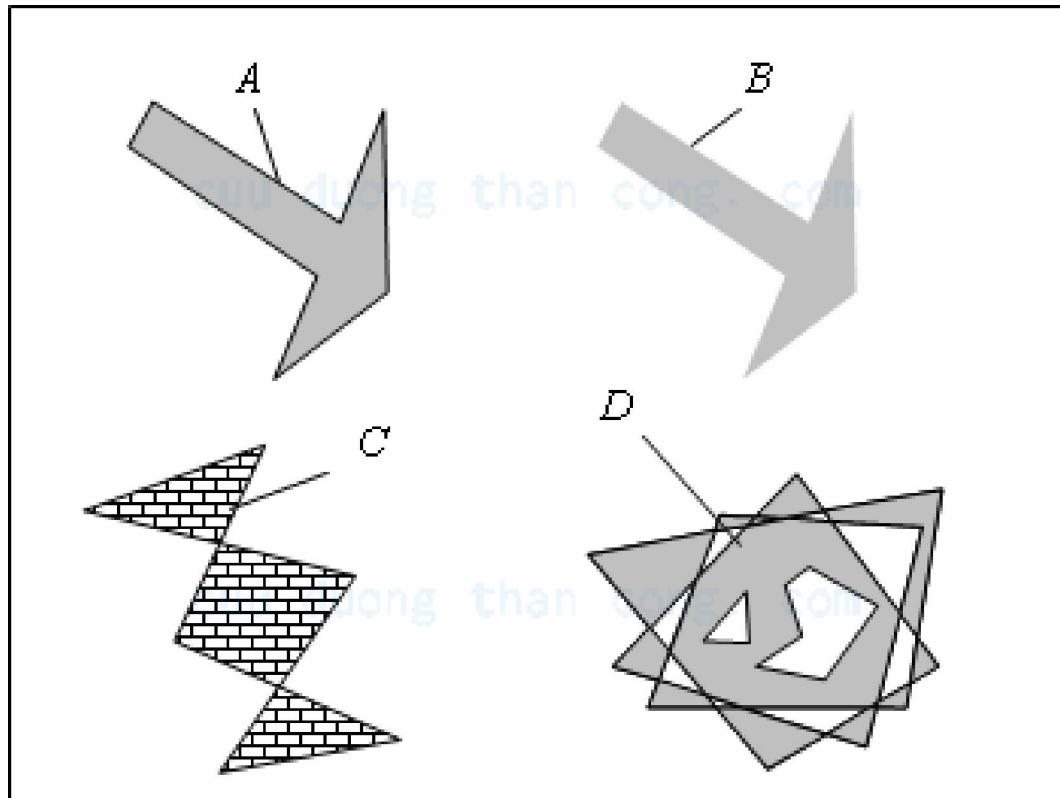
Courier bold

*cuu duong thanh **Courier** italic*



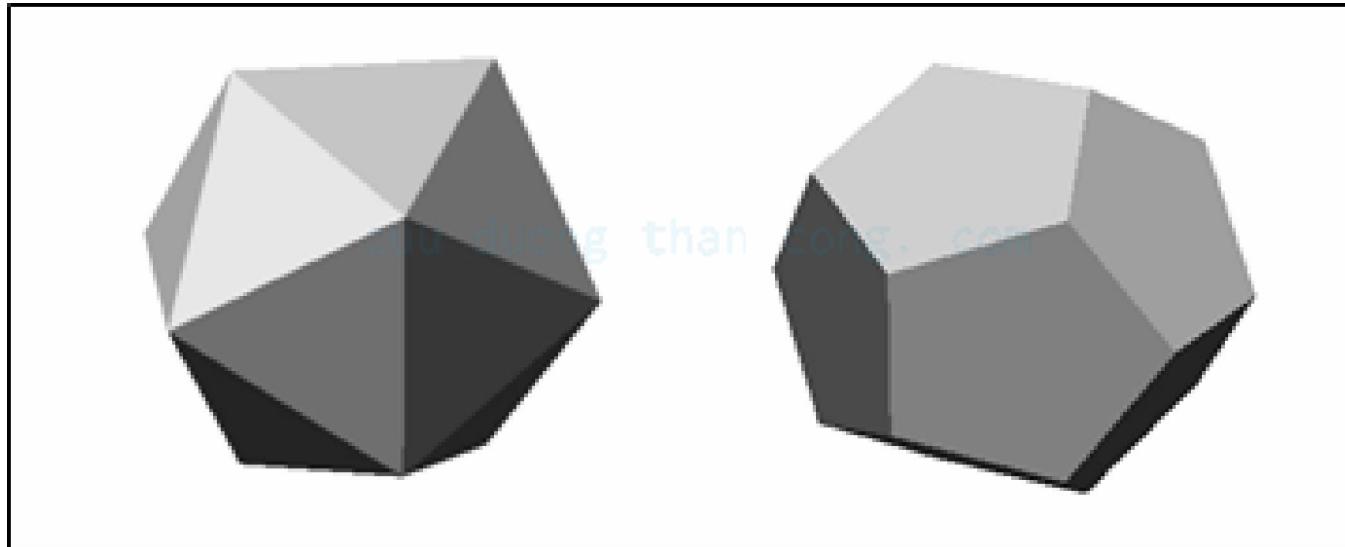
VÙNG TÔ

- ❑ Hình được tô bởi màu hoặc mẫu tô. Đường biên thường là hình đa giác.
- ❑ Thủ tục : **fillPolygon(poly, pattern);**



VÙNG TÔ

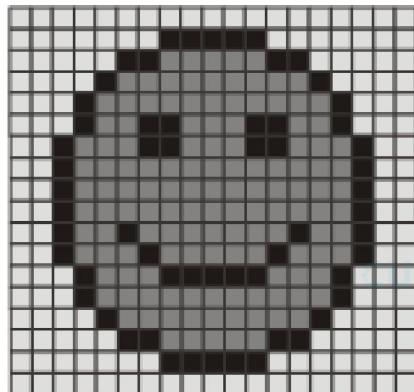
- Dùng vùng tô để mô phỏng các mặt khác nhau của vật thể tô màu



cuu duong than cong. com

ẢNH RASTER

- ❑ Được tạo bởi các pixel
- ❑ Lưu trữ dưới dạng mảng các giá trị



2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	7
2	2	2	2	2	7	7	1
2	2	2	2	7	1	1	1
2	2	2	7	1	1	1	1
2	2	2	7	1	1	7	7
2	2	2	7	1	1	7	7
2	2	2	2	2	2	2	2

- ❑ Phương pháp tạo ảnh raster
 - Thiết kế thủ công
 - Tạo bằng thuật toán
 - Quét
- ❑ Ảnh raster có thể tiến hành xử lý

THIẾT BỊ HIỂN THỊ ĐỒ HỌA

- ❑ Thiết bị đồ họa vector: tạo hình ảnh từ những đoạn thẳng
 - Ưu điểm: nhanh
 - Khuyết điểm: không tô màu cho vùng được
- ❑ Các loại thiết bị vector thường gặp
 - Máy vẽ:
 - Máy vẽ flatbed
 - Máy vẽ dạng trống
 - Màn hình vector

THIẾT BỊ HIỂN THỊ ĐỒ HỌA

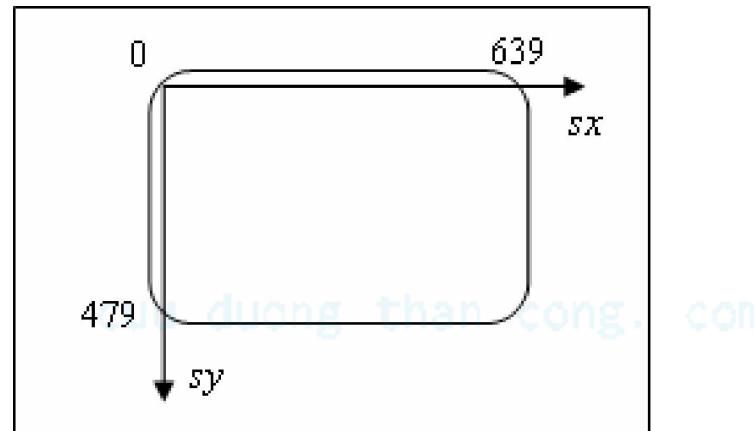
- ❑ Thiết bị raster: Tạo hình ảnh từ ma trận điểm
- ❑ Phân loại:
 - Màn hình video
 - Màn hình tấm phẳng
 - Các thiết bị sao chép cứng
 - Máy in film (film recorder)
 - Máy in laser
 - Máy in phun

THIẾT BỊ RASTER

❑ Các vấn đề cần tìm hiểu

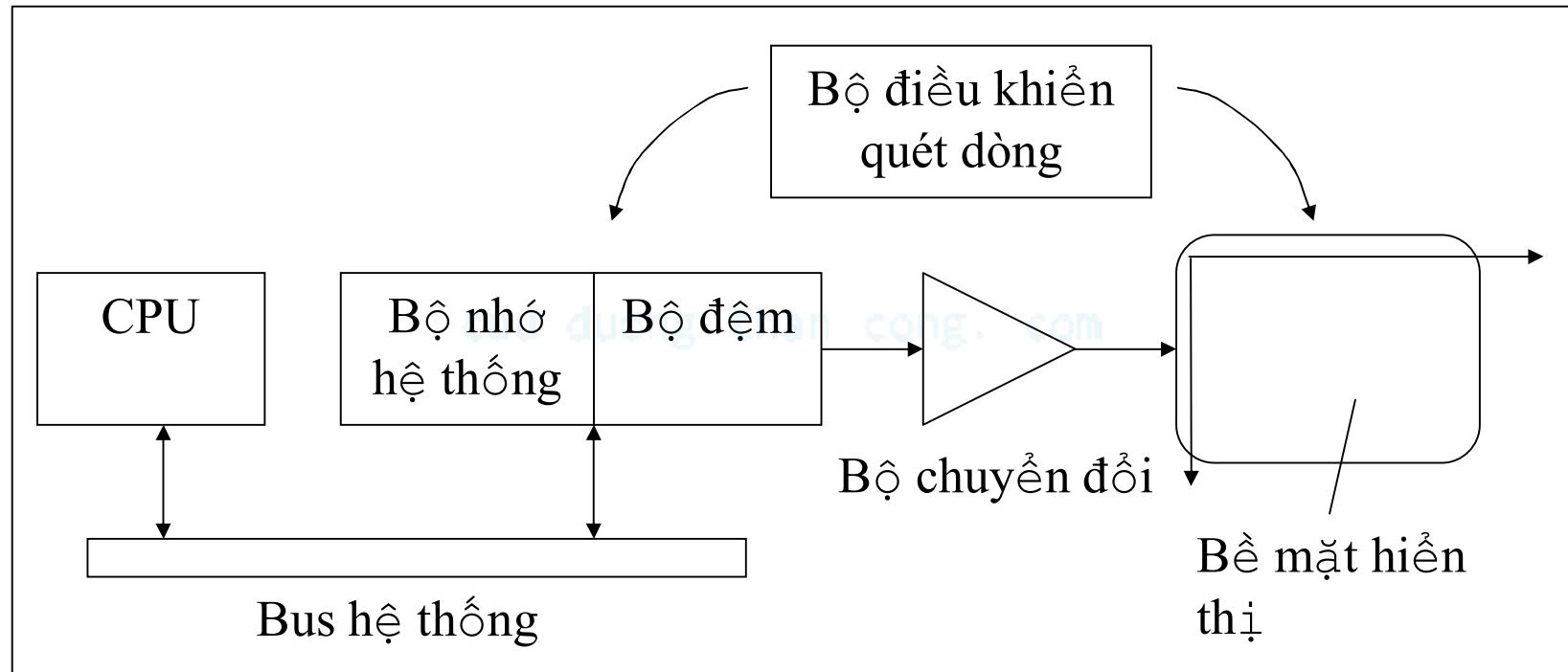
- Bề mặt hiển thị
- Bộ đệm frame
- Quá trình quét dòng

❑ Bề mặt hiển thị (display surface)



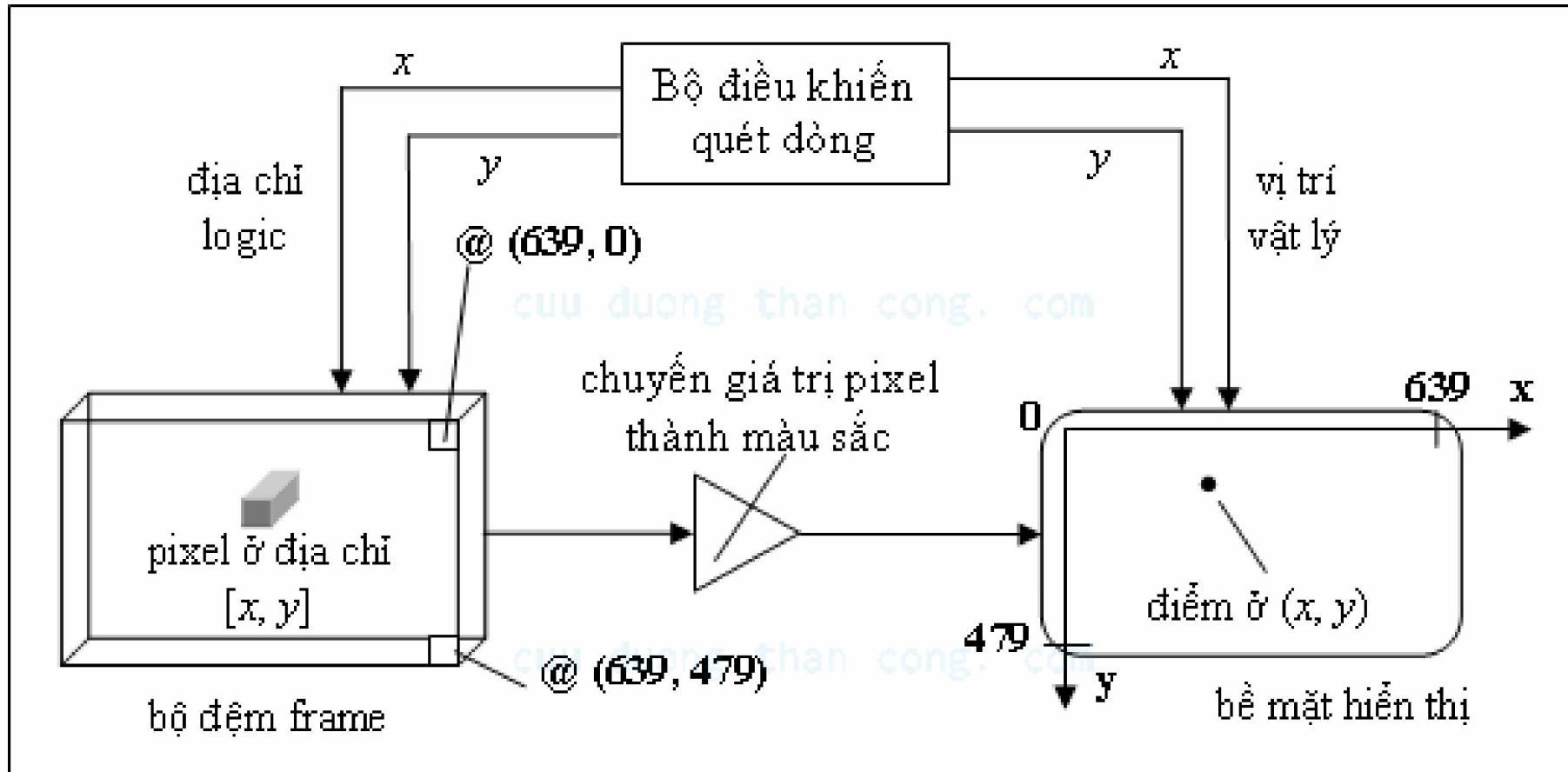
THIẾT BỊ RASTER

☐ Bộ đếm frame



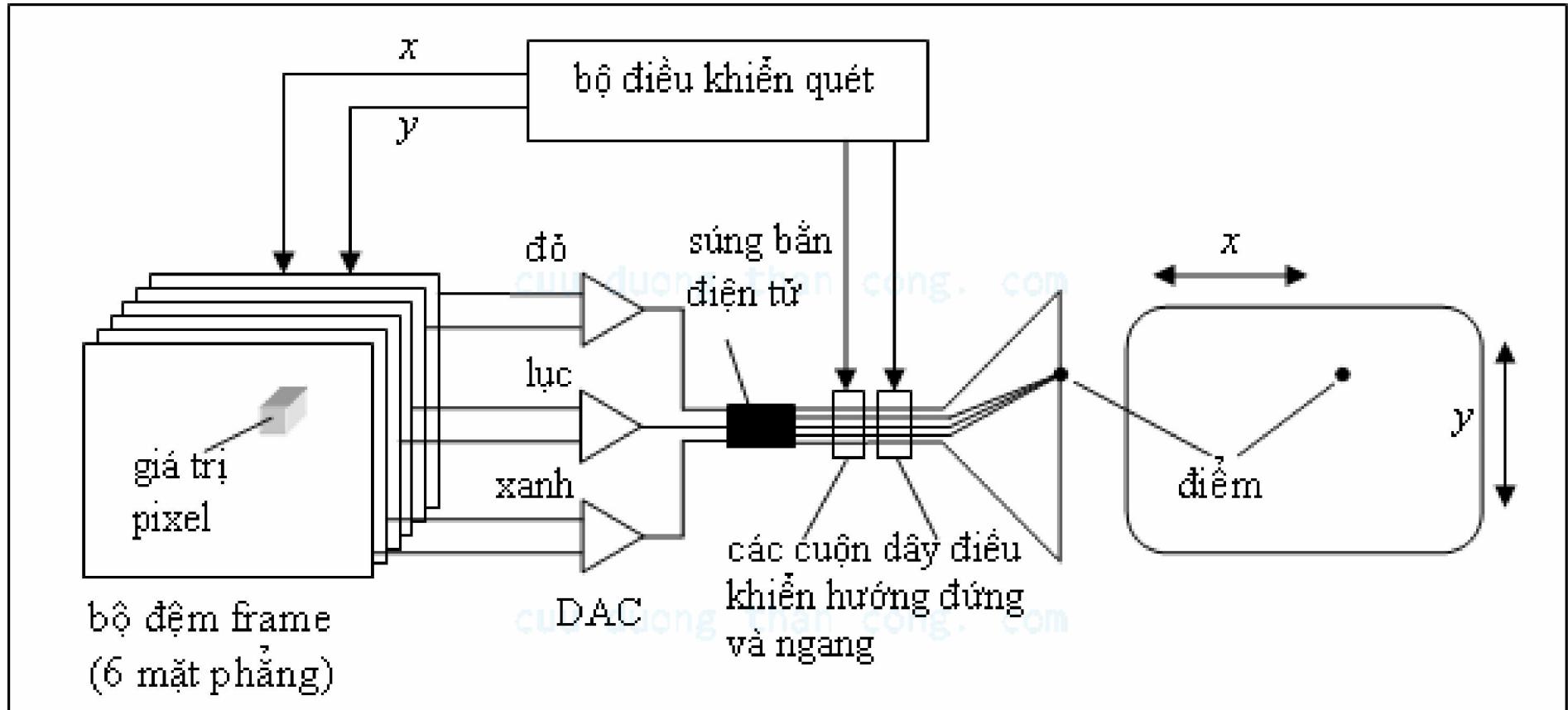
THIẾT BỊ RASTER

Quá trình quét dòng



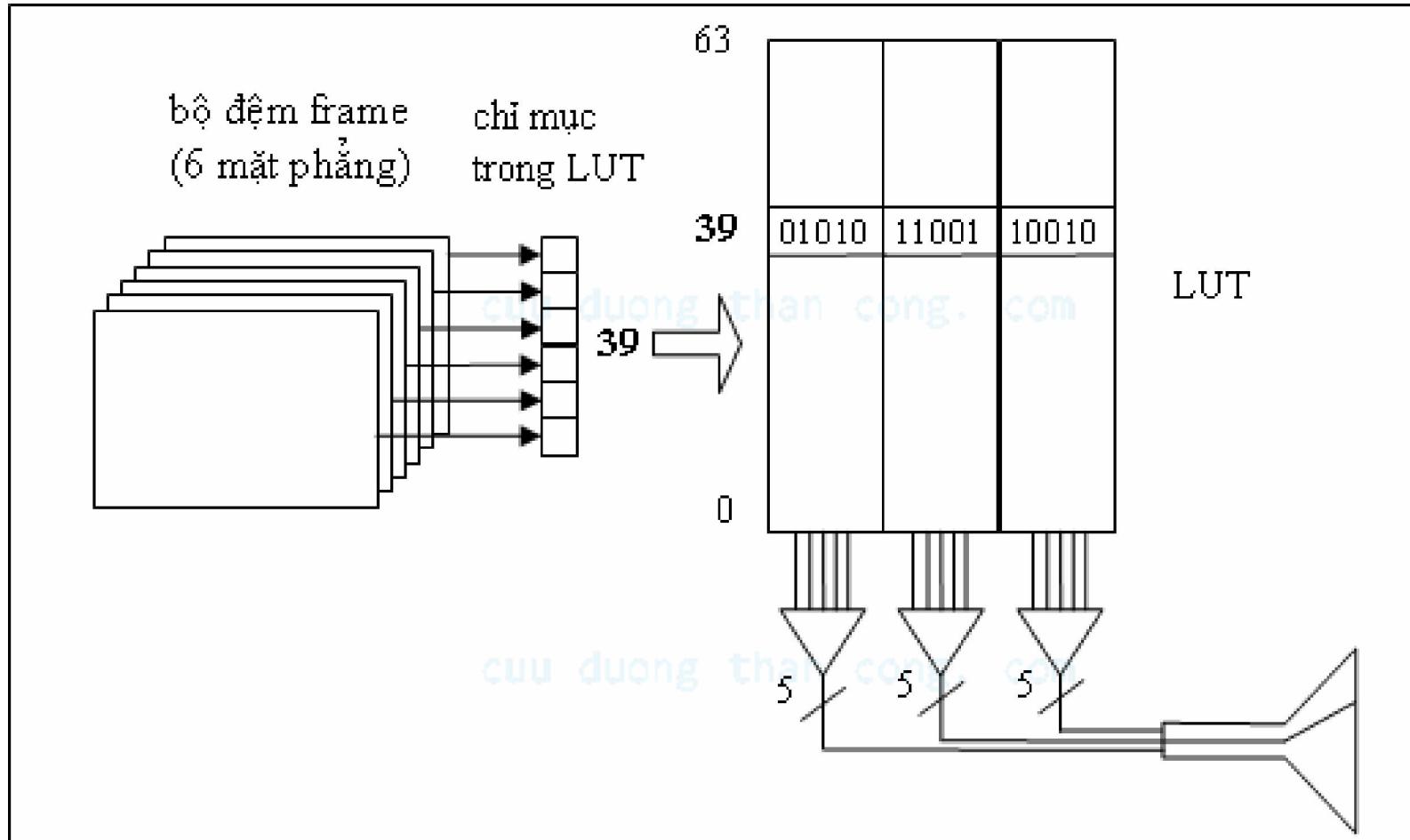
THIẾT BỊ RASTER

☐ Màn hình video màu



THIẾT BỊ RASTER

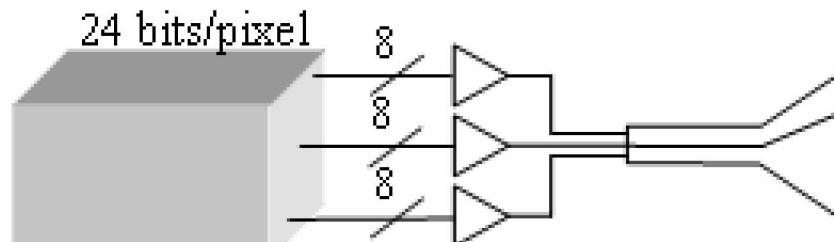
☐ Màu chỉ mục và bảng tìm kiếm



THIẾT BỊ RASTER

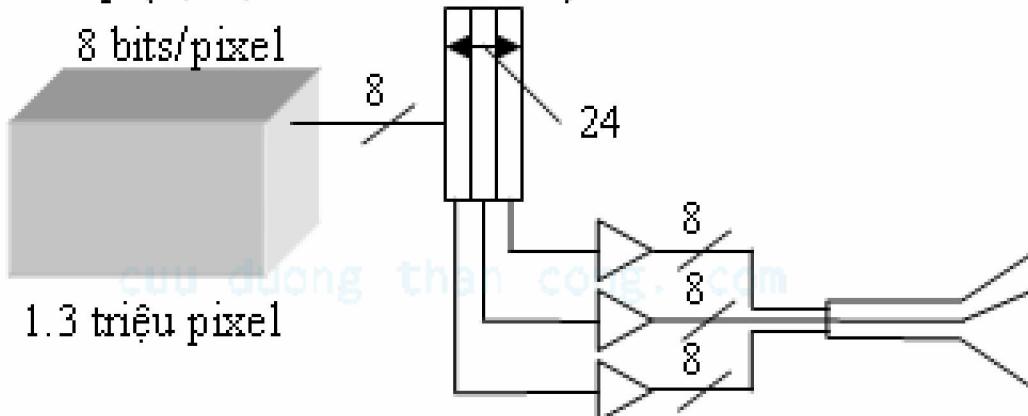
☐ Màu chỉ mục và bảng tìm kiếm

giá thành cao (bộ đệm frame ~4 MB)



1.3 triệu pixel

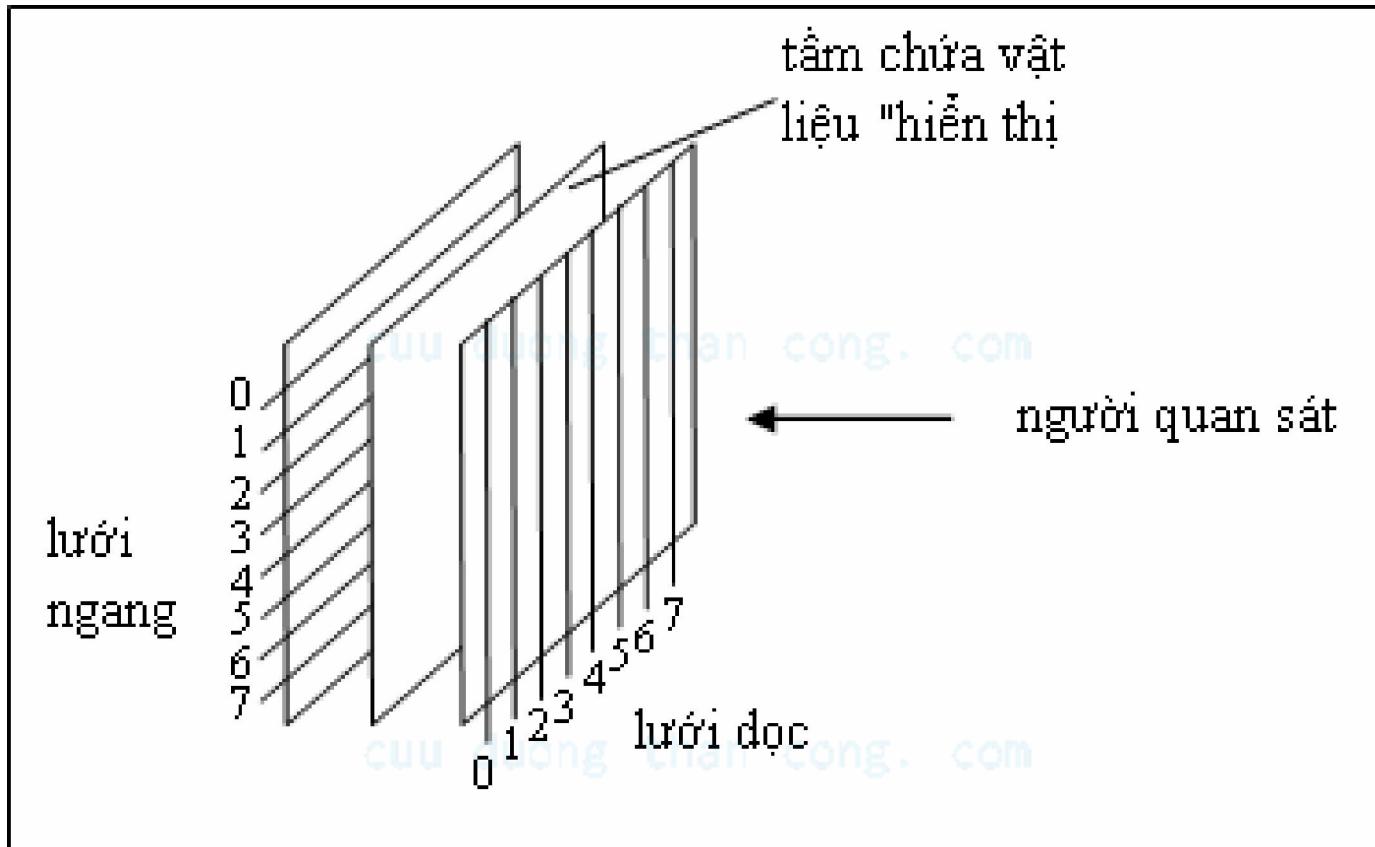
giá thành thấp (bộ đệm frame ~1 MB)



1.3 triệu pixel

THIẾT BỊ RASTER

☐ Màn hình tấm phẳng



Trường Đại Học Bách Khoa TP Hồ Chí Minh Khoa Khoa học & Kỹ thuật Máy tính

CHƯƠNG 2: BƯỚC ĐẦU TẠO HÌNH ẢNH

cuu duong thanh cong. com

cuu duong thanh cong. com

NỘI DUNG TRÌNH BÀY

- ❑ Xây dựng chương trình đồ họa.
- ❑ Thành phần cơ bản trong một chương trình sử dụng OpenGL.
- ❑ Vẽ điểm, vẽ đoạn thẳng, vẽ đường gấp khúc, vẽ hình chữ nhật.
- ❑ Giao tiếp với chuột và bàn phím.
- ❑ Một số ứng dụng.

cuu duong than cong. com

XÂY DỰNG CHƯƠNG TRÌNH ĐỒ HỌA

❑ Môi trường lập trình

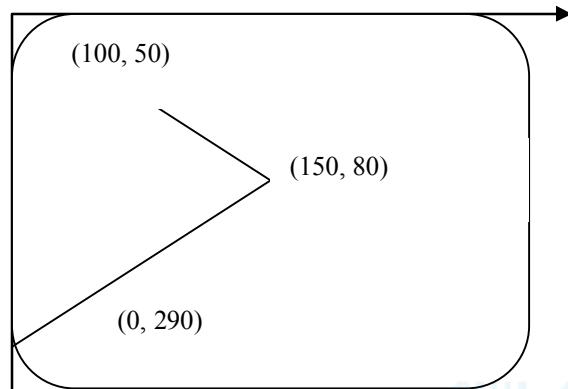
- Phần cứng: màn hình, card đồ họa.
- Phần mềm: hệ điều hành (Window), ngôn ngữ lập trình (MS Visual C++), thư viện đồ họa (OpenGL, DirectX).

❑ Trình tự xây dựng chương trình đồ họa

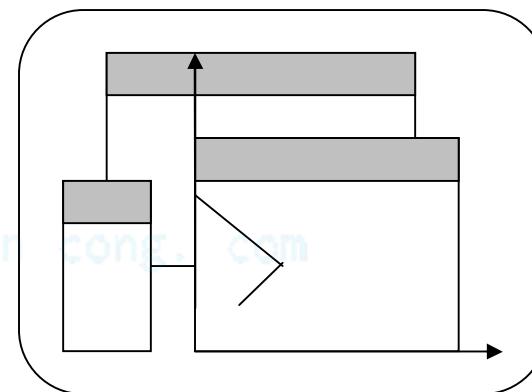
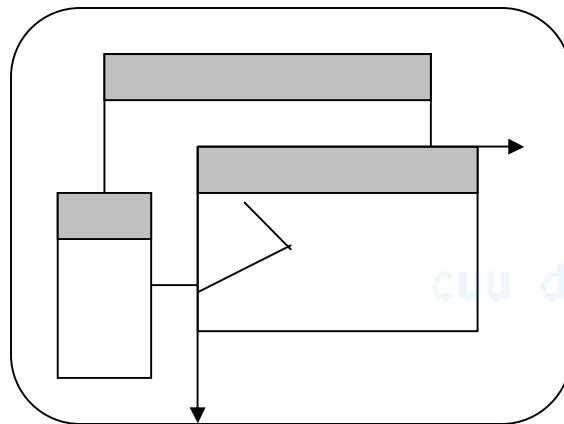
- Thiết lập chế độ hiển thị (văn bản, đồ họa)
- Thiết lập hệ trục tọa độ
- Sử dụng các hàm của môi trường lập trình để tạo dựng hình ảnh.

THIẾT LẬP TRỰC TỌA ĐỘ

❑ Môi trường lập trình DOS



❑ Môi trường lập trình Window



SỬ DỤNG CÁC HÀM ĐỂ XÂY DỰNG HÌNH ẢNH

- ❑ Hàm do hệ điều hành và ngôn ngữ lập trình cung cấp:
 - setPixel(x, y, color)
tên khác: putPixel(), SetPixel() hoặc drawPoint()
 - line(100, 50, 150, 80);
line(150, 80, 0, 290);
- ❑ Hàm do thư viện đồ họa cung cấp.
- ❑ Hàm tự xây dựng.

THÀNH PHẦN CƠ BẢN CỦA CT SỬ DỤNG OpenGL

- ❑ OpenGL là thư viện lập trình đồ họa độc lập thiết bị
 - Không phụ thuộc vào phần cứng và hệ điều hành
 - Cách làm thống nhất cho cả đồ họa 2D và 3D
- ❑ Lập trình Window
 - Lập trình sự kiện là gì?
 - Hàng đợi sự kiện (event queue).
 - Callback function.
 - Đăng ký hàm xử lý sự kiện:
 - glutDisplayFunc(myDisplay)
 - glutReshapeFunc(myReshape)
 - glutMouseFunc(myMouse)
 - glutKeyboardFunc(myKeyboard)

THÀNH PHẦN CƠ BẢN CỦA CT SỬ DỤNG OpenGL

```
// phần #include những file header cần thiết - xem phụ lục 1
void main(int argc, char** argv)
{
    glutInit(&argc, argv); //initialize the tool kit
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);//set the display mode
    glutInitWindowSize(640, 480); //set window size
    glutInitWindowPosition(100, 150); // set window position on screen
    glutCreateWindow("My first program"); // open the screen window

    // register the callback function
    glutDisplayFunc(myDisplay);

    myInit();           //additional initialization as necessary
    glutMainLoop();
}
```

THÀNH PHẦN CƠ BẢN CỦA CT SỬ DỤNG OpenGL

- ❑ Thiết lập hệ trục tọa độ

```
void myInit()
```

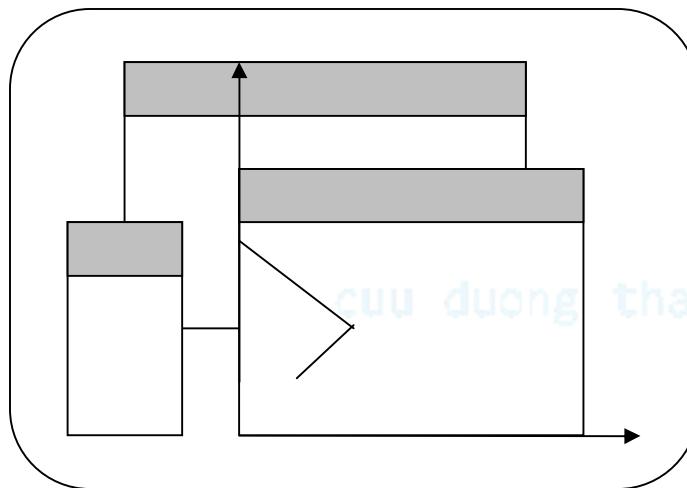
```
{
```

```
    glMatrixMode(GL_PROJECTION);
```

```
    glLoadIdentity();
```

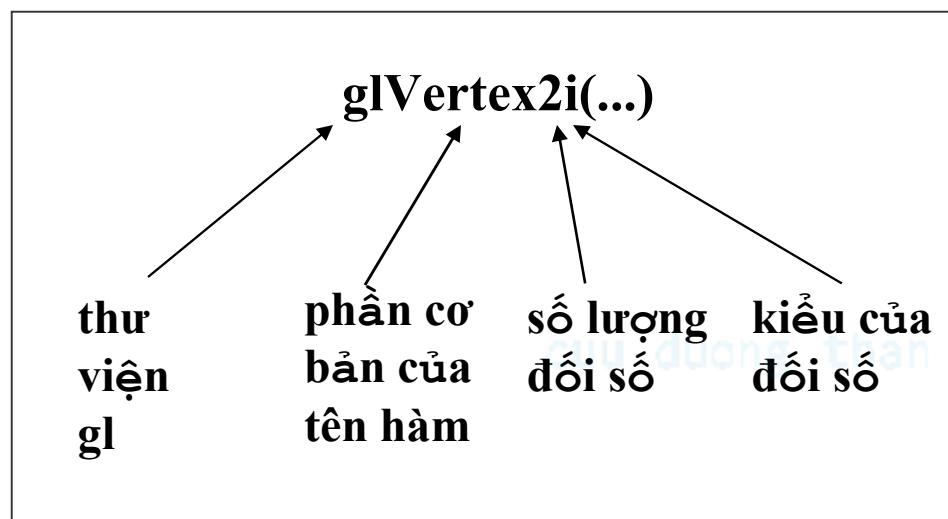
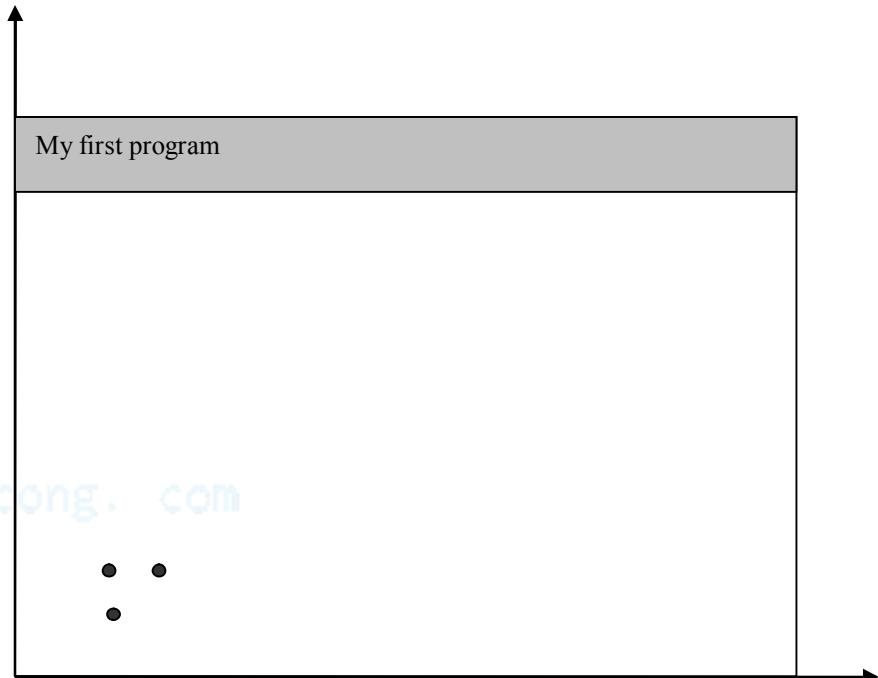
```
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);
```

```
}
```



VẼ ĐIỂM

```
glBegin(GL_POINTS);  
    glVertex2i(100, 50);  
    glVertex2i(100, 130);  
    glVertex2i(150, 130);  
glEnd();
```



VẼ ĐIỂM

☐ Trạng thái trong OpenGL

- glColor3f(1.0, 0.0, 0.0); // đổi màu vẽ thành màu đỏ
- glClearColor(1.0, 1.0, 1.0, 0.0); // set white background color
- glPointSize(4.0);
- glLineWidth(2.0);

cuu duong than cong. com

cuu duong than cong. com

VẼ ĐIỂM

- ❑ Một chương trình hoàn chỉnh

```
int main(int argc, char* argv[])
```

```
{
```

```
    glutInit(&argc, argv); //initialize the tool kit
```

```
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);//set the display mode
```

```
    glutInitWindowSize(640, 480); //set window size
```

```
    glutInitWindowPosition(100, 150); // set window position on screen
```

```
    glutCreateWindow("My first program"); // open the screen window
```

```
    glutDisplayFunc(myDisplay);// register redraw function
```

```
    myInit();
```

```
    glutMainLoop();// go into a perpetual loop
```

```
    return 0;
```

```
}
```

VẼ ĐIỂM

```
void myInit()
{
    glClearColor(1.0,1.0,1.0,0.0);// set white background color
    glColor3f(0.0f, 0.0f, 0.0f);      // set the drawing color
    glPointSize(4.0);                // a 'dot' is 4 x 4 pixels
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}
```

cuu duong than cong. com

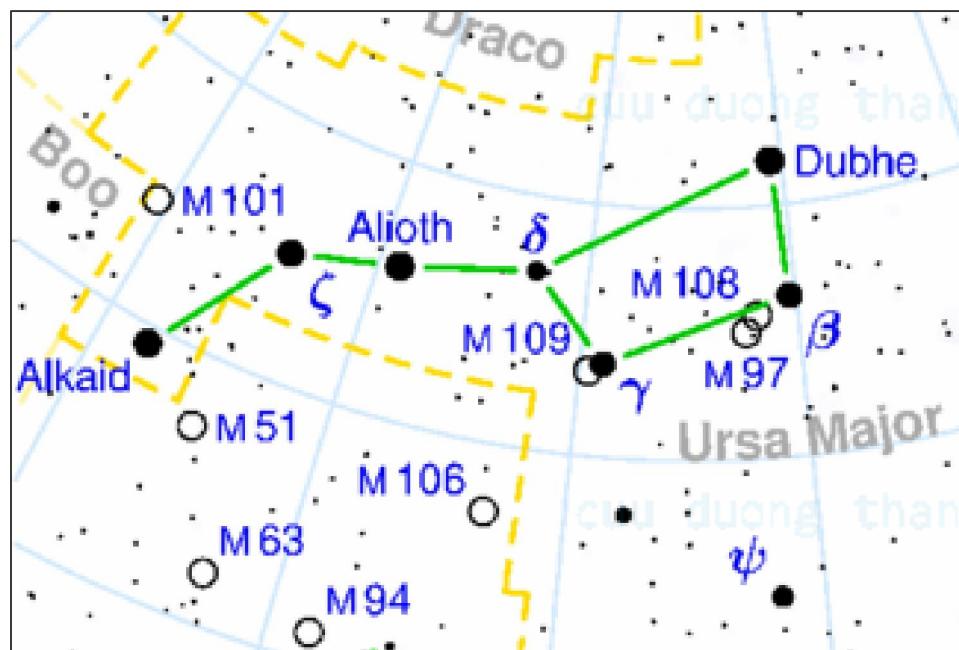
VẼ ĐIỂM

```
void myDisplay()
{
    glClear(GL_COLOR_BUFFER_BIT); // clear the screen
    glBegin(GL_POINTS);
        glVertex2i(100, 50); // draw three points
        glVertex2i(100, 130);
        glVertex2i(150, 130);
    glEnd();
    glFlush(); // send all output to display
}
```

MỘT SỐ VÍ DỤ

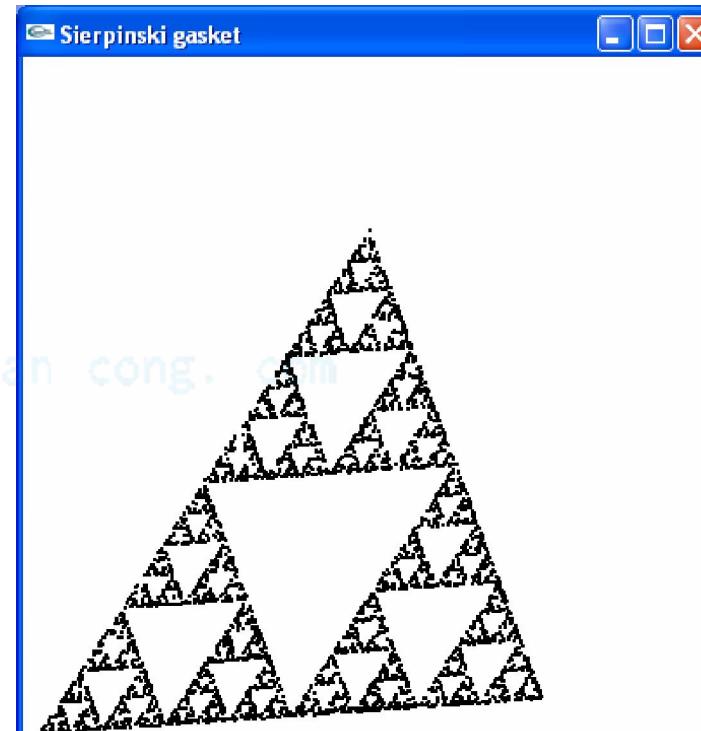
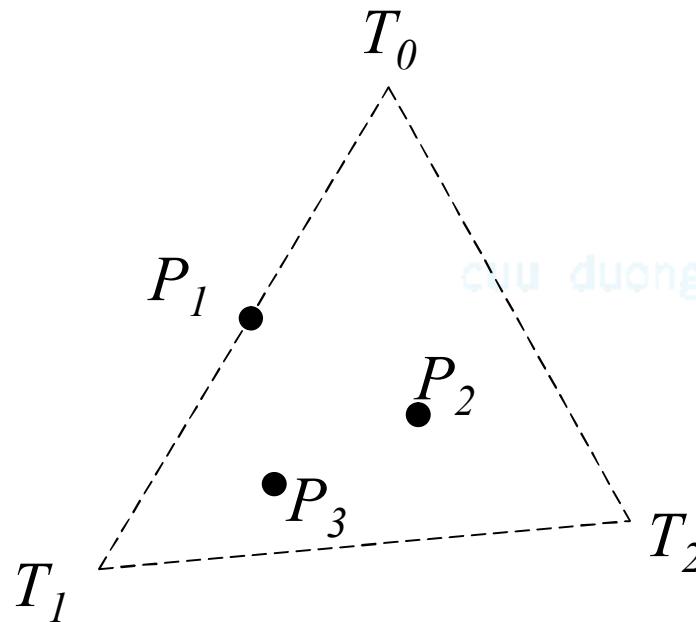
□ Vẽ chòm sao Đại Hùng:

{Dubhe, 289, 190}, {Merak, 320, 128}, {Phecda, 239, 67},
{Megrez, 194, 101}, {Alioth, 129, 83}, {Mizar, 75, 73},
{Alcor, 74, 74}, {Alkaid, 20, 10}



MỘT SỐ VÍ DỤ

□ Vẽ Sierpinski gasket



MỘT SỐ VÍ DỤ

□ Vẽ Sierpinski gasket

1. Chọn 3 điểm cố định T_0, T_1, T_2 để tạo nên một tam giác. Lưu ý rằng chúng ta không vẽ 3 điểm này lên màn hình
2. Chọn điểm khởi đầu p_0 . Điểm p_0 được chọn ngẫu nhiên trong số 3 điểm T_0, T_1, T_2 . Sau đó vẽ p_0 .

Lặp lại những bước sau cho đến khi đạt được một kết quả vừa ý

3. Chọn một điểm bất kỳ trong số 3 điểm T_0, T_1, T_2 . Gọi điểm đó là T .
4. Tạo điểm tiếp theo (p_k) bằng cách lấy điểm trung điểm của đoạn thẳng nối T và điểm trước đó (p_{k-1}). Tức là : $p_k = \text{điểm giữa của } p_{k-1} \text{ và } T$
5. Dùng hàm `drawDot()` để vẽ p_k .

cuuduongthancong.com

MỘT SỐ VÍ DỤ

❑ Vẽ Sierpinski gasket

void Sierpinski()

{

```
GLintPoint T[3]={{10, 10}, {300, 30}, {200, 300}} ;
```

```
int index = random(3) ;
```

```
GLintPoint point = T[index] ;
```

```
glClear(GL_COLOR_BUFFER_BIT);
```

```
drawDot(point.x, point.y) ;
```

```
for(int i=0; i<1000;i++)
```

{

```
    index = random(3);
```

```
    point.x = (point.x + T[index].x) / 2;
```

```
    point.y = (point.y + T[index].y) / 2;
```

```
    drawDot(point.x, point.y) ;
```

}

```
glFlush();
```

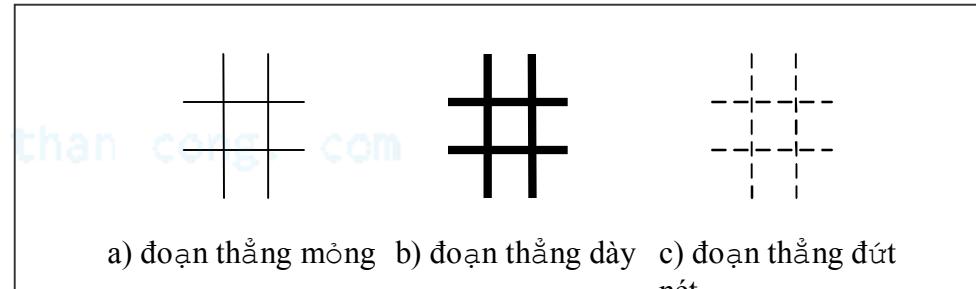
}

MỘT SỐ VÍ DỤ

```
class GLintPoint{  
    public :  
        GLint x, y ;  
};  
int random(int m)  
{  
    return rand() % m; suu duong than cong. com  
}  
void drawDot(GLint x, GLint y)  
{ //vẽ một điểm ở tọa độ (x, y)  
    glBegin(GL_POINTS);  
        glVertex2i(x, y); suu duong than cong. com  
    glEnd();  
}
```

VẼ ĐOẠN THẲNG

- ❑ `glBegin(GL_LINES);
 glVertex2i(40, 100);
 glVertex2i(202, 96);
 glEnd();`
- ❑ `void drawLineInt(GLint x1, GLint y1, GLint x2, GLint y2)
{
 glBegin(GL_LINES);
 glVertex2i(x1, y1);
 glVertex2i(x2, y2);
 glEnd();
}`
- ❑ `glBegin(GL_LINES);
 glVertex2i(10, 20); // vẽ đoạn thẳng thứ nhất
 glVertex2i(40, 20);
 glVertex2i(20, 10); // vẽ đoạn thẳng thứ hai
 glVertex2i(20, 40);
 thêm 4 lời gọi hàm glVertex2i() để vẽ hai đoạn thẳng còn lại
 glEnd();`



a) đoạn thẳng mỏng b) đoạn thẳng dày c) đoạn thẳng đứt nét

VẼ ĐƯỜNG GẤP KHÚC

- ❑ $p_0 = (x_0, y_0), p_1 = (x_1, y_1), \dots, p_n = (x_n, y_n)$
- ❑ `glBegin(GL_LINE_STRIP); //vẽ đường gấp khúc mở`

```
glVertex2i(20, 10);
glVertex2i(50, 10);
glVertex2i(20, 80);
glVertex2i(50, 80);
```

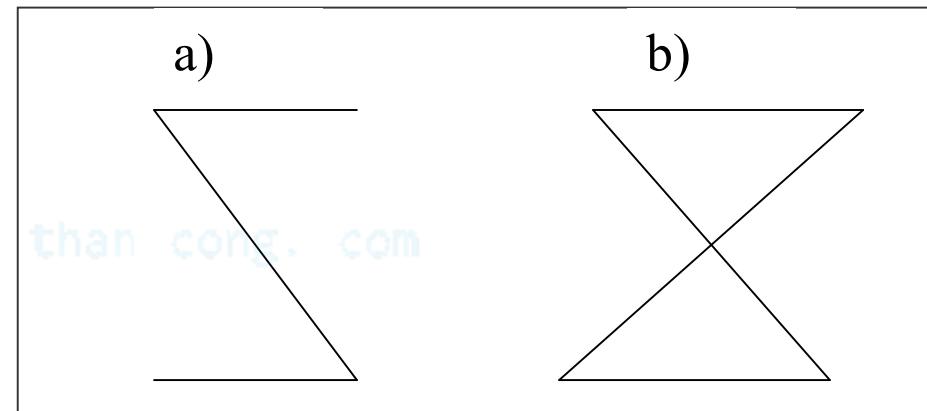
```
glEnd();
```

cuu duong than cong. com

❑ GL_LINE_LOOP

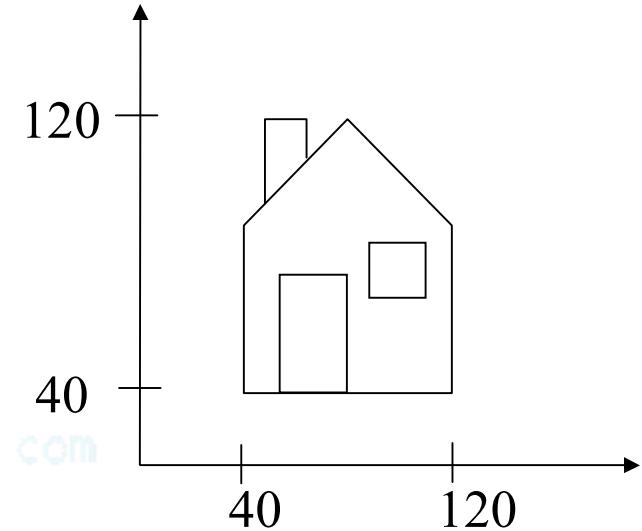
vẽ đường gấp khúc khép kín

cuu duong than cong. com



VÍ DỤ

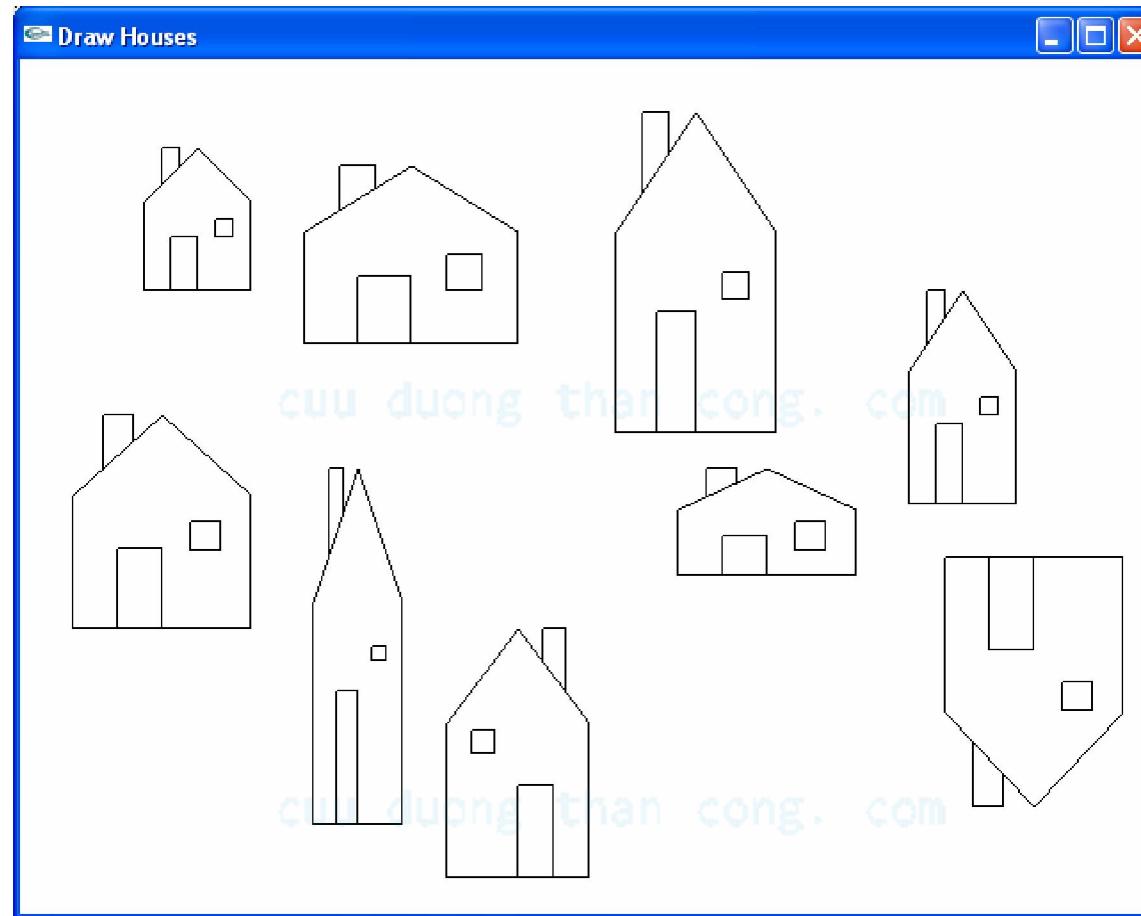
```
□ void hardwiredHouse()
{
    glBegin(GL_LINE_LOOP); //vẽ khung ngôi nhà
    glVertex2i(40, 40);
    glVertex2i(40, 90);
    glVertex2i(70, 120);
    glVertex2i(100, 90);
    glVertex2i(100, 40);
    glEnd();
    glBegin(GL_LINE_STRIP); //vẽ ống khói
    glVertex2i(50, 100);
    glVertex2i(50, 120);
    glVertex2i(60, 120);
    glVertex2i(60, 110);
    glEnd();
    . . . // vẽ cửa ra vào
    . . . // vẽ cửa sổ
}
```



VÍ DỤ

```
□ void parameterizedHouse(GLintPoint peak,GLint width,GLint height)
  // tọa độ của nóc nhà là peak,
  // chiều cao, chiều rộng của ngôi nhà là height và width
{
    glBegin(GL_LINE_LOOP);
    glVertex2i(peak.x,          peak.y);
    glVertex2i(peak.x + width/2,peak.y - 3*height/8);
    glVertex2i(peak.x + width/2,peak.y -  height);
    glVertex2i(peak.x - width/2,peak.y -  height);
    glVertex2i(peak.x - width/2,peak.y - 3*height/8);
    glEnd();
    vẽ ống khói
    vẽ cửa ra vào
    vẽ cửa sổ
}
```

VÍ DỤ



VẼ ĐƯỜNG GẤP KHÚC

```
□ class GLintPointArray
{
    const int MAX_NUM = 100;
    public:
        int num ;
        GLintPoint pt[MAX_NUM] ;
};

□ void drawPolyLine(GLintPointArray poly,int closed)
{
    glBegin(closed ? GL_LINE_LOOP : GL_LINE_STRIP);
    for(int i=0;i<poly.num;i++)
        glVertex2i(poly.pt[i].x, poly.pt[i].y);
    glEnd();
    glFlush();
}
```

VẼ ĐOẠN THẲNG DÙNG moveto(), lineto()

□ GLintPoint CP; //global current position

```
void moveto(GLint x, GLint y)
```

```
{
```

```
    CP.x = x; CP.y = y; //update CP
```

```
}
```

```
void lineto(GLint x, GLint y)
```

```
{
```

```
    glBegin(GL_LINES); //draw the line
```

```
        glVertex2i(CP.x, CP.y);
```

```
        glVertex2i(x, y);
```

```
    glEnd();
```

```
    glFlush();
```

```
    CP.x = x; CP.y = y; //update CP
```

```
}
```

VẼ HÌNH CHỮ NHẬT

❑ Cú pháp

```
glRecti(GLint x1, GLint y1, GLint x2, GLint y2);
```

// vẽ một hình chữ nhật mà hai góc đối diện có tọa độ là (x1, y1) và (x2, y2)

// hình chữ nhật sẽ được tô bằng màu vẽ hiện hành (current color)

❑ Ví dụ

```
glClearColor(1.0, 1.0, 1.0, 0.0); // nền màu trắng
```

```
glClear(GL_COLOR_BUFFER_BIT); // xóa cửa sổ
```

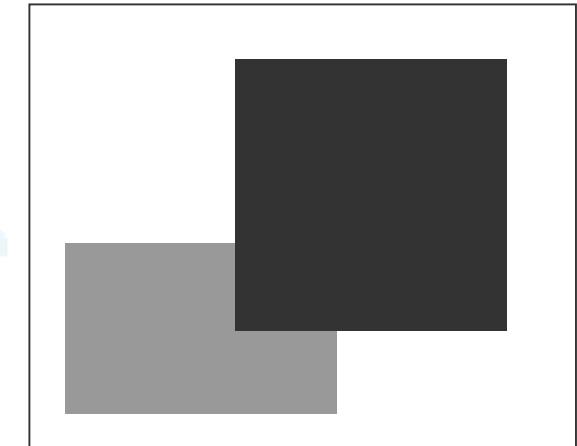
```
glColor3f(0.6, 0.6, 0.6); // bright gray
```

```
glRecti(20, 20, 100, 70);
```

```
glColor3f(0.2, 0.2, 0.2); // dark gray
```

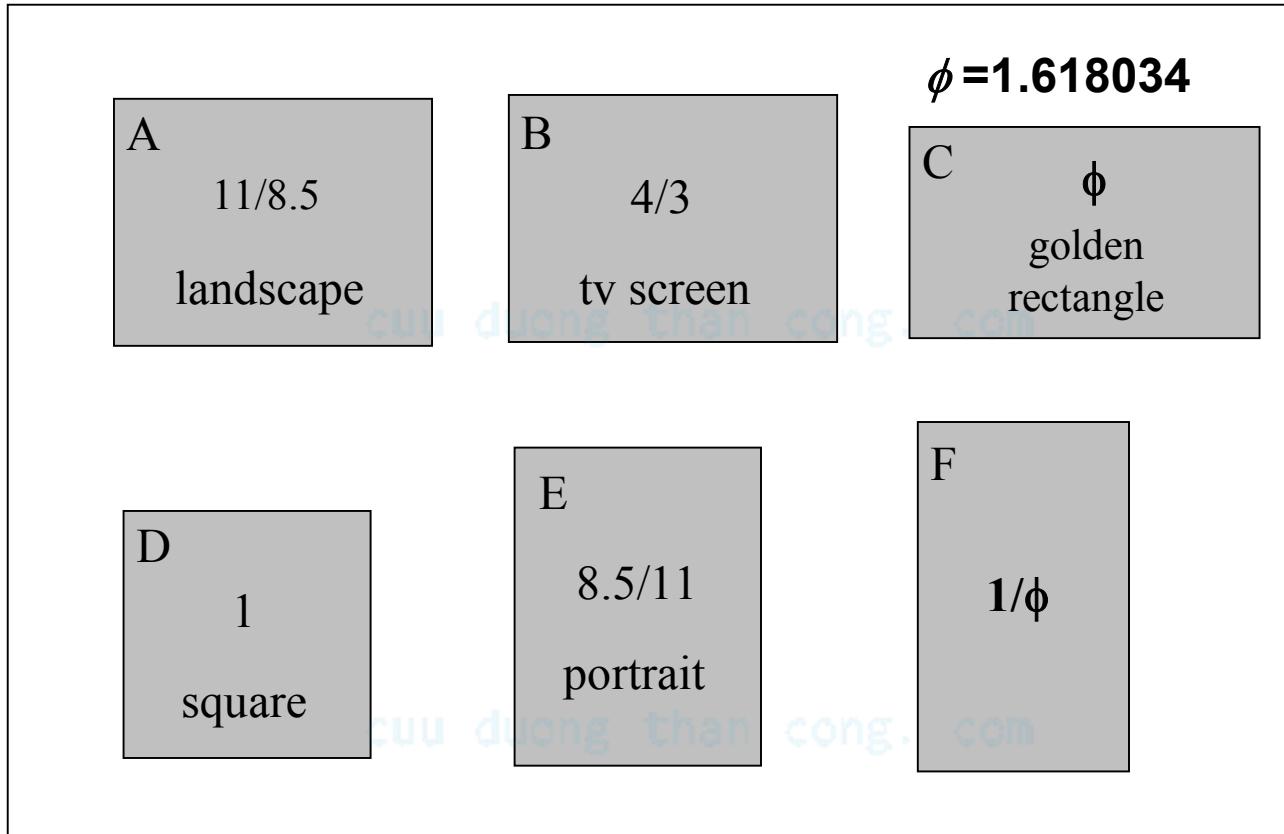
```
glRecti(20, 20, 100, 70);
```

```
glFlush();
```



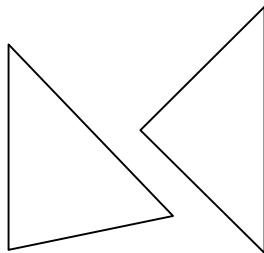
HỆ SỐ TỶ LỆ CỦA HÌNH CHỮ NHẬT

$$\text{aspect ratio} = \frac{\text{width}}{\text{height}}$$

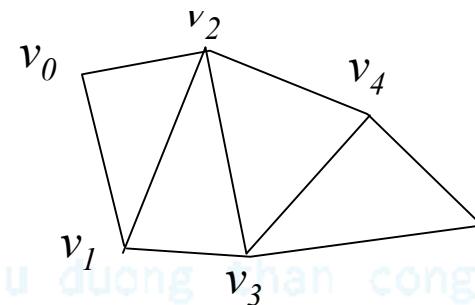


NHỮNG ĐỐI TƯỢNG ĐỒ HỌA CƠ BẢN KHÁC TRONG OpenGL

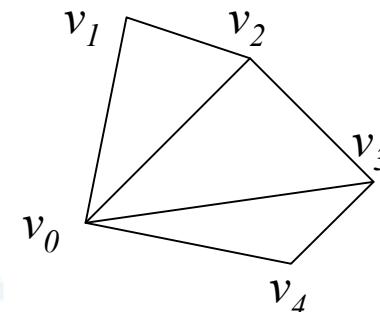
GL_TRIANGLES



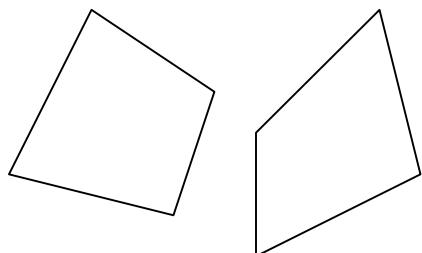
GL_TRIANGLE_STRIP



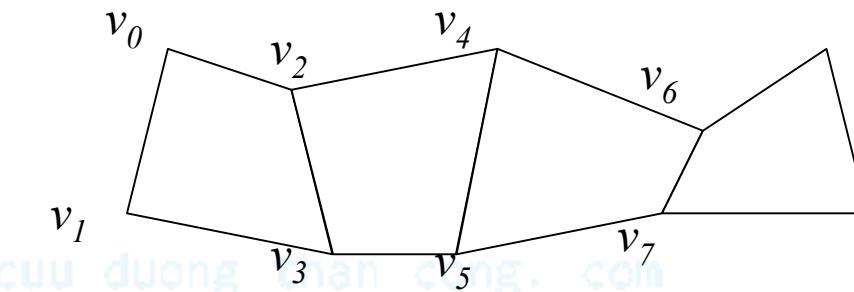
GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP



GIAO TIẾP VỚI CHUỘT

Thao tác bấm chuột

```
 glutMouseFunc(myMouse)
 void myMouse(int button, int state, int x, int y);
    button: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON và
              GLUT_RIGHT_BUTTON
    state: GLUT_UP và GLUT_DOWN
    x, y: tọa độ màn hình, trục x chạy từ trái sang phải, trục y chạy từ
          trên xuống dưới
 void myMouse(int button, int state, int x, int y)
{
    if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
        drawDot(x, screenHeight - y);
    else if(button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
        exit(-1);
}
```

GIAO TIẾP VỚI CHUỘT

Thao tác di chuyển chuột

- glutMotionFunc(myMovedMouse)
- void myMovedMouse(int x, int y)
- void myMovedMouse(int mouseX, int mouseY)

```
{  
    GLint      x = mouseX;  
    GLint      y = mouseY;  
    GLint      brushSize = 20;  
    glRecti(x, y, x + brushSize, y + brushSize);  
    glFlush();  
}
```

GIAO TIẾP VỚI BÀN PHÍM

```
□ glutKeyboardFunc (myKeyboard)
□ void myKeyboard(unsigned int key, int x, int y);
    - key: mã ASCII của phím bị nhấn
    - x, y: vị trí chuột
□ void myKeyboard(unsigned char theKey, int mouseX, int mouseY) {
    GLint      x = mouseX;
    GLint      y = screenHeight - mouseY;
    switch(theKey)
    {
        case 'p': drawDot(x, y); // vẽ một điểm ở vị trí con chuột
                    break;
        case GLUT_KEY_LEFT: List[++last].x = x ;
                            List[last].y = y; // thêm điểm
                            break;
        case 'E': exit(-1);
        default: break;
    }
}
```

Trường Đại Học Bách Khoa TP Hồ Chí Minh
Khoa Khoa học & Kỹ thuật Máy tính

CHƯƠNG 3:
XÂY DỰNG CÔNG CỤ
VẼ HÌNH ẢNH

cuuduongthancong.com

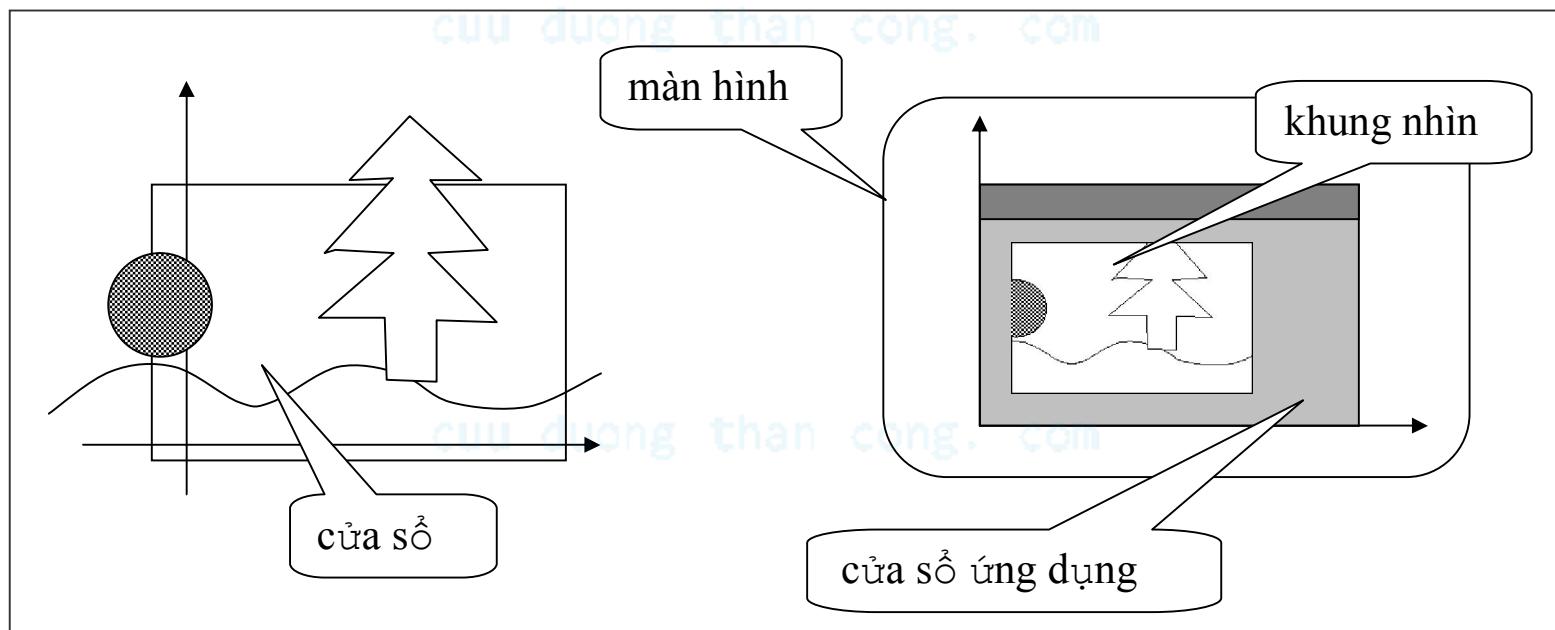
NỘI DUNG TRÌNH BÀY

- ❑ Cửa sổ và khung nhìn
- ❑ Phép biến đổi từ cửa sổ sang khung nhìn
- ❑ Giải thuật cắt xén
- ❑ Xây dựng lớp **Canvas** phục vụ cho việc vẽ hình ảnh
- ❑ Vẽ tương đối và đồ họa con rùa
- ❑ Tạo hình ảnh từ đa giác đều
- ❑ Vẽ đường tròn và cung tròn
- ❑ Biểu diễn và vẽ đường cong theo dạng tham số

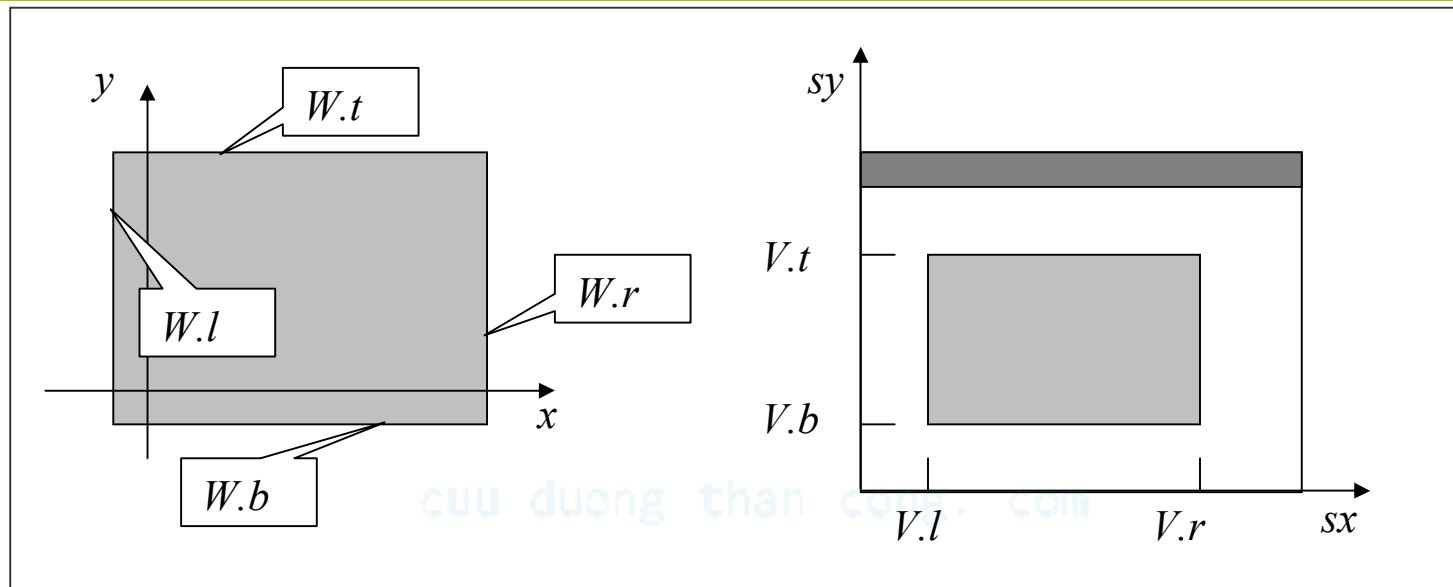
cuu duong than cong. com

CỬA SỔ VÀ KHUNG NHÌN

- ❑ **Hệ trục tọa độ thế giới:** hệ trục miêu tả đối tượng, không quan tâm đến đơn vị đo.
- ❑ **Cửa sổ:** hình chữ nhật trong hệ trục tọa độ thế giới. Phần nằm trong cửa sổ sẽ được vẽ, phần nằm ngoài bị loại bỏ.
- ❑ **Khung nhìn:** hình chữ nhật trong **cửa sổ màn hình**, cho phép hiển thị hình ảnh ở đâu trên màn hình.

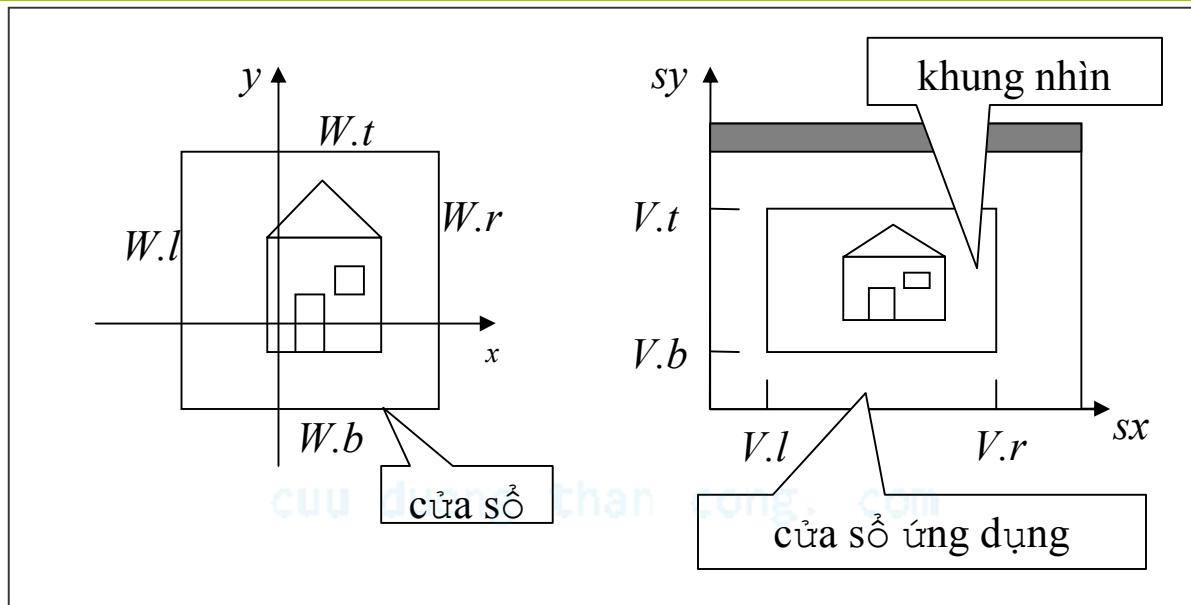


PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN



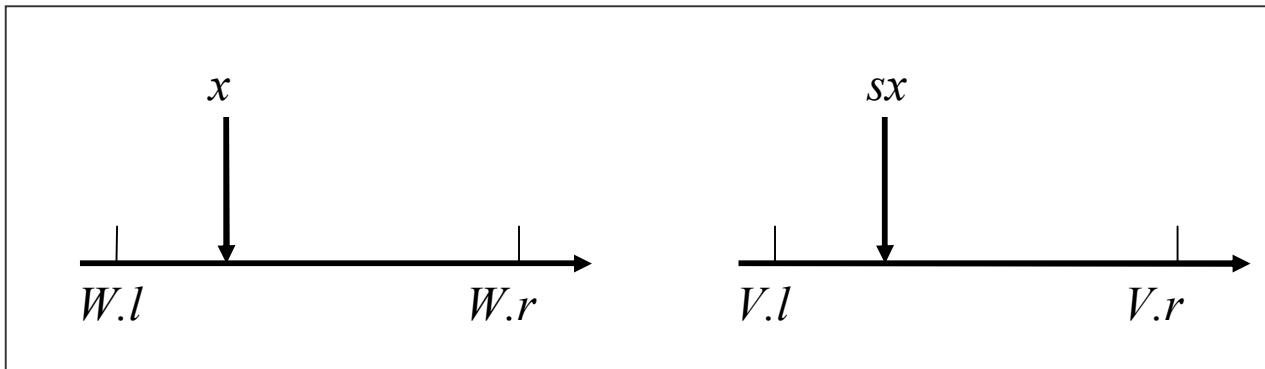
- ❑ Cửa sổ là hình chữ nhật có vị trí và kích thước bất kỳ
- ❑ Khung nhìn cũng là hình chữ nhật có vị trí và kích thước bất kỳ, nhưng phải nằm trong cửa sổ ứng dụng
- ❑ Hệ số tỷ lệ của cửa sổ và khung nhìn không nhất thiết bằng nhau. Khi hai giá trị này khác nhau, hình ảnh sẽ bị biến dạng

PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN



- (x, y) nằm trong cửa sổ \rightarrow tìm (sx, sy) thuộc khung nhìn
- Phép biến đổi phải bảo toàn tỷ lệ khoảng cách
- sx phụ thuộc tuyến tính vào x , sy phụ thuộc tuyến tính vào y :
$$sx = Ax + C$$
$$sy = By + D$$

PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN



$$\frac{sx - V.l}{V.r - V.l} = \frac{x - W.l}{W.r - W.l}$$
 cuu duong than cong
$$sx = \frac{V.r - V.l}{W.r - W.l}x + \left(V.l - \frac{V.r - V.l}{W.r - W.l}W.l \right)$$

$$A = \frac{V.r - V.l}{W.r - W.l}$$

$$C = V.l - \frac{V.r - V.l}{W.r - W.l}W.l = V.l - AW.l$$

$$B = \frac{V.t - V.b}{W.t - W.b}$$

$$D = V.b - \frac{V.t - V.b}{W.t - W.b}W.b = V.b - BW.b$$

PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN

Hiện thực trong OpenGL

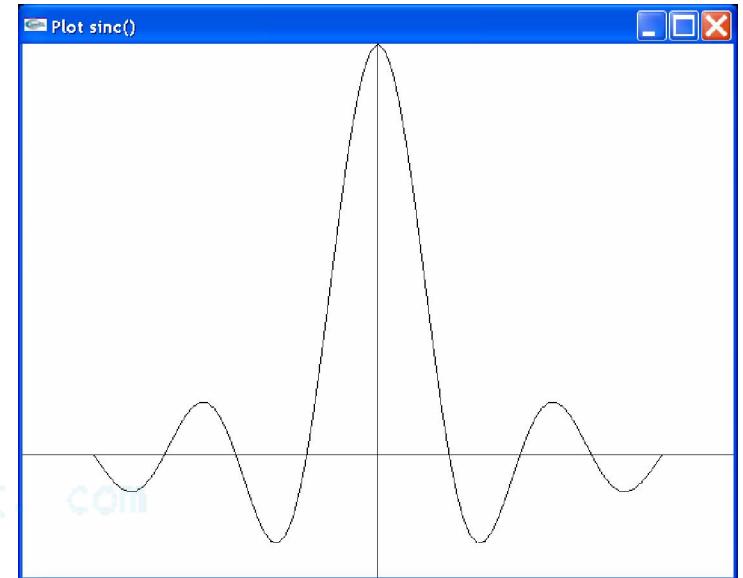
```
void setWindow(float left, float right, float  
bottom, float top)  
{  
    glMatrixMode(GL_PROJECTION) ;  
    glLoadIdentity() ;  
    gluOrtho2D(left, right, bottom, top) ;  
}  
void setViewport(int left, int right, int bottom,  
int top)  
{  
    glViewport(left, bottom, right - left, top -  
bottom) ;  
}
```

PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN

Ví dụ

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

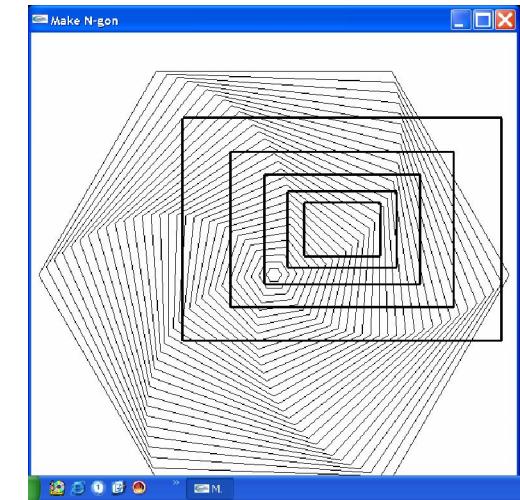
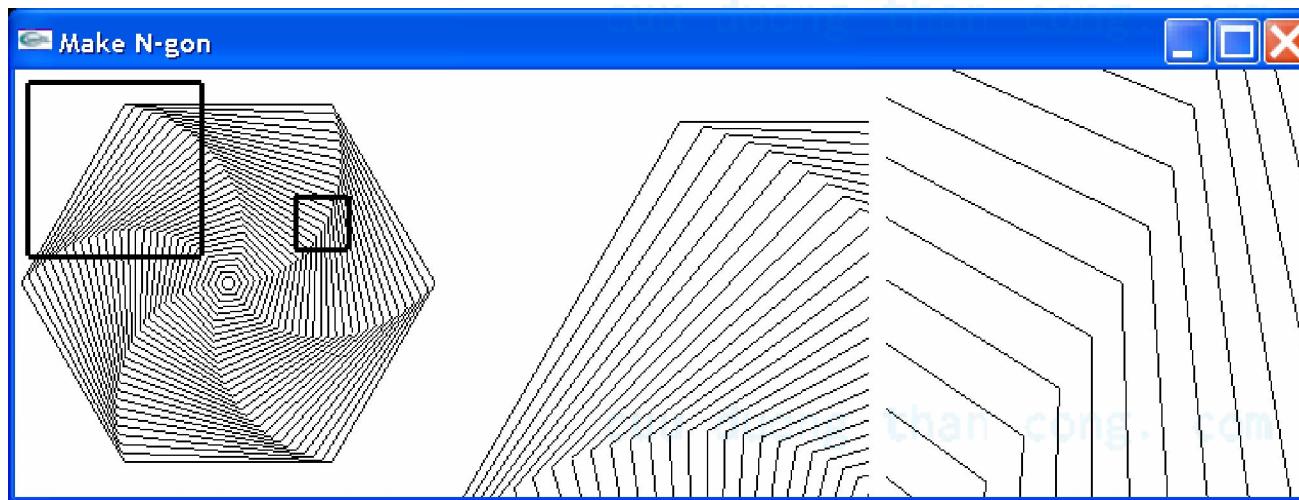
```
void myDisplay()
{
    setWindow(-5.0, 5.0, -0.3, 1.0);
    setViewport(0, 640, 0, 480);
    glBegin(GL_LINE_STRIP);
    for(GLfloat x = -4.0; x < 4.0; x+=0.1)
    {
        GLfloat y = sin(3.14159 * x) / (3.14159 * x);
        glVertex2f(x, y);
    }
    glEnd();
    glFlush();
}
```



PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN

Ứng dụng

- Cắt xén một phần của hình ảnh
- Phóng to, thu nhỏ và dạo trong khung cảnh

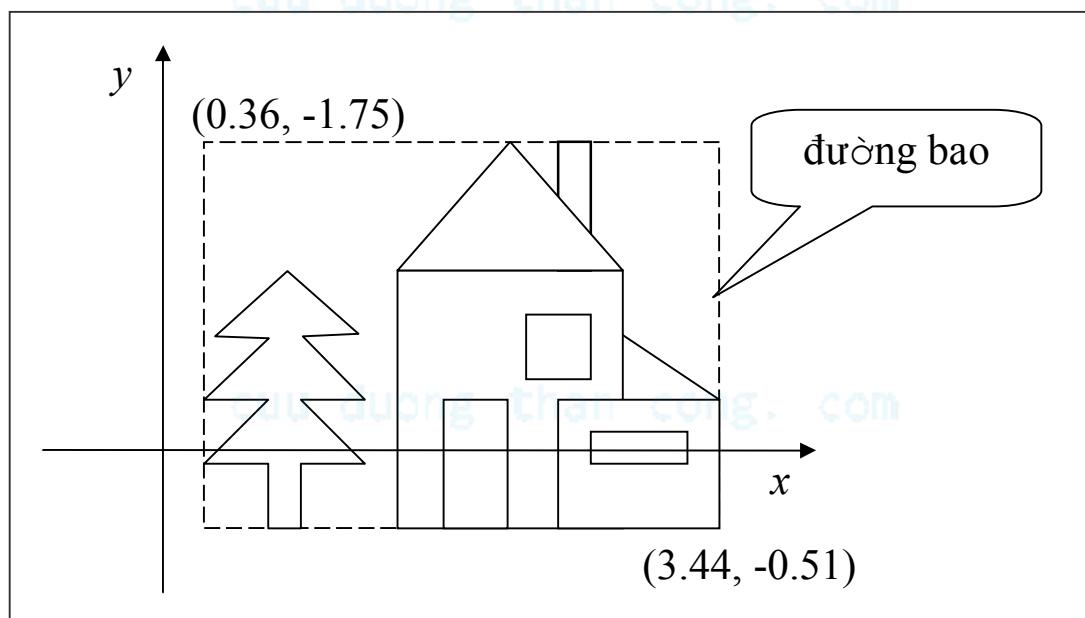


PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN

Thiết lập cửa sổ và khung nhìn tự động

Thiết lập cửa sổ

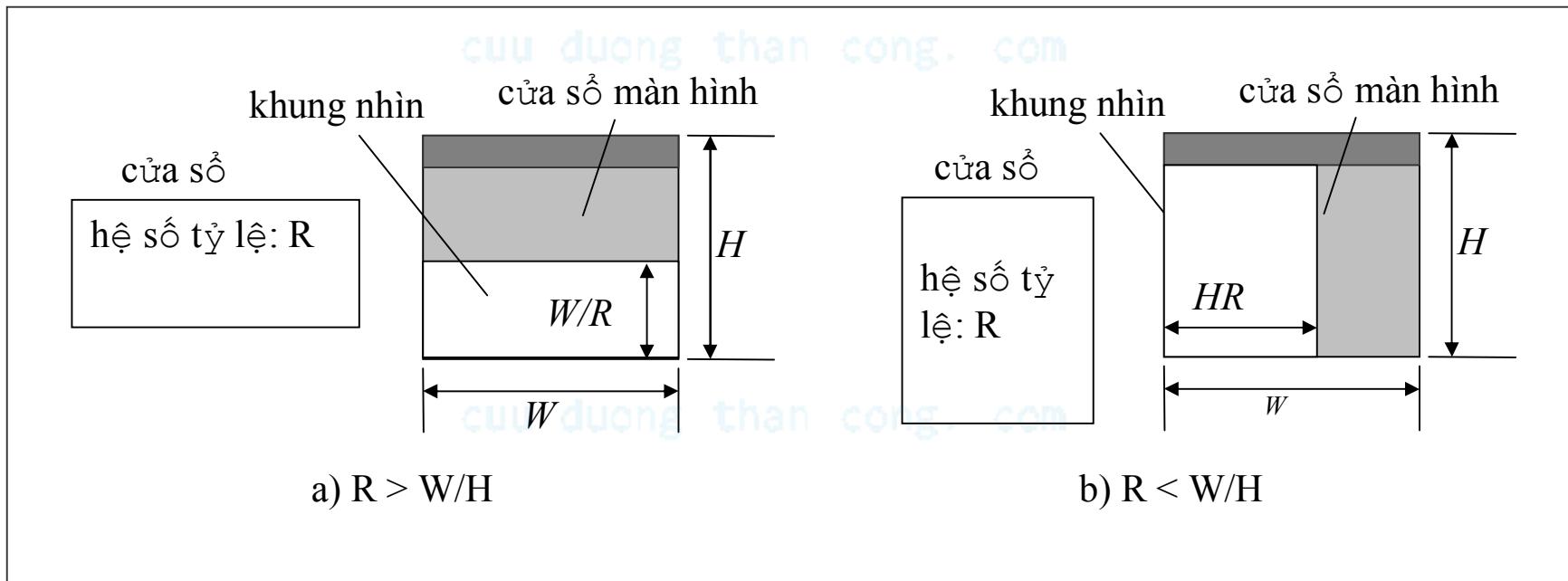
- Thực hiện thủ tục vẽ hình nhưng không thực hiện thao tác vẽ mà chỉ để tính đường bao. Sau đó, thiết lập cửa sổ.
- Thực hiện thủ tục vẽ hình một lần nữa. Nhưng lần này thực hiện thao tác vẽ.



PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN

Thiết lập cửa sổ và khung nhìn tự động

- ❑ Thiết lập khung nhìn bảo toàn tỷ lệ khoảng cách
 - $R > W/H$: setViewport(0, W, 0, W/R);
 - $R < W/H$: setViewport(0, H*R, 0, H);



PHÉP ÁNH XẠ TỪ CỬA SỔ SANG KHUNG NHÌN

Thiết lập cửa sổ và khung nhìn tự động

Sự kiện Resize

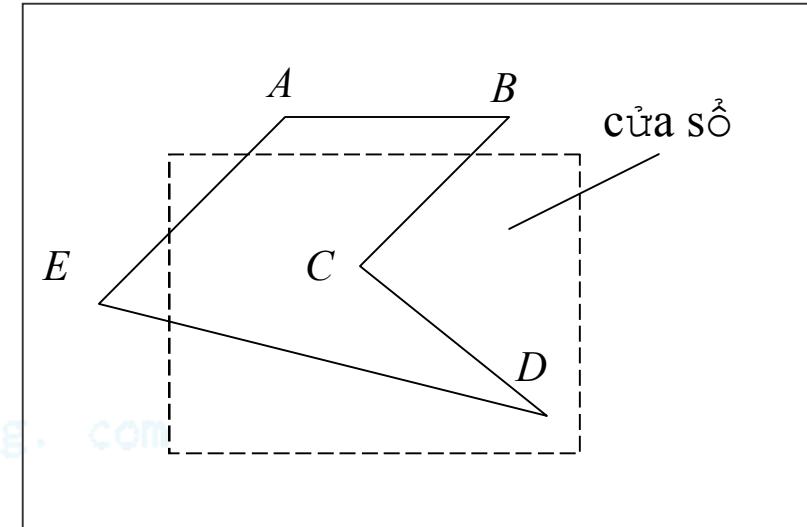
```
- glutReshapeFunc (myReshape) ;  
- void myReshape (GLsizei W, GLsizei H) ;  
- void myReshape (GLsizei W, GLsizei H)  
{  
    if (R > W/H) // R là biến toàn cục, R=hệ số tỷ lệ của cửa sổ  
        setViewport (0, W, 0, W/R) ;  
    else  
        setViewport (0, H*R, 0, H) ;  
}
```

GIẢI THUẬT CẮT XÉN

Cắt xén đoạn thẳng

Xây dựng hàm `clipSegment(p1, p2, window)` trả về 0 nếu đoạn thẳng nằm ngoài cửa sổ, trả về 1 trong các trường hợp còn lại.

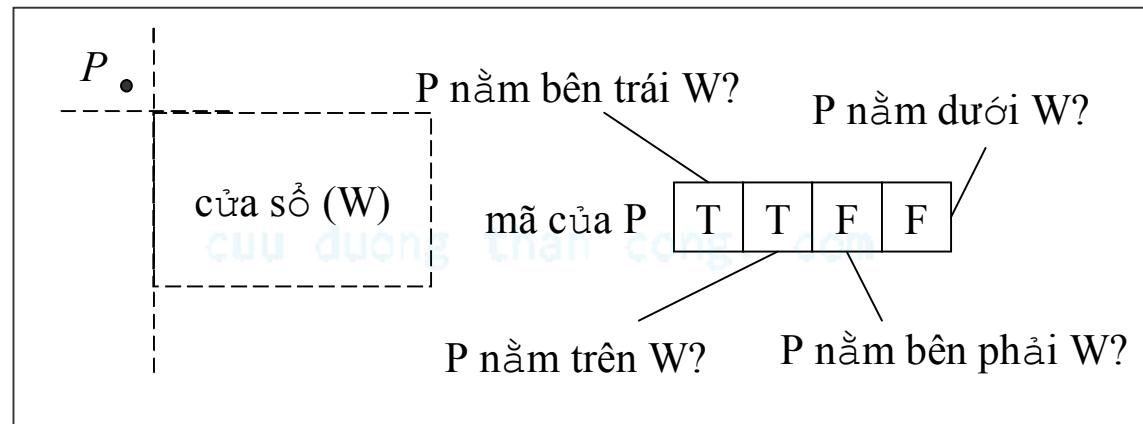
- Nằm trong cửa sổ (CD), trả về 1.
- Nằm ngoài cửa sổ (AB), trả về 0.
- Một đầu mút nằm trong cửa sổ, một đầu mút nằm ngoài (ED), cắt bỏ phần nằm ngoài và trả về 1.
- Hai đầu mút nằm ngoài, một phần đoạn thẳng nằm bên trong (EA), cắt bỏ phần nằm ngoài và trả về 1.



GIẢI THUẬT CẮT XÉN

Giải thuật Cohen-Sutherland

- ☐ Mã trong ngoài cửa sổ: mã hóa vị trí của điểm so với cửa sổ



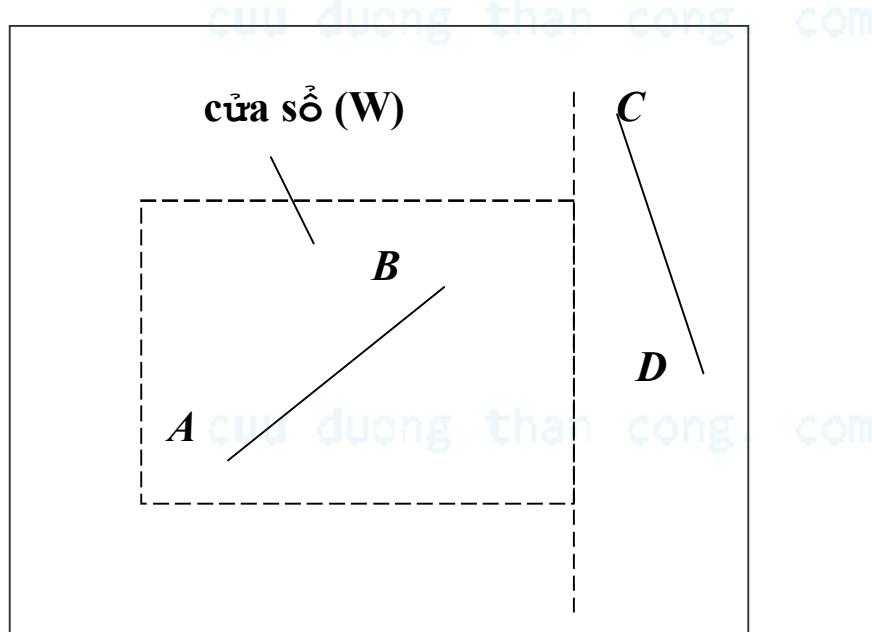
TTFF	FTFF	FTTF
TTFF	FFFF cửa sổ	FFTF
TFFT	FFFT	FFTT

GIẢI THUẬT CẮT XÉN

Giải thuật Cohen-Sutherland

Trường hợp chấp nhận đơn giản và loại bỏ đơn giản

- Chấp nhận đơn giản (AB), dùng cửa sổ lớn. Mã của hai đầu mút đều là FFFF.
- Loại bỏ đơn giản (CD), dùng cửa sổ nhỏ. Mã hai đầu mút đều có cùng giá trị T ở một trường



GIẢI THUẬT CẮT XÉN

Giải thuật Cohen-Sutherland

- ❑ Các trường hợp còn lại:

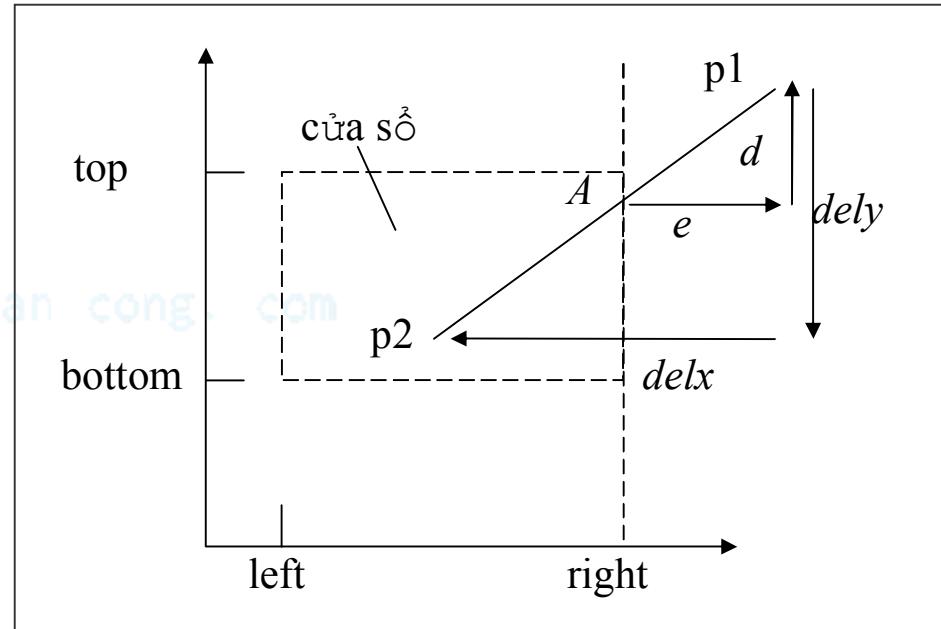
$$\frac{d}{dely} = \frac{e}{delx}$$

$$e = p1.x - W.right;$$

$$delx = p2.x - p1.x;$$

$$dely = p2.y - p1.y;$$

cuu duong thanh cong, com

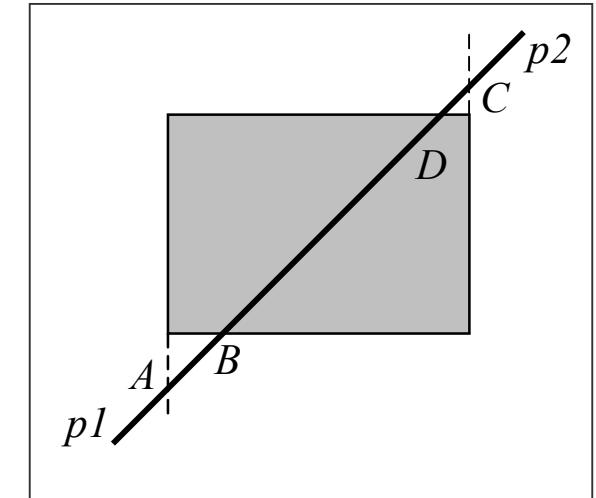


$$p1.y = p1.y + (W.right - p1.x) * dely / delx$$

cuu duong thanh cong, com

GIẢI THUẬT CẮT XÉN

```
int clipSegment(Point2& p1, Point2& p2, RealRect W)
{
    do{
        if (trivial accept) return 1;
        if (trivial reject) return 0;
        if (p1 nằm ngoài)
        {
            if (p1 nằm bên trái) cắt xén p1 với cạnh trái
            else if (p1 nằm bên phải) cắt xén p1 với cạnh phải
            else if (p1 nằm dưới) cắt xén p1 với cạnh dưới
            else if (p1 nằm trên) cắt xén p1 với cạnh trên
        }
        else      //p2 nằm ngoài
        {
            if (p2 nằm bên trái) cắt xén p2 với cạnh trái
            else if (p2 nằm bên phải) cắt xén p2 với cạnh phải
            else if (p2 nằm dưới) cắt xén p2 với cạnh dưới
            else if (p2 nằm trên) cắt xén p2 với cạnh trên
        }
    }while(1);
}
```



XÂY DỰNG LỚP CANVAS

Mục đích:

- Cung cấp những tiện ích để vẽ các đối tượng như đường thẳng, đa giác v.v
- Cung cấp cách làm đơn giản để tạo cửa sổ ứng dụng, thiết lập cửa sổ khung nhìn, thiết lập ánh xạ biến đổi từ cửa sổ sang khung nhìn, cùng với những tiện ích trong đồ họa con rùa

cuuduongthancong.com

XÂY DỰNG LỚP CANVAS

Các lớp hỗ trợ

```
class Point2
{
    public:
        Point2() { x = y = 0.0f; } // constructor
        Point2(float xx, float yy) { x = xx; y = yy; }
        void set(float xx, float yy) { x = xx; y = yy; }
        float getX() { return x; }
        float getY() { return y; }
        void draw() { glBegin(GL_POINTS);
                      glVertex2f((GLfloat)x, (GLfloat)y);
                      glEnd();
        }

    private:
        float x, y;
};
```

XÂY DỰNG LỚP CANVAS

Các lớp hỗ trợ

```
class IntRect
{
    public:
        IntRect() { l = 0; r = 100; b = 0; t = 100; }
        IntRect( int left, int right, int bottom, int top )
            { l = left; r = right; b = bottom; t = top; }
        void set( int left, int right, int bottom, int top )
            { l = left; r = right; b = bottom; t = top; }
        void draw(); //draw this rectangle using OpenGL
    private:
        int l, r, b, t;
};

class RealRect
{
    giống như lớp intRect ngoại trừ dùng
    float thay cho int
};
```

XÂY DỰNG LỚP CANVAS

```
class Canvas{  
public:  
    Canvas(int width, int height, char* windowTitle);  
    void setWindow(float l, float r, float b, float t);  
    void setViewport(int l, int r, int b, int t);  
    IntRect getViewport();  
    RealRect getWindow();  
    float getWindowAspectRatio();  
    void clearScreen();  
    void setBackgroundColor(float r, float g, float b);  
    void setColor(float r, float g, float b);  
    void lineTo(float x, float y);  
    void lineTo(Point2 p);  
    void moveTo(float x, float y);  
    void moveTo(Point2 p);  
    những phương thức khác sẽ được định nghĩa sau  
private:  
    Point2 CP; //current position in the world  
    IntRect viewport; // the current viewport  
    RealRect window; // the current window  
    những biến thành viên khác sẽ được định nghĩa sau };
```

XÂY DỰNG LỚP CANVAS

```
Canvas::Canvas(int width, int height, char* windowTitle)
{
    char* argv[1];
    char    dummyString[8];
    argv[0] = dummyString;
    int      argc = 1;
    glutInit(&argc, argv); //initialize the tool kit
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); //set the display mode
    glutInitWindowSize(width, height); //set window size
    glutInitWindowPosition(20, 20); // set window position on screen
    glutCreateWindow(windowTitle); // open the screen window
    setWindow(0, (float)width, 0, (float)height); //default window
    setViewport(0, width, 0, height); //default viewport
    CP.set(0.0f, 0.0f); //initialize the CP to (0, 0)
};
```

XÂY DỰNG LỚP CANVAS

```
void Canvas::moveTo(float x, float y)
{
    CP.set(x, y);
}

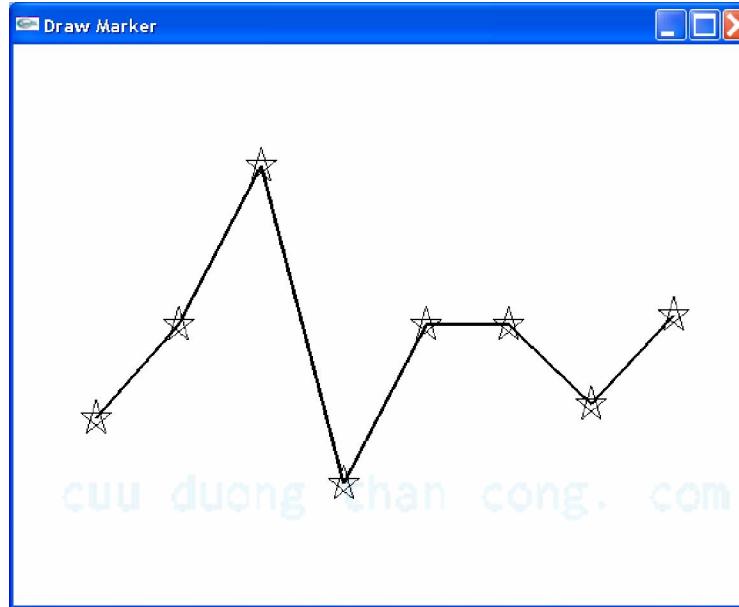
void Canvas::lineTo(float x, float y)
{
    glBegin(GL_LINE);
        glVertex2f((GLfloat) CP.getX(), (GLfloat) CP.getY());
        glVertex2f((GLfloat) x, (GLfloat) y);
    glEnd();
    CP.set(x, y);
    glFlush();
}

void Canvas::setWindow(float l, float r, float b, float t)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D((GLdouble)l,(GLdouble)r,(GLdouble)b,(GLdouble)t);
    window.set(l, r, b, t);
}
```

XÂY DỰNG LỚP CANVAS

```
Canvas cvs(640, 480, "try out Canvas");
void display()
{
    cvs.clearScreen(); //xóa màn hình
    cvs.setWindow(-10.0, 10.0, -10.0, 10.0);
    cvs.setViewport(10, 460, 10, 460);
    cvs.moveTo(0, -10.0); // vẽ đoạn thẳng
    cvs.lineTo(0, 10.0);
    RealRect box(-2.0, 2.0, -1.0, 1.0); //tạo hình chữ nhật
    box.draw();
    .....
}
void main()
{
    cvs.setBackgroundColor(1.0, 1.0, 1.0);
    cvs.setColor(0.0, 0.0, 0.0);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
}
```

VẼ TƯƠNG ĐỐI



```
moveTo(first data point);
drawMarker();
for(each remaining data point)
{
    lineTo(the next point);
    drawMarker();
}
```

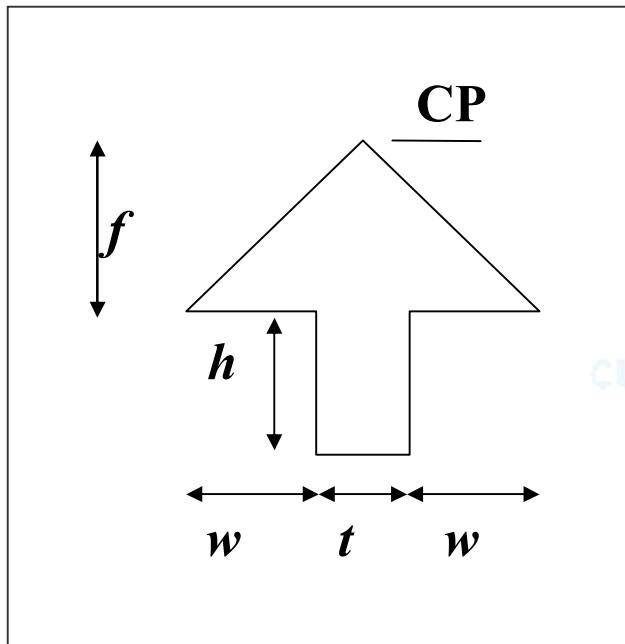
VẼ TƯƠNG ĐỐI

```
void Canvas::moveRel(float dx, float dy)
{
    CP.set(CP.getX() + dx, CP.getY() + dy);
}

void Canvas::lineRel(float dx, float dy)
{
    float x =CP.getX() + dx;
    float y =CP.getY() + dy;
    lineTo(x, y);
    CP.set(x, y);
}
```

cuu duong than cong. com

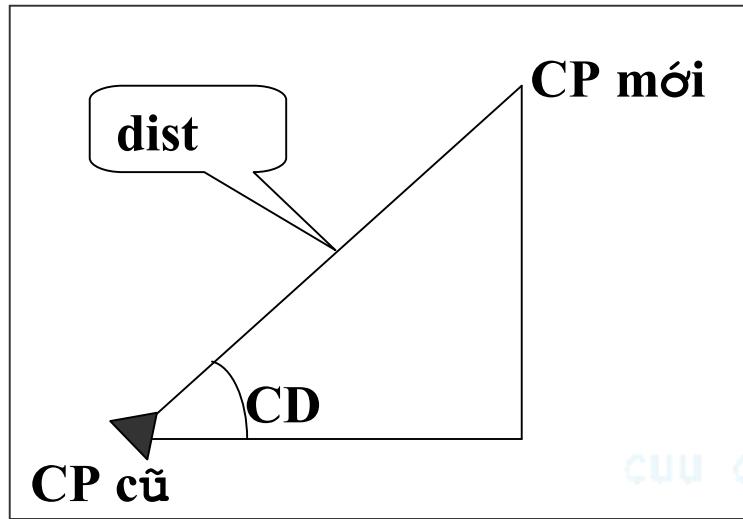
VẼ TƯƠNG ĐỐI



```
void arrow(float f,float h,  
          float t,float w)  
{  
    cvs.lineRel(-w-t/2, -f);  
    cvs.lineRel(w, 0);  
    cvs.lineRel(0, -h);  
    cvs.lineRel(t, 0);  
    cvs.lineRel(0, h);  
    cvs.lineRel(w, 0);  
    cvs.lineRel(-w-t/2, f);  
}
```

CuuDuongThanCong.com

ĐỒ HỌA CON RÙA



Thêm vào lớp Canvas:

- Biến CD chứa hướng hiện hành
- turnTo(float angle)

```
CD = angle;
```
- turn(float angle)

```
CD += angle; (CCW)
```
- forward(float dist,int isVisible)

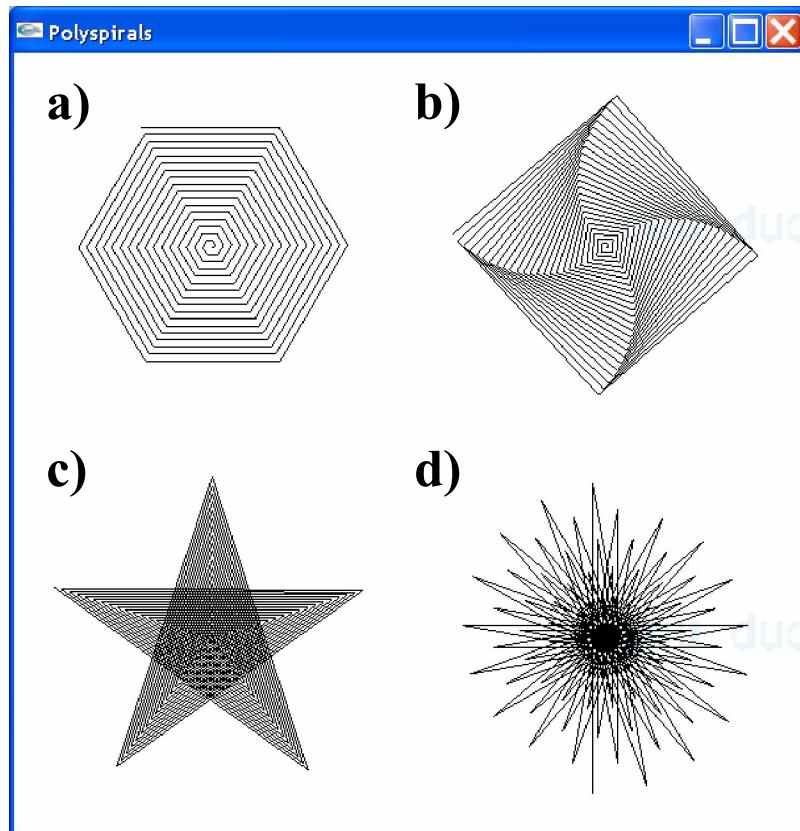
```
void Canvas::forward(float dist, int isVisible) {  
    const float RadPerDeg=0.017453393;  
    float x = CP.getX() + dist*cos(RadPerDeg *CD) ;  
    float y = CP.getY() + dist*sin(RadPerDeg *CD) ;  
    if( isVisible)      lineTo(x, y);  
    else                moveTo(x, y);  
}
```

```
void Canvas::forward(float dist, int isVisible) {  
    const float RadPerDeg=0.017453393;  
    float x = CP.getX() + dist*cos(RadPerDeg *CD) ;  
    float y = CP.getY() + dist*sin(RadPerDeg *CD) ;  
    if( isVisible)      lineTo(x, y);  
    else                moveTo(x, y);  
}
```

ĐỒ HỌA CON RÙA

Ví dụ:

Vẽ polyspirals



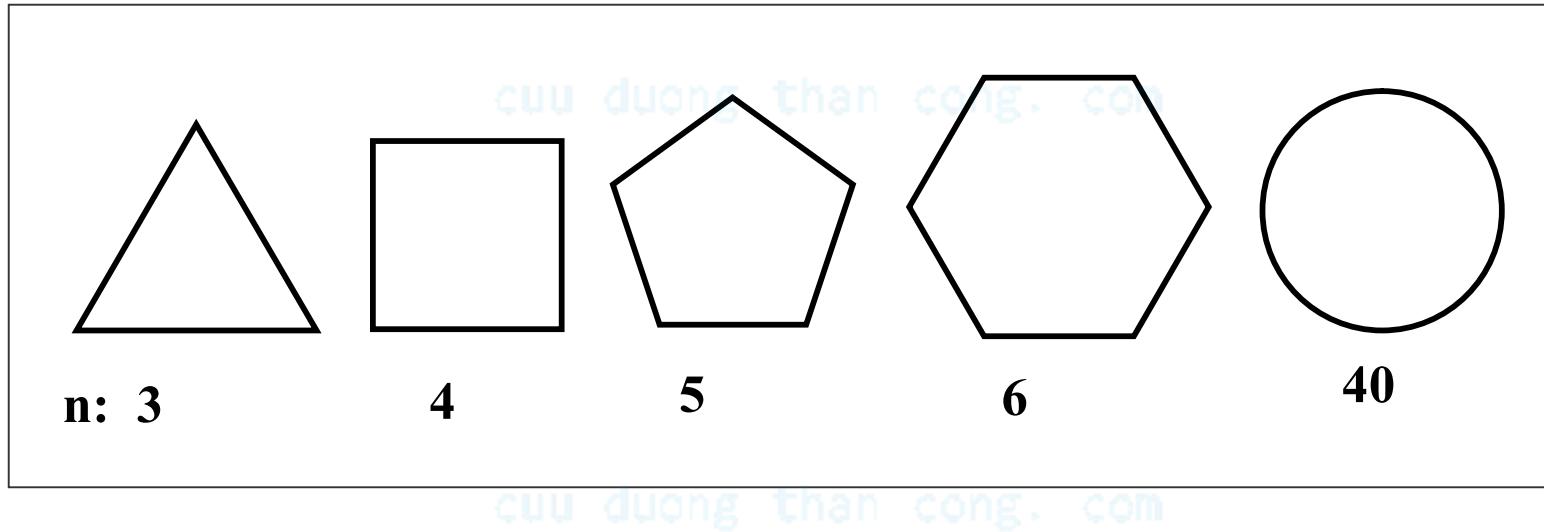
```
for(some number of iteration) {  
    //draw a line in current  
    forward(length, 1);  
    //turn through angle degreee  
    directionturn(angle);  
    // increment the line length  
    length += increment;  
}
```

- (a) 60°
- (b) 89.5°
- (c) -144°
- (d) 170°

TAO HÌNH ẢNH TỪ ĐA GIÁC ĐỀU

Đa giác đều

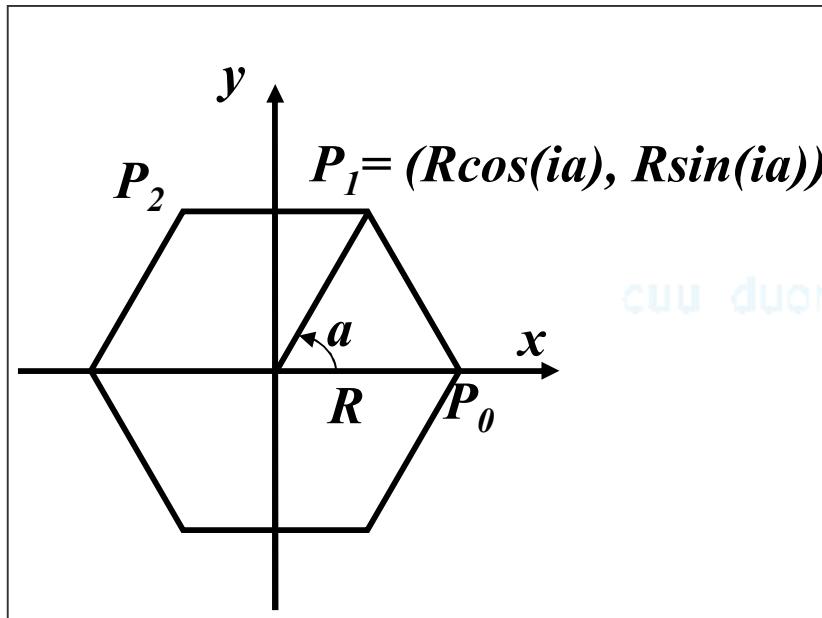
- Định nghĩa: đa giác đơn, các cạnh bằng nhau, hai cạnh kề nhau hợp với nhau một góc bằng nhau.



TAO HÌNH ẢNH TỪ ĐA GIÁC ĐỀU

Vẽ đa giác đều

$P_i = (R\cos(2\pi i/n), R\sin(2\pi i/n))$ với $i = 0, 1, \dots, n-1$.



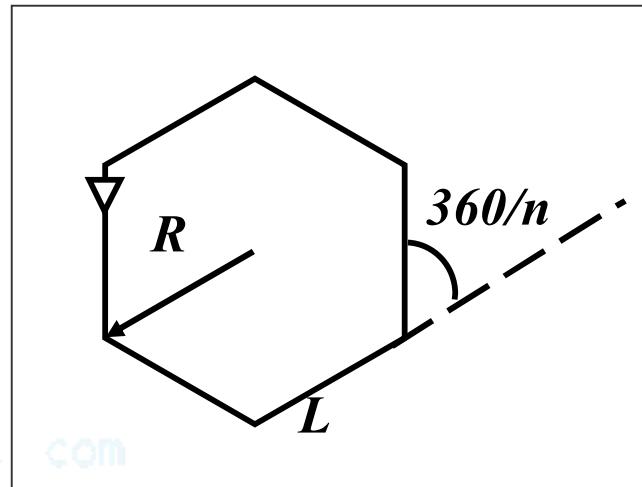
```
void ngon(int n,
          float cx, float cy,
          float r, float rotA) {
    if (n < 3) return;
    double angle = rotA*PI/180;
    double angleInc = 2*PI/n;
    cvs.moveTo(r*cos(angle)+cx,
               r*sin(angle)+cy);
    for(int k=0;k<n;k++)
    {
        angle += angleInc;
        cvs.lineTo(r*cos(angle)+cx,
                   r*sin(angle)+cy);
    }
}
```

TAO HÌNH ẢNH TỪ ĐA GIÁC ĐỀU

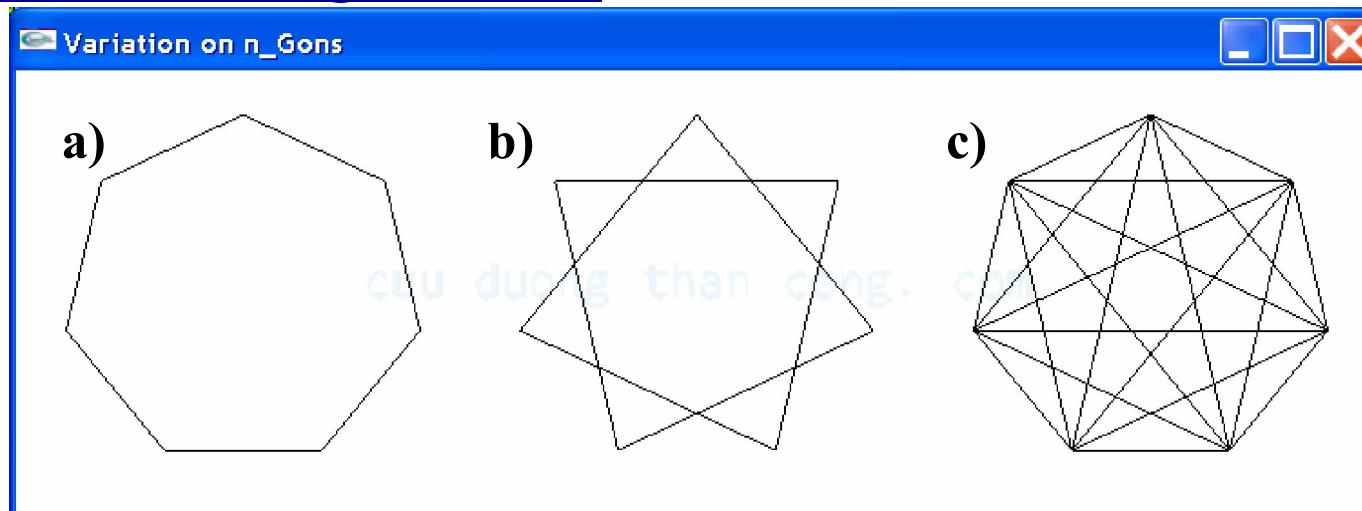
Vẽ đa giác bằng đồ họa con rùa

```
for(i=0;i<6;i++)  
{  
    cvs.forward(L, 1);  
    cvs.turn(60);  
}
```

cuu duong than cong. com



Biến thể của đa giác đều



VẼ ĐƯỜNG TRÒN VÀ CUNG TRÒN

❑ Vẽ đường tròn

```
void drawCircle(Point2 center, float radius)
{
    const int numVerts = 50;
    ngon(numVerts,center.getX(),center.getY(),radius,0);
}
```

❑ Cách chỉ định một đường tròn

- Tâm và bán kính
- Tâm và một điểm nằm trên đường tròn
- Ba điểm nằm trên đường tròn

VẼ ĐƯỜNG TRÒN VÀ CUNG TRÒN

☐ Vẽ cung tròn

```
void drawArc(Point2 center, float r,  
             float rotA, float sweep)
```

```
{
```

```
    const int n=30; //number segments  
    float angle = rotA*PI/180;  
    float angleInc = 2*PI/n;  
    float cx=center.getX(), cy=center.getY();  
    cvs.moveTo(r*cos(angle)+cx, r*sin(angle)+cy);  
    for(int k=1;k<n;k++)
```

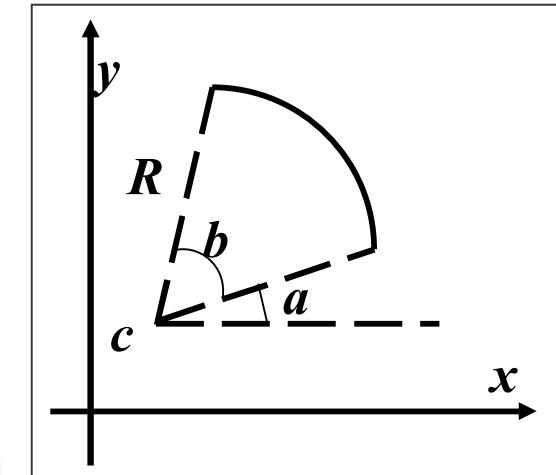
```
{
```

```
    angle += angleInc;  
    cvs.lineTo(r*cos(angle)+cx, r*sin(angle)+cy);
```

```
}
```

```
}
```

cuu duong than cong. com



DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

- **Dạng ẩn:** mô tả đường cong bằng hàm $F(x, y)$. Hàm này cho biết mối quan hệ giữa x và y : điểm (x, y) nằm trên đường cong nếu và chỉ nếu $F(x, y) = 0$.
 - $F(x, y) = (y - Ay)(Bx - Ax) - (x - Ax)(By - Ay)$ (đthẳng)
 - $F(x, y) = x^2 + y^2 - R^2$ (đtròn)
- **Hàm trong ngoài**
 - $F(x, y) = 0$, nếu (x, y) nằm trên đường cong
 - $F(x, y) > 0$, nếu (x, y) nằm ngoài đường cong
 - $F(x, y) < 0$, nếu (x, y) nằm trong đường cong
- **Nhược điểm của dạng ẩn**
 - Đối với hàm đa trị, không thể suy ra $y=g(x)$ từ $F(x, y)$, chẳng hạn từ dạng ẩn của đường tròn, ta có:

$$y = \pm\sqrt{R^2 - x^2}$$

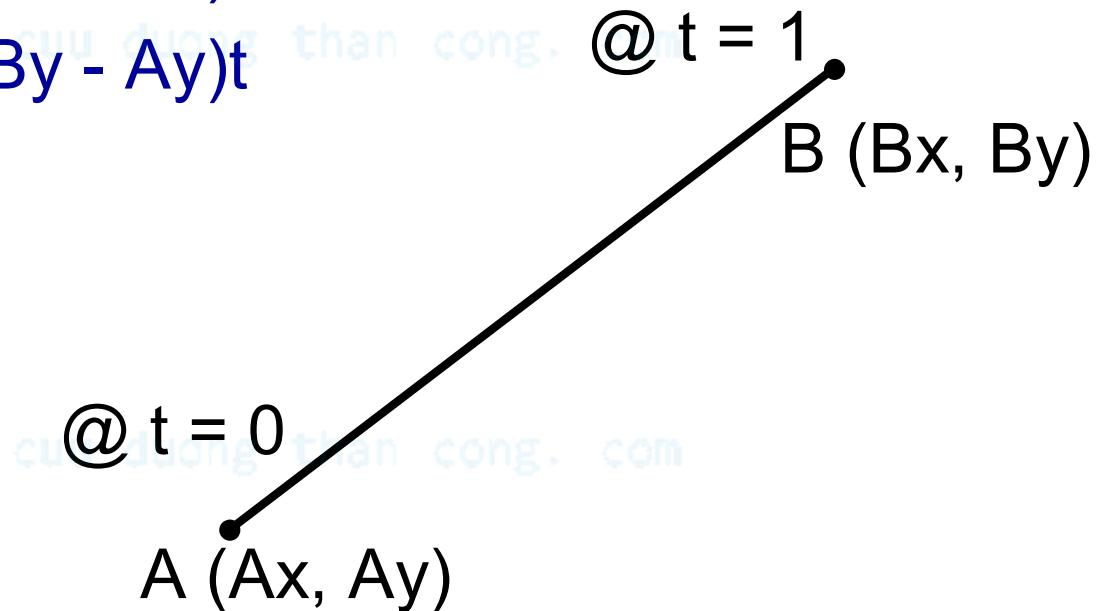
DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

Dạng biểu diễn tham số

- Ví dụ 1: Đoạn thẳng có hai đầu mút là A và B. Ở thời điểm $t = 0$, đi qua điểm A; ở thời điểm $t = 1$ qua điểm B.

$$x(t) = Ax + (Bx - Ax)t$$

$$y(t) = Ay + (By - Ay)t$$



DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

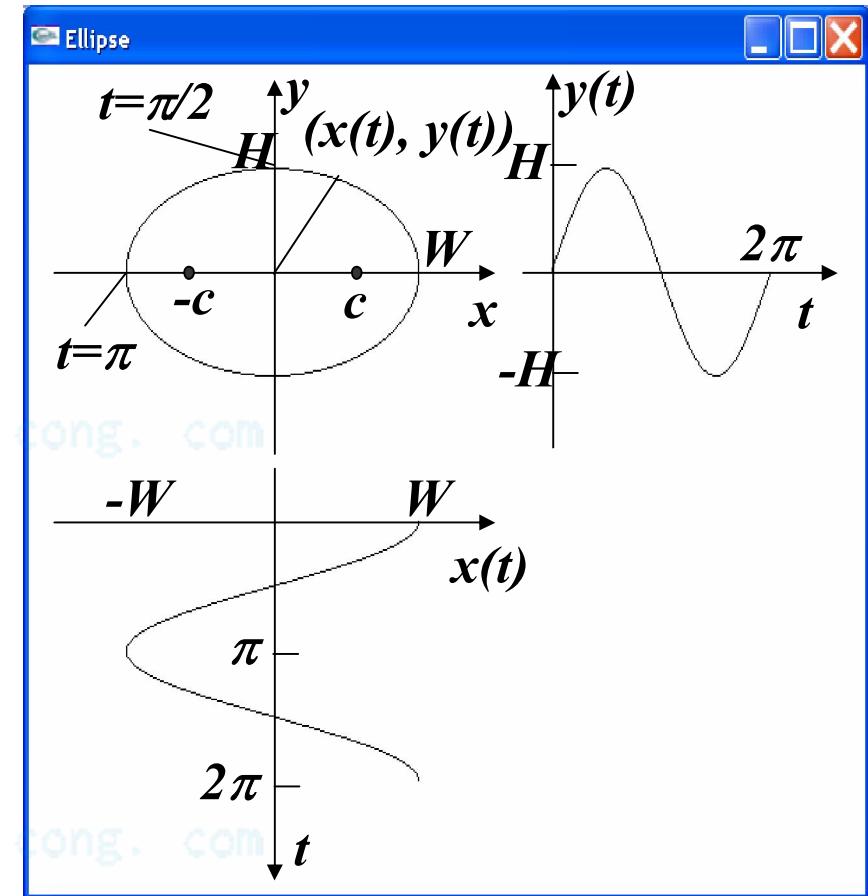
Dạng biểu diễn tham số

- Ví dụ 2: Đường ellipse có các bán kính là W và H

$$x(t) = W\cos(t)$$

$$y(t) = H\sin(t)$$

với ($0 \leq t \leq 2\pi$)



DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

Dạng biểu diễn tham số

- dạng ẩn và dạng tham số có cùng biểu diễn một đường cong hay không?
- từ dạng tham số tìm dạng ẩn?
 - Ví dụ: đối với hình ellipse

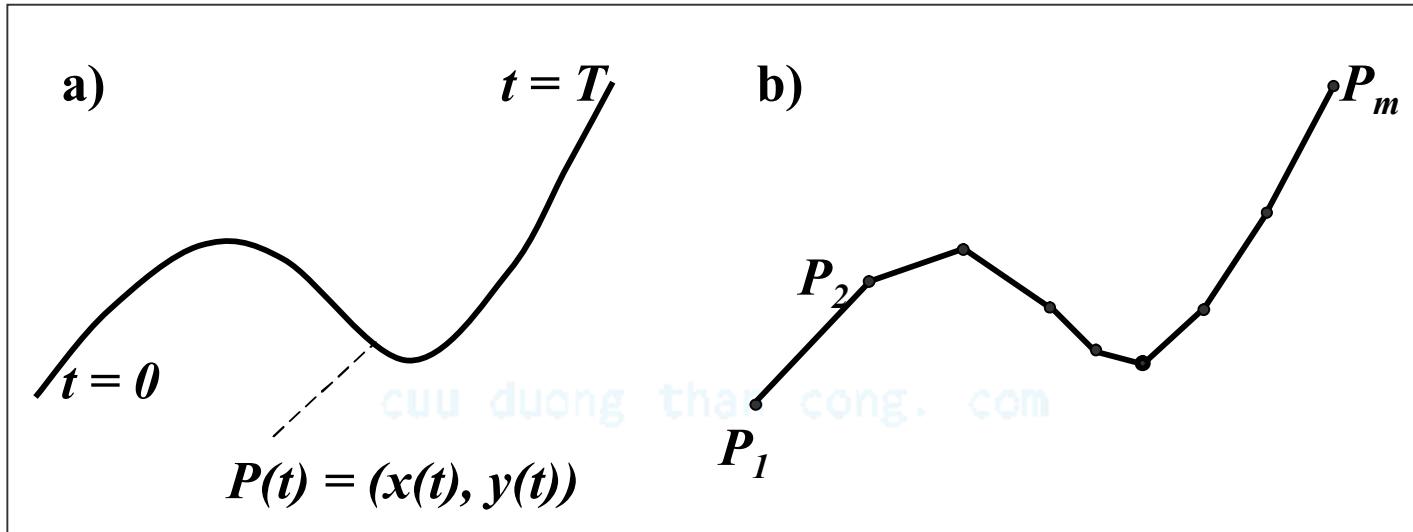
$$x(t) = W\cos(t) \rightarrow \cos(t) = x/W$$

$$y(t) = H\sin(t) \quad \sin(t) = y/H$$

$$\cos^2(t) + \sin^2(t) = 1 \rightarrow \left(\frac{x}{W}\right)^2 + \left(\frac{y}{H}\right)^2 = 1$$

DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

Vẽ đường cong biểu diễn dưới dạng tham số



```
//draw the curve (x(t), t(t)) using
//the array t[0], ..., t[n-1] of "sample-times"
glBegin(GL_LINES) ;
    for (int i=0; i<n; i++)
         glVertex2f(x(t[i]), y(t[i]));
glEnd() ;
```

DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

Superellipse

– Dạng ẩn

$$\left(\frac{x}{W}\right)^n + \left(\frac{y}{H}\right)^n = 1$$

– Dạng tham số

$$x(t) = W \cos(t) \left| \cos^{2/n-1}(t) \right|$$

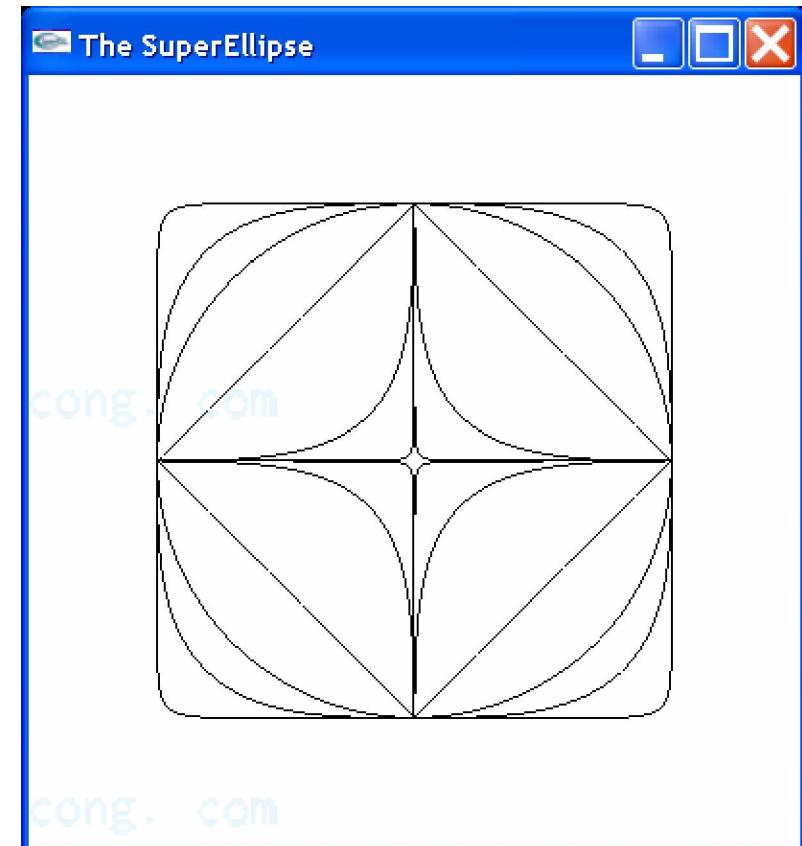
$$y(t) = H \sin(t) \left| \sin^{2/n-1}(t) \right|$$

$$n = 2m/(2n+1)$$

$n < 1$ co vào

$n > 1$ phình ra

$n = 1$ hình vuông



DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

Superhyperbola

– Dạng biểu diễn tham số

$$x(t) = W \sec(t) |\sec^{2/n-1}(t)|$$

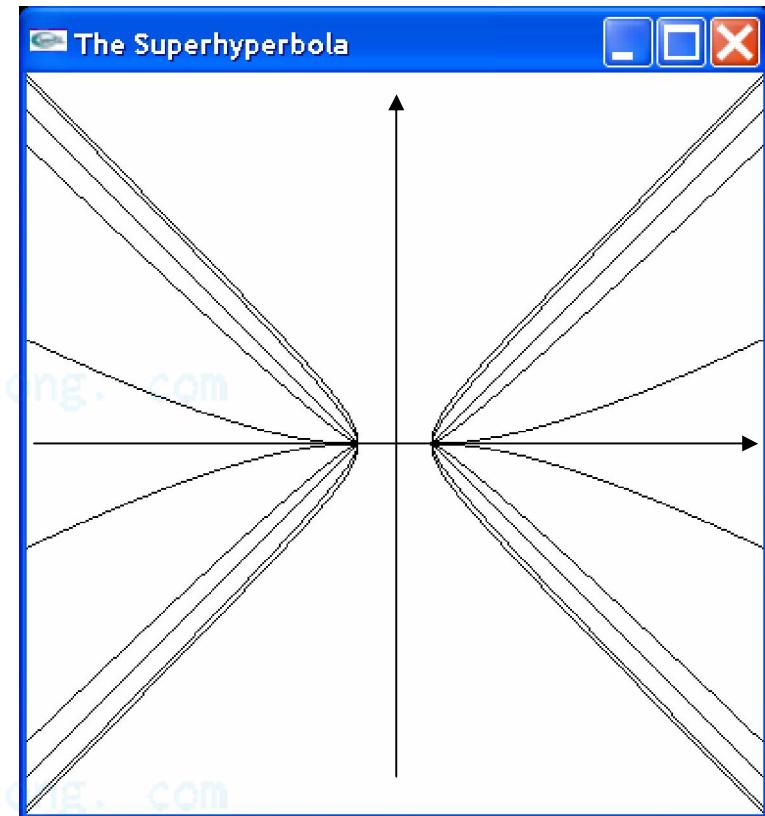
$$y(t) = H \tan(t) |\tan^{2/n-1}(t)|$$

$$n = 2m/(2n+1)$$

n < 1 co vào

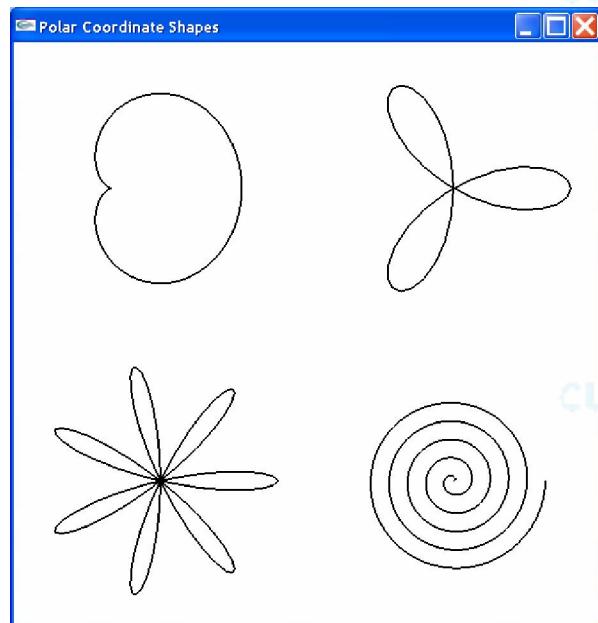
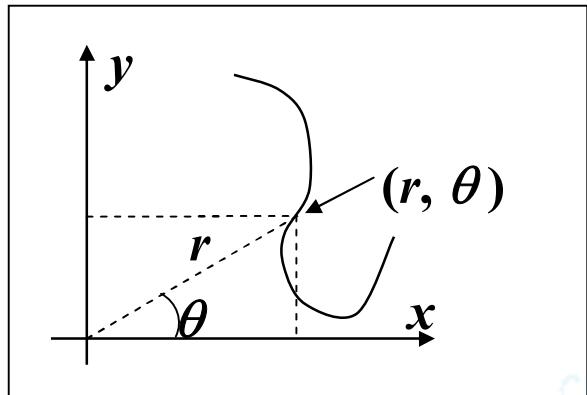
n > 1 phình ra

n = 1 đường thẳng



DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

Đường cong trong hệ tọa độ cực



$$x(t) = r(t) \cos(\theta(t))$$
$$y(t) = r(t) \sin(\theta(t))$$

$$x = f(\theta) \cos(\theta)$$
$$y = f(\theta) \sin(\theta)$$

Ví dụ

cardioid

$$f(\theta) = K(1 + \cos(\theta))$$

rose

$$f(\theta) = K \cos(n\theta)$$

Archimedean spiral

$$f(\theta) = K\theta$$

DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

Đường cong trong hệ tọa độ cực

- Mặt cắt nón (conic section)

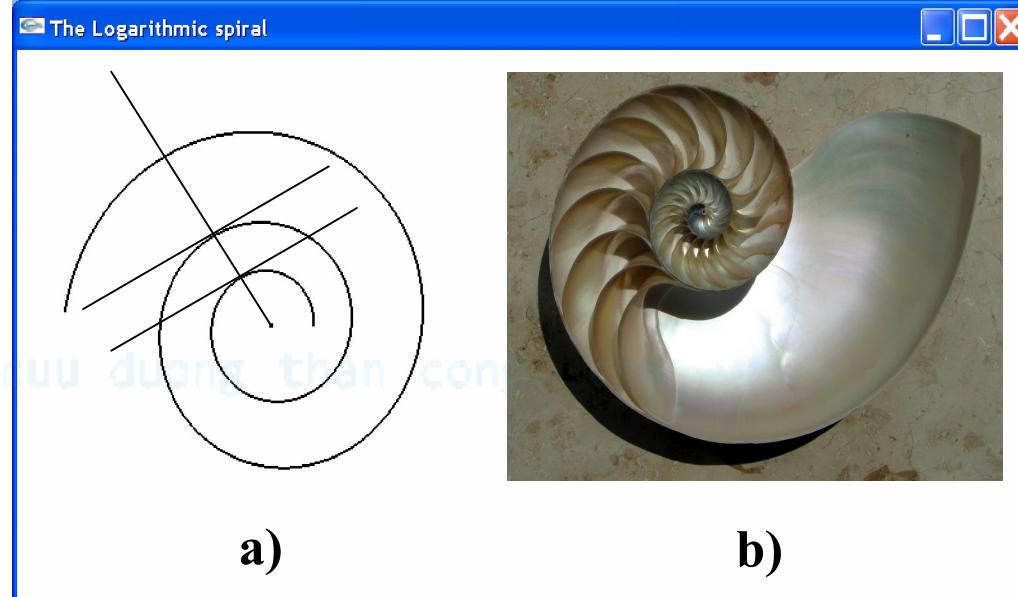
$$f(\theta) = \frac{1}{1 \pm a \cos(\theta)}$$

$a=1 \rightarrow$ parabola
 $0 \leq a < 1 \rightarrow$ ellipse
 $a > 1 \rightarrow$ hyperbola

- Đường xoắn ốc logarit

$$f(\theta) = K e^{a\theta}$$

$$a = \cot(\theta)$$



DẠNG BIỂU DIỄN THAM SỐ CỦA ĐƯỜNG CONG

Đường cong 3D

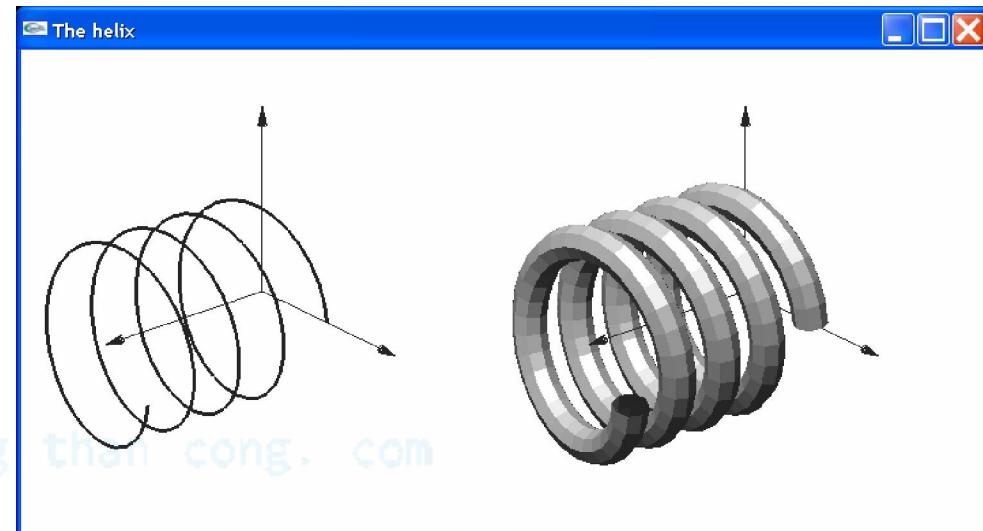
$$P(t) = (x(t), y(t), z(t))$$

Đường helix

$$x(t) = \cos(t)$$

$$y(t) = \sin(t)$$

$$z(t) = bt$$

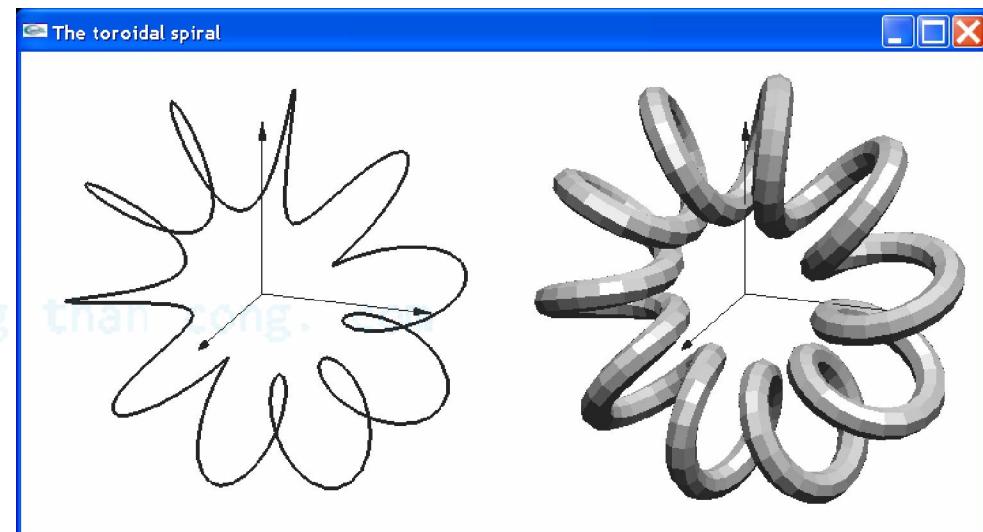


Đường toroidal spiral

$$x(t) = (a\sin(ct) + b)\cos(t),$$

$$y(t) = (a\sin(ct) + b)\sin(t),$$

$$z(t) = a\cos(ct)$$



Trường Đại Học Bách Khoa TP Hồ Chí Minh
Khoa Khoa học & Kỹ thuật Máy tính



ĐỒ HỌA MÁY TÍNH

CHƯƠNG 4:

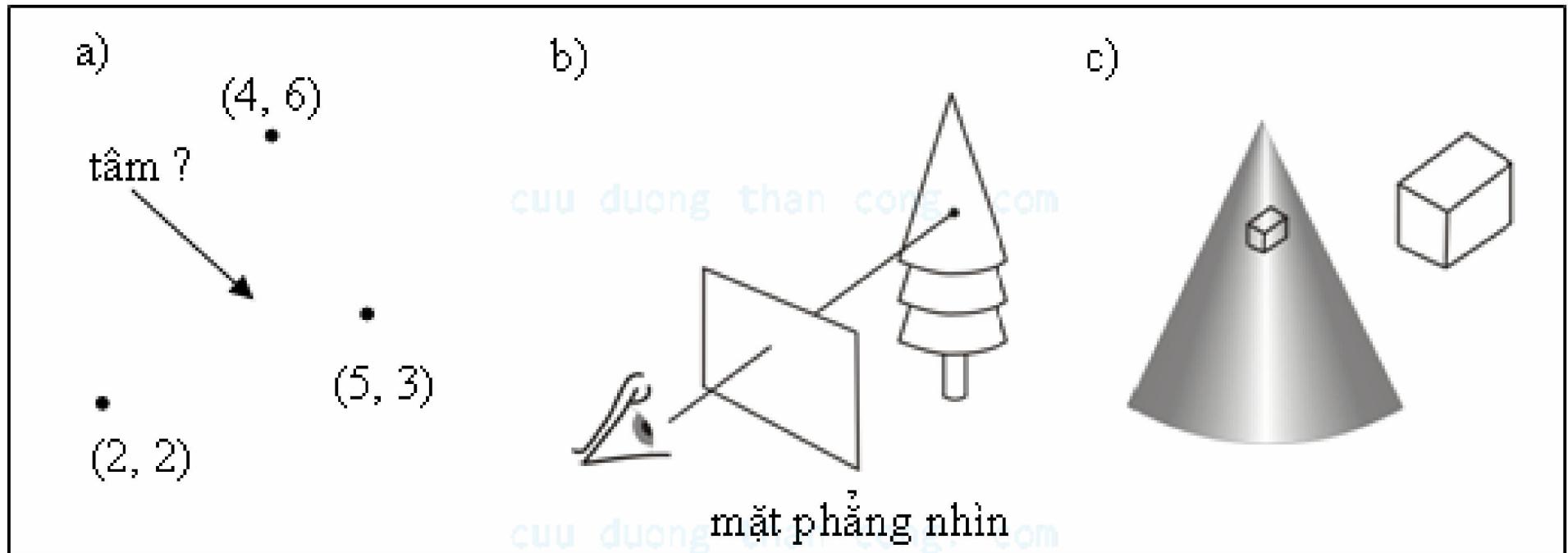
VECTOR TRONG ĐỒ HỌA MÁY TÍNH

NỘI DUNG TRÌNH BÀY

- ❑ Giới thiệu
- ❑ Ôn tập kiến thức về vector
- ❑ Tích vô hướng
- ❑ Tích có hướng
- ❑ Biểu diễn đối tượng hình học
- ❑ Giao của hai đoạn thẳng
- ❑ Đường tròn đi qua ba điểm
- ❑ Giao của đường thẳng và mặt phẳng
- ❑ Bài toán liên quan đến đa giác

GIỚI THIỆU

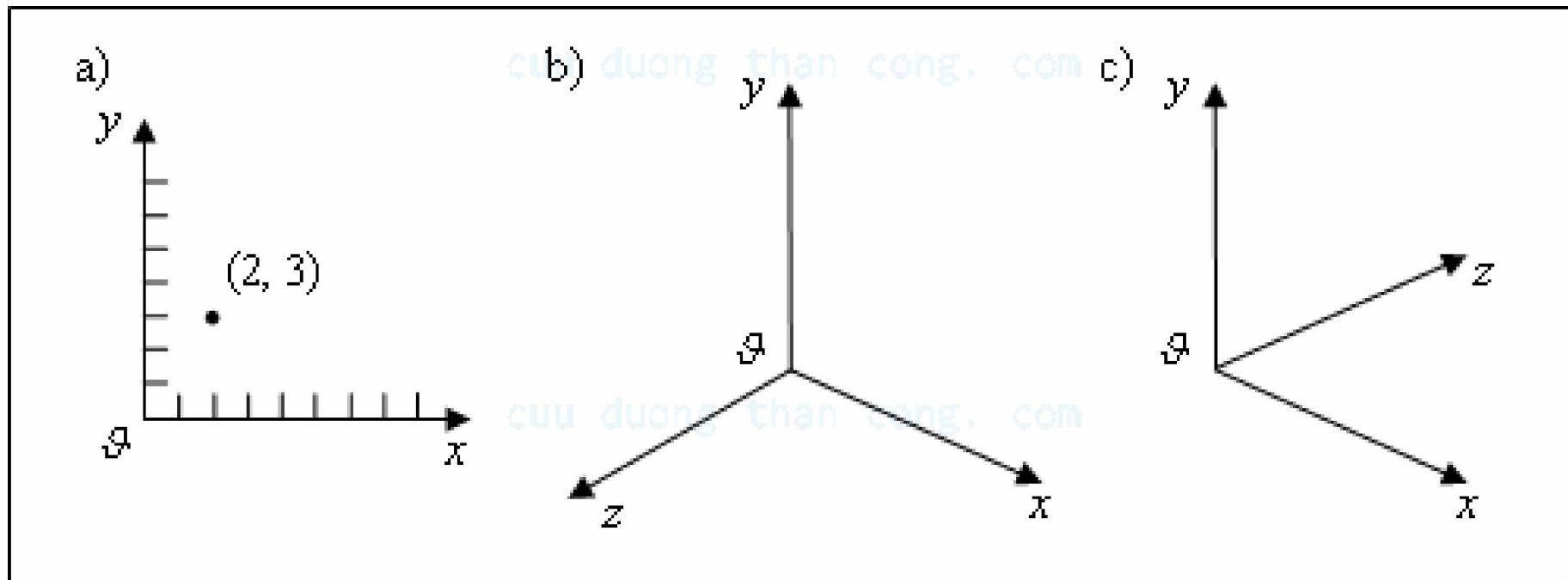
- ☐ Tại sao vector lại quan trọng trong đồ họa máy tính



GIỚI THIỆU

□ Hệ trục tọa độ

- Hệ trục tọa độ bàn tay phải (dùng trong toán học v.v)
- Hệ trục tọa độ bàn tay trái (trong đồ họa)
- Đơn vị của trục tọa độ không quan trọng



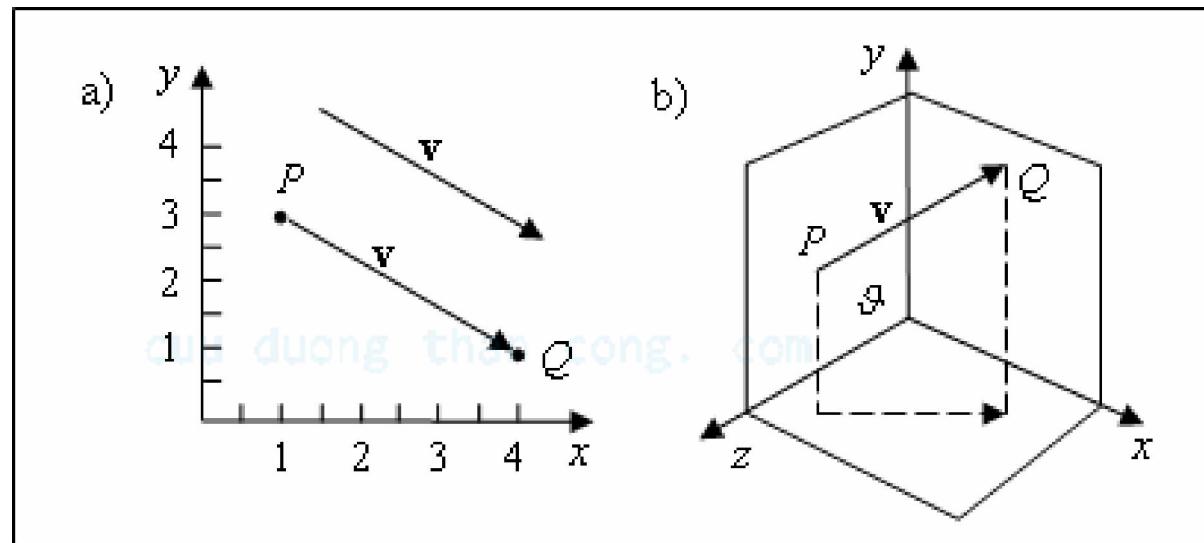
ÔN TẬP KIẾN THỨC VỀ VECTOR

Định nghĩa:

Vector là đại lượng có độ dài và hướng. Nó thường được dùng để biểu diễn các đại lượng vật lý như lực, vận tốc.

Lưu ý:

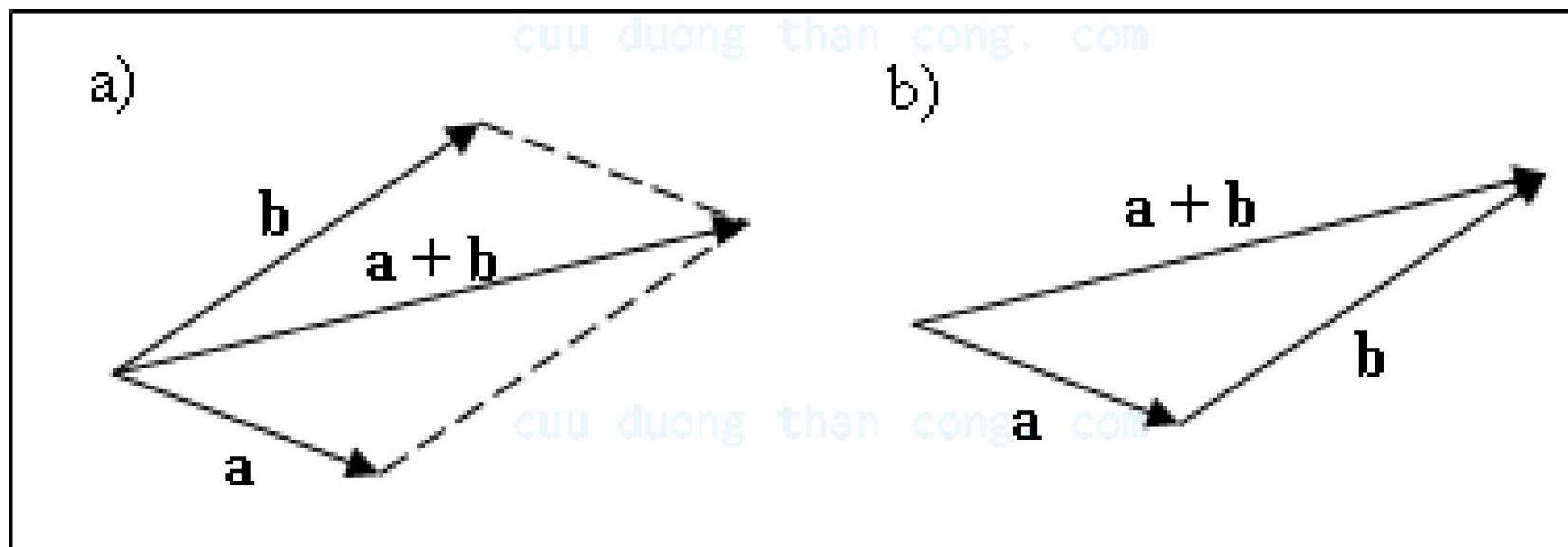
- Điểm đặt của vector không quan trọng
- Vector vị trí



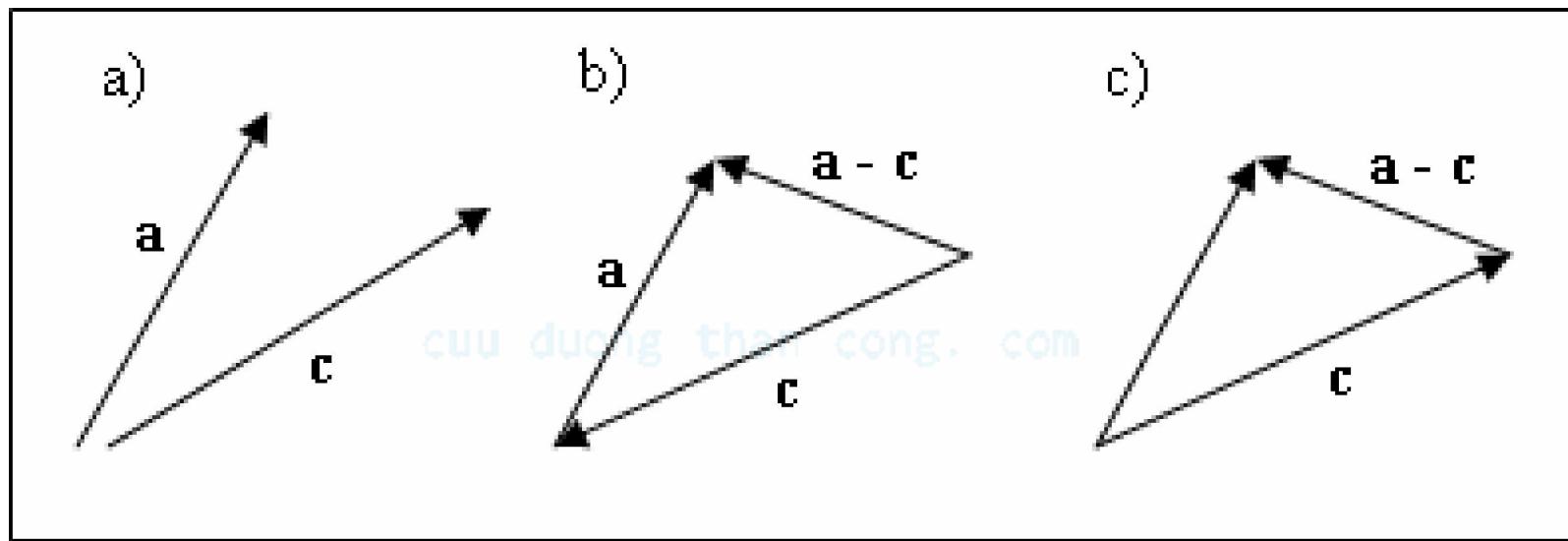
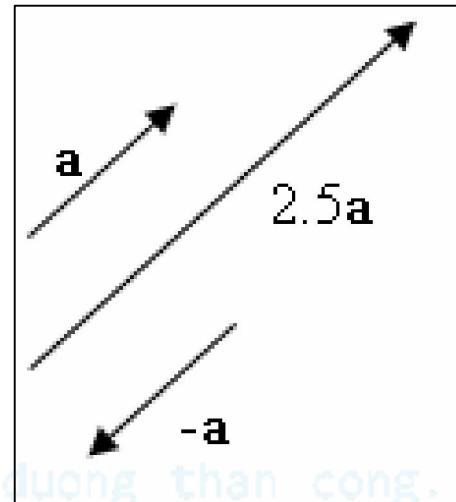
ÔN TẬP KIẾN THỨC VỀ VECTOR

$$\mathbf{a} = (2, 5, 6), \mathbf{b} = (-2, 7, 1)$$

- Phép cộng: $\mathbf{a} + \mathbf{b} = (0, 12, 7)$
- Phép nhân tỷ lệ: $6\mathbf{a} = (12, 30, 39)$
- Phép trừ: $\mathbf{a} - \mathbf{b} = \mathbf{a} + (-\mathbf{b}) = (4, -2, 5)$



ÔN TẬP KIẾN THỨC VỀ VECTOR



ÔN TẬP KIẾN THỨC VỀ VECTOR

- ☐ Tổ hợp tuyến tính của m vector v_1, v_2, \dots, v_m là vector

$$w = a_1v_1 + a_2v_2 + \dots + a_mv_m$$

(với a_1, a_2, \dots, a_m là các đại lượng vô hướng)

- ☐ Tổ hợp affine là tổ hợp tuyến tính với

$$a_1 + a_2 + \dots + a_m = 1$$

- ☐ Tổ hợp lồi là tổ hợp tuyến tính với

$$a_1 + a_2 + \dots + a_m = 1 \text{ và}$$

$$a_i \geq 0, \text{ với } i=1, \dots, m$$

- ☐ Độ lớn của vector: $|w| = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2}$

- ☐ Vector đơn vị: $\mathbf{u}_a = \frac{\mathbf{a}}{|\mathbf{a}|}$

TÍCH VÔ HƯỚNG

- Định nghĩa: Tích vô hướng d của hai vector n chiều $\mathbf{v} = (v_1, v_2, \dots, v_n)$ và $\mathbf{w} = (w_1, w_2, \dots, w_n)$ và được ký hiệu là $\mathbf{v} \bullet \mathbf{w}$ và có giá trị

$$d = \mathbf{v} \bullet \mathbf{w} = \sum_{i=1}^n v_i w_i$$

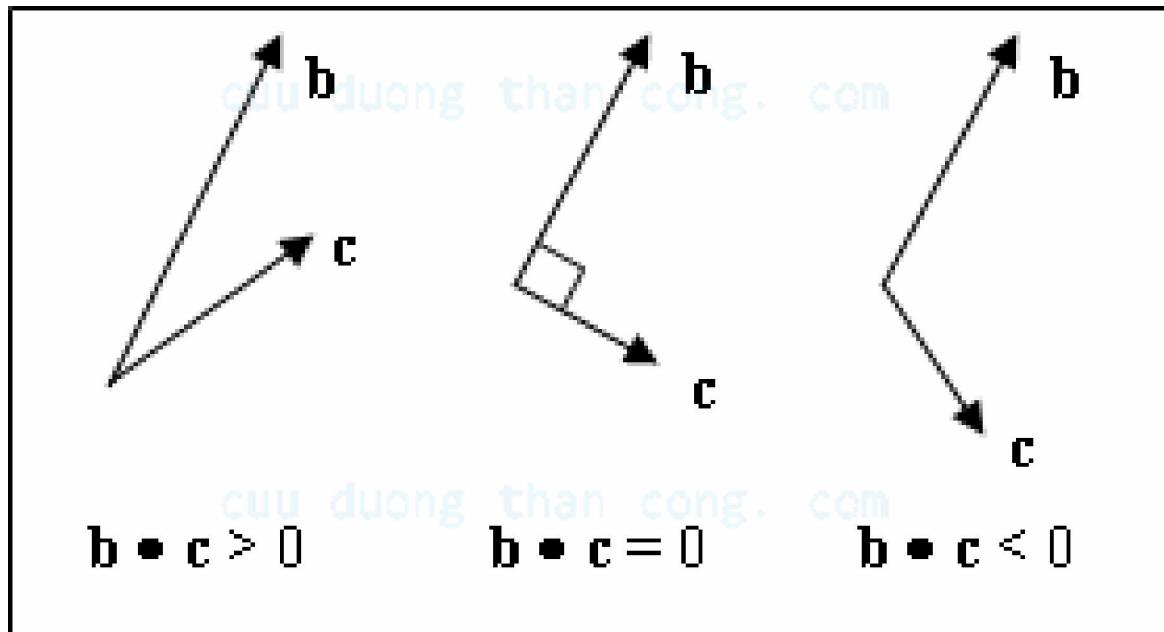
- Tính chất:
- Tính đối xứng (symmetry): $\mathbf{a} \bullet \mathbf{b} = \mathbf{b} \bullet \mathbf{a}$
 - Tính tuyến tính (linearity): $(\mathbf{a} + \mathbf{c}) \bullet \mathbf{b} = \mathbf{a} \bullet \mathbf{b} + \mathbf{c} \bullet \mathbf{b}$
 - Tính đồng nhất (homogeneity): $(s\mathbf{a}) \bullet \mathbf{b} = s(\mathbf{a} \bullet \mathbf{b})$
 - $|\mathbf{b}|^2 = \mathbf{b} \bullet \mathbf{b}$

TÍCH VÔ HƯỚNG

□ Góc giữa hai vector:

$$\mathbf{b} \bullet \mathbf{c} = |\mathbf{b}| |\mathbf{c}| \cos(\theta)$$

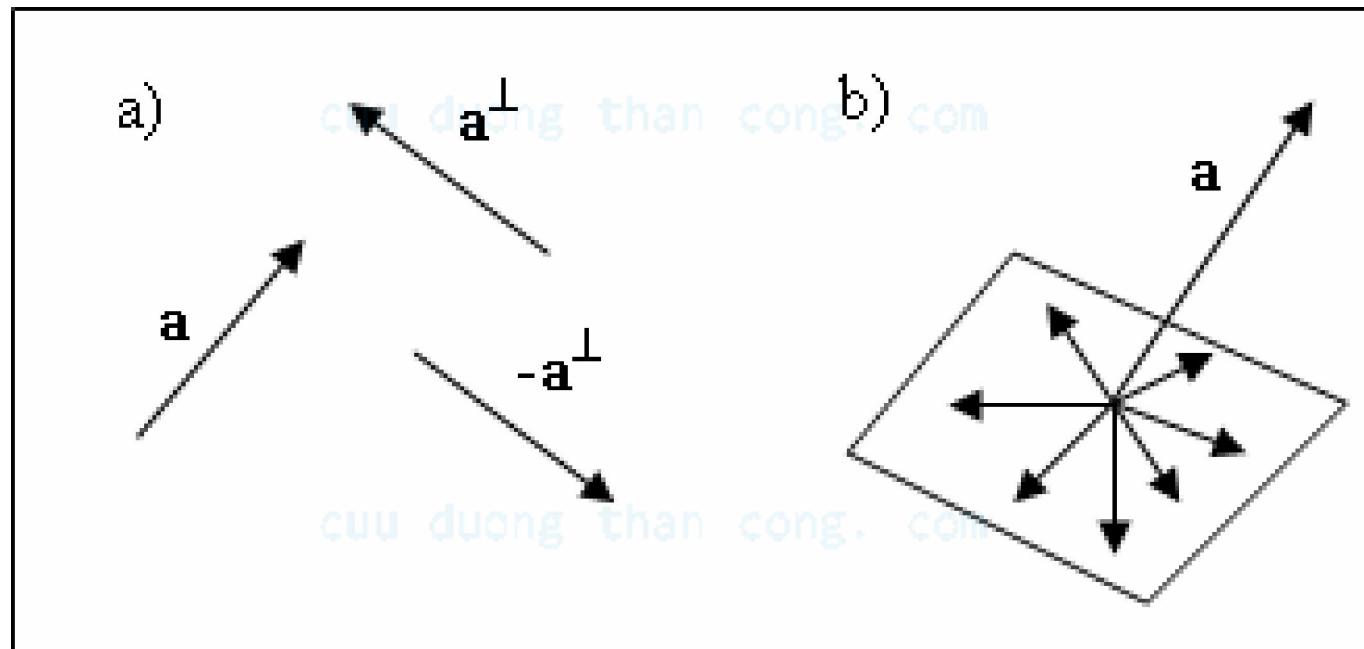
$$\cos(\theta) = \mathbf{u}_b \bullet \mathbf{u}_c$$



TÍCH VÔ HƯỚNG

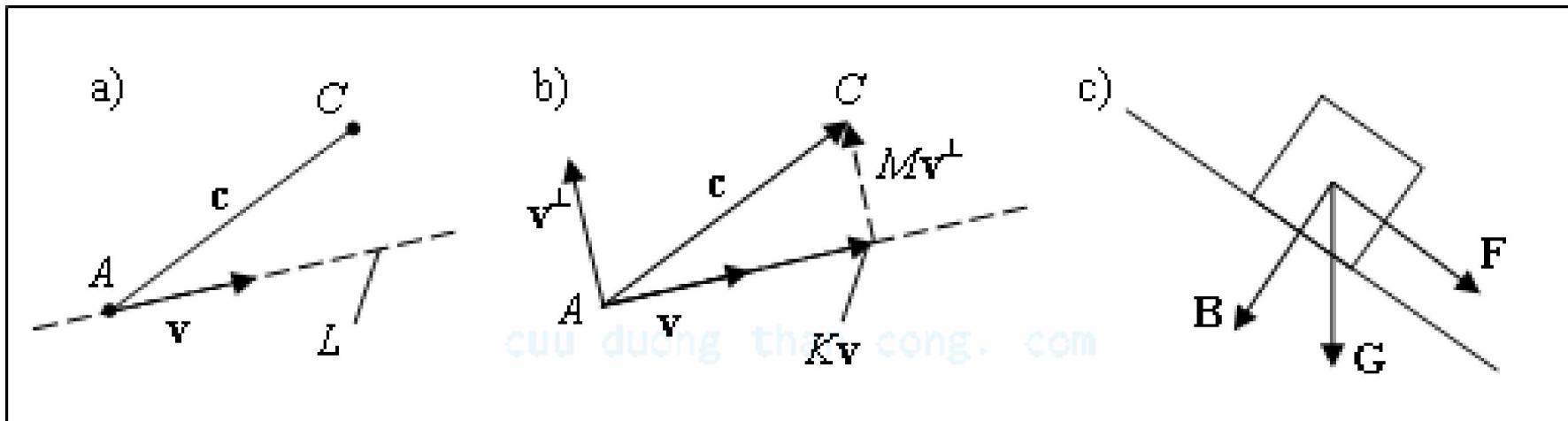
□ Vector vuông góc với vector 2 chiều

Cho $\mathbf{a} = (ax, ay)$. Thì $\mathbf{a}^\perp = (-ay, ax)$ là vector **vuông góc ngược chiều kim đồng hồ** với \mathbf{a} . Vector này thường được gọi là vector "perp" (viết tắt của perpendicular).



TÍCH VÔ HƯỚNG

□ Phép chiếu trực giao và khoảng cách từ một điểm đến đt



$$\mathbf{c} = K\mathbf{v} + M\mathbf{v}^\perp \quad (\text{cần xác định } K \text{ và } M)$$

$$\mathbf{c} \bullet \mathbf{v} = K\mathbf{v} \bullet \mathbf{v} + M\mathbf{v}^\perp \bullet \mathbf{v} \rightarrow K = \frac{\mathbf{c} \bullet \mathbf{v}}{\mathbf{v} \bullet \mathbf{v}} \quad M = \frac{\mathbf{c} \bullet \mathbf{v}^\perp}{\mathbf{v}^\perp \bullet \mathbf{v}^\perp}$$

$$\mathbf{c} = \left(\frac{\mathbf{v} \bullet \mathbf{c}}{|\mathbf{v}|^2} \right) \mathbf{v} + \left(\frac{\mathbf{v}^\perp \bullet \mathbf{c}}{|\mathbf{v}|^2} \right) \mathbf{v}^\perp \quad \text{distance} = \left| \frac{\mathbf{v}^\perp \bullet \mathbf{c}}{|\mathbf{v}|^2} \mathbf{v}^\perp \right| = \frac{|\mathbf{v}^\perp \bullet \mathbf{c}|}{|\mathbf{v}|}$$

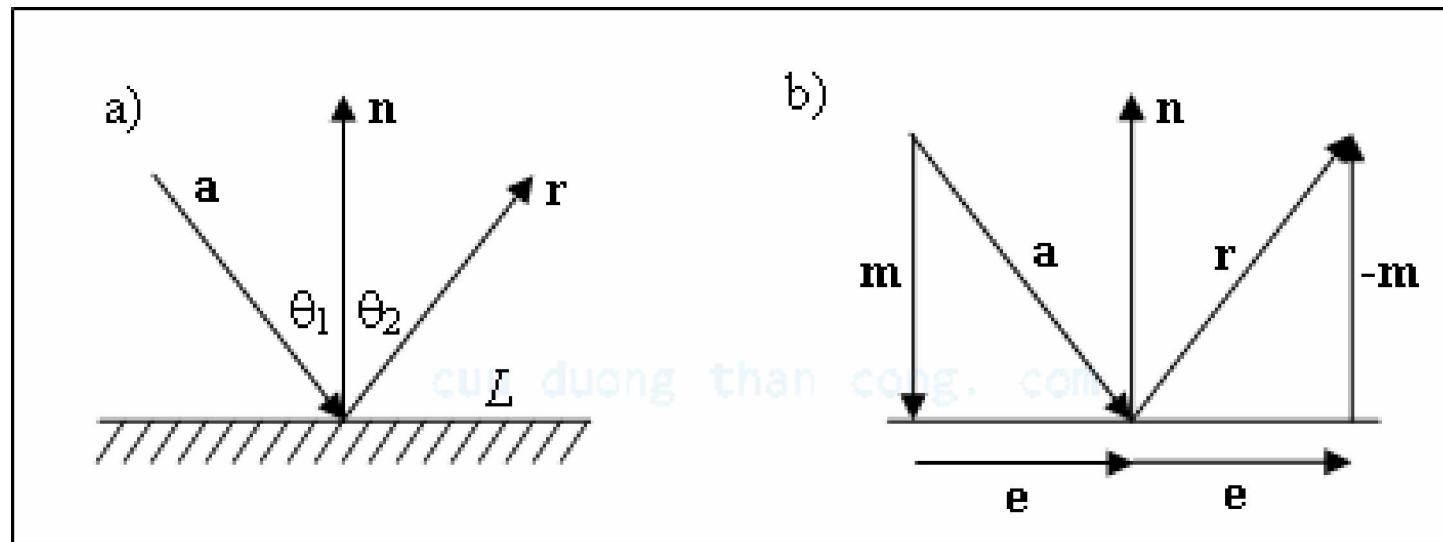
TÍCH VÔ HƯỚNG

□ Tìm tia phản xạ

$$\mathbf{r} = \mathbf{e} - \mathbf{m}, \mathbf{e} = \mathbf{a} - \mathbf{m} \rightarrow \mathbf{r} = \mathbf{a} - 2\mathbf{m}$$

$$\mathbf{m} = \frac{\mathbf{a} \bullet \mathbf{n}}{|\mathbf{n}|^2} \mathbf{n} = (\mathbf{a} \bullet \mathbf{u}_n) \mathbf{u}_n$$

$$\mathbf{r} = \mathbf{a} - 2(\mathbf{a} \bullet \mathbf{u}_n) \mathbf{u}_n$$



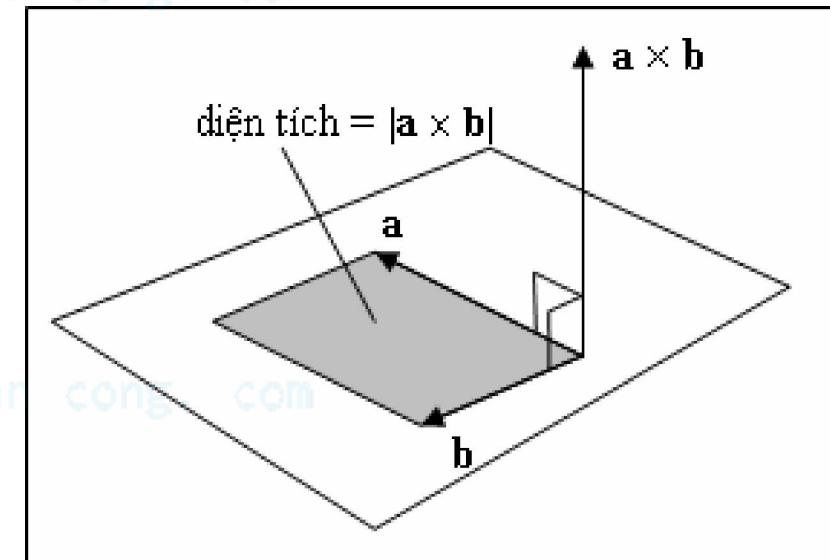
TÍCH CÓ HƯỚNG

- ❑ Tích có hướng của hai vector là một vector
- ❑ Tích có hướng chỉ được định nghĩa cho vector 3 chiều
- ❑ Cho hai vector 3 chiều $\mathbf{a} = (a_x, a_y, a_z)$ và $\mathbf{b} = (b_x, b_y, b_z)$,
thì tích có hướng của chúng như sau

$$\mathbf{a} \times \mathbf{b} = (a_y b_z - a_z b_y) \mathbf{i} + (a_z b_x - a_x b_z) \mathbf{j} + (a_x b_y - a_y b_x) \mathbf{k}$$

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

$$|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}| |\mathbf{b}| \sin(\theta)$$



BIỂU DIỄN ĐỐI TƯỢNG HÌNH HỌC

☐ Hệ tọa độ và khung tọa độ

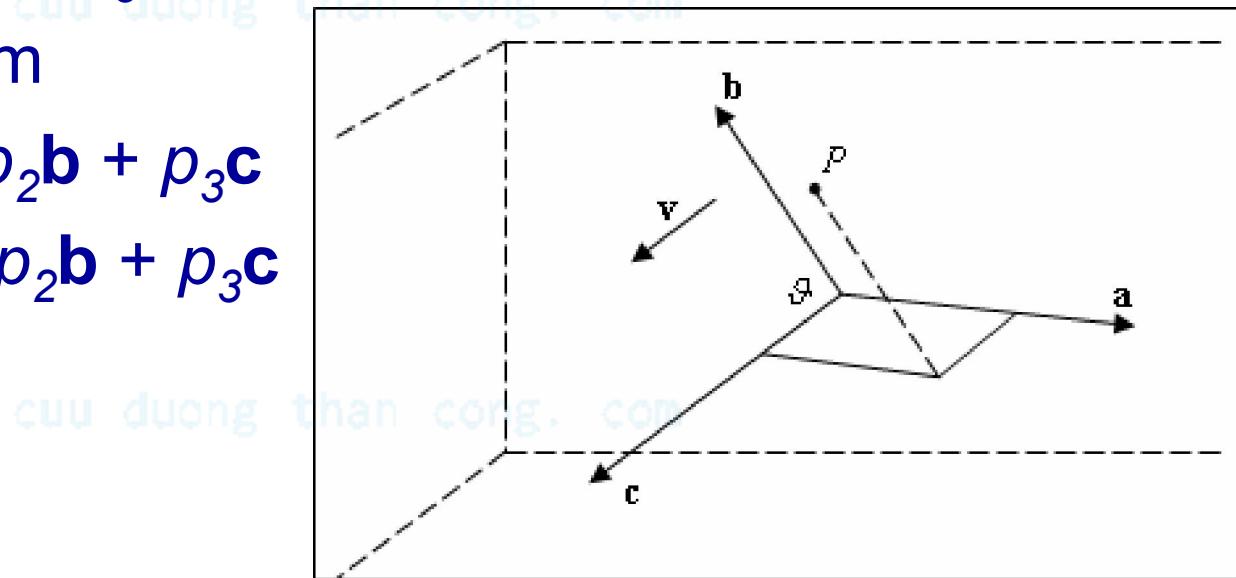
- $(3, 2, 7)$ là điểm hay là vector?
- Khung tọa độ: gốc \varnothing và 3 trục $\mathbf{a}, \mathbf{b}, \mathbf{c}$
- Biểu diễn vector \mathbf{v} bằng cách tìm (v_1, v_2, v_3) sao cho

$$\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c}$$

- Biểu diễn điểm

$$P - \varnothing = p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$$

$$P = \varnothing + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$$



BIỂU DIỄN ĐỒI TƯỢNG HÌNH HỌC

□ Biểu diễn đồng nhất

$$\mathbf{v} = (\mathbf{a}, \mathbf{b}, \mathbf{c}, \vartheta) \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{pmatrix}$$
$$P = (\mathbf{a}, \mathbf{b}, \mathbf{c}, \vartheta) \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ 1 \end{pmatrix}$$

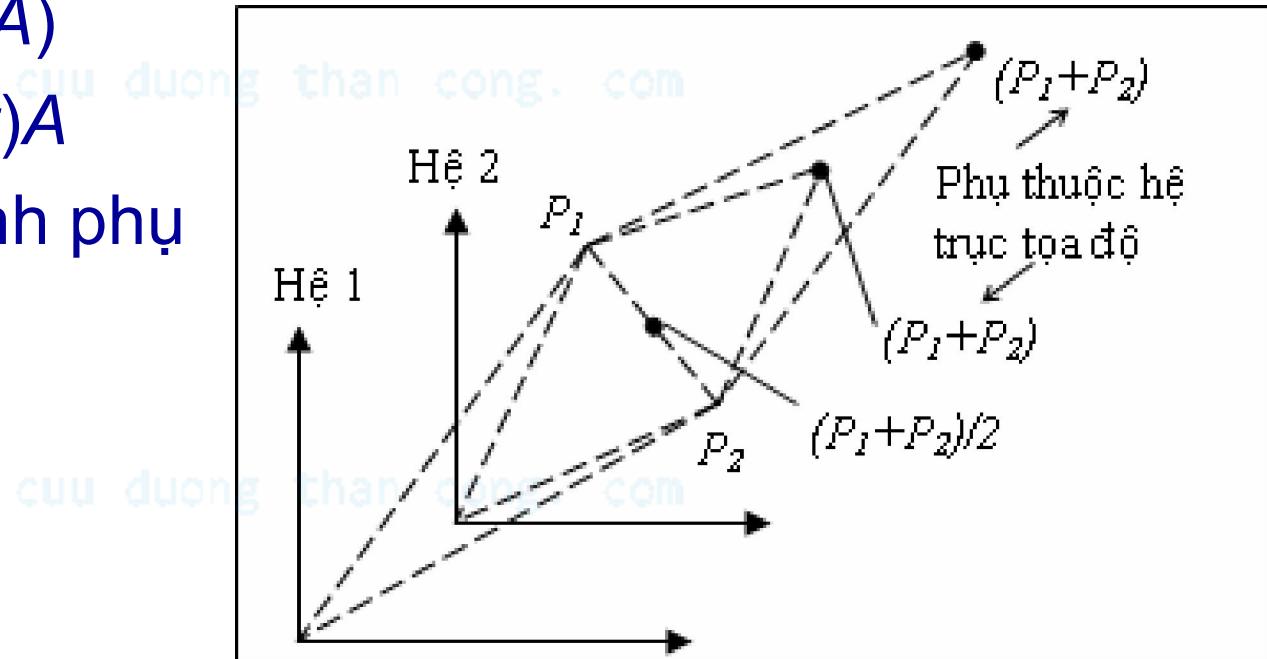
- Hệ tọa độ thông thường → hệ tọa độ đồng nhất
điểm: thêm 1; vector : thêm 0
- Hệ tọa độ đồng nhất → hệ tọa độ thông thường
điểm: xóa 1; vector : xóa 0.

BIỂU DIỄN ĐỐI TƯỢNG HÌNH HỌC

- Điểm – điểm = vector; $(x, y, z, 1) - (u, v, w, 1) = (x - u, y - v, z - w, 0)$.
- Điểm + vector = điểm; $(x, y, z, 1) + (d, e, f, 0) = (x + d, y + e, z + f, 1)$.
- Vector + vector = vector; $(d, e, f, 0) + (m, n, r, 0) = (d + m, e + n, f + r, 0)$ cụu đường thanh công. cùm
- Đại lượng vô hướng x Vector = Vector; $3(d, e, f, 0) = (3d, 3e, 3f, 0)$
- Tổ hợp tuyến tính của vector là vector; $\mathbf{v} = (v_1, v_2, v_3, 0)$ và $\mathbf{w} = (w_1, w_2, w_3, 0)$, a, b là hai đại lượng vô hướng thì $a\mathbf{v} + b\mathbf{w} = (av_1 + bw_1, av_2 + bw_2, av_3 + bw_3, 0)$

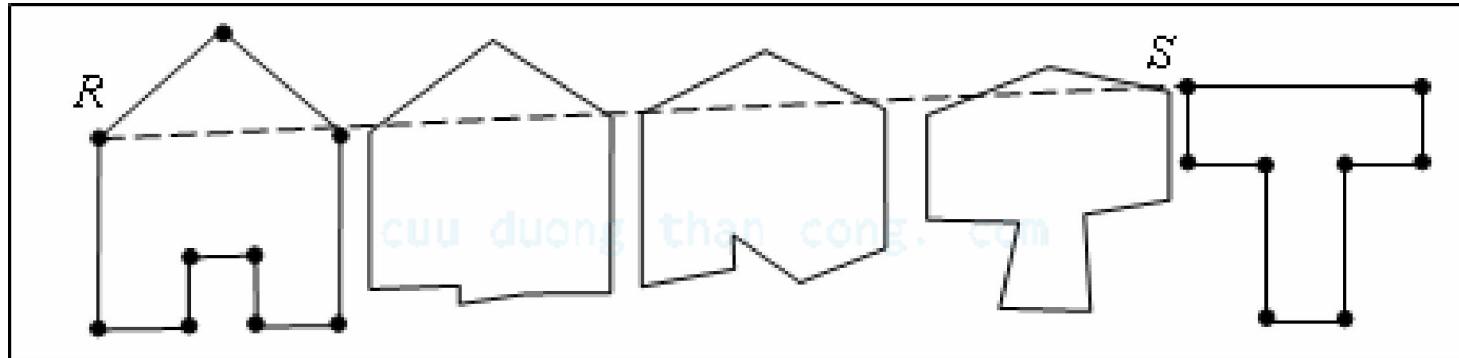
BIỂU DIỄN ĐỐI TƯỢNG HÌNH HỌC

- Tổ hợp affine các điểm: là một điểm. $P = (P_1, P_2, P_3, 1)$ và $R = (R_1, R_2, R_3, 1)$, gọi f và g là hai giá trị vô hướng:
 $fP + gR = (fP_1 + gR_1, fP_2 + gR_2, fP_3 + gR_3, f + g)$.
- Điểm cộng vector là tổ hợp affine các điểm
 - $P = A + t(B - A)$
 - $P = tB + (1 - t)A$
- Tổ hợp tuyến tính phụ thuộc hệ tọa độ



BIỂU DIỄN ĐỐI TƯỢNG HÌNH HỌC

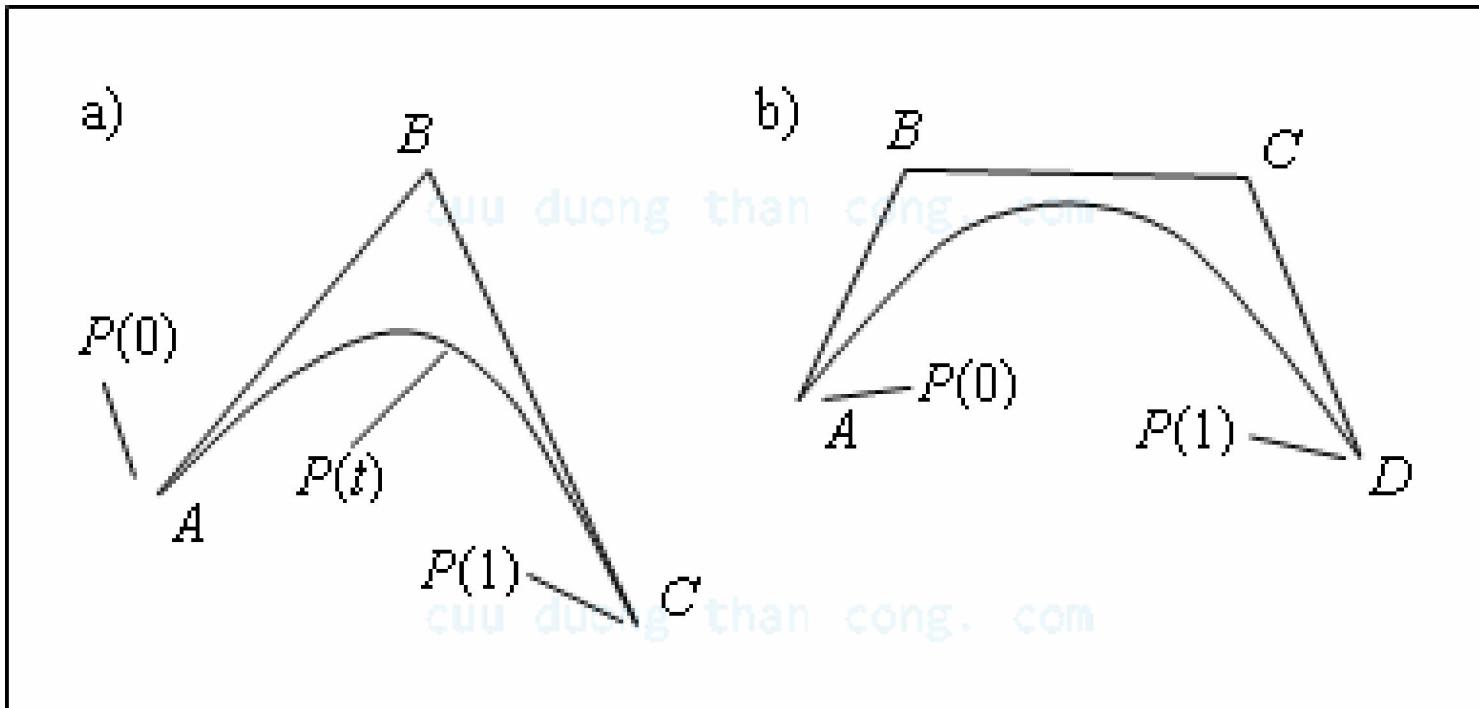
- ❑ Nội suy tuyến tính hai điểm: $P = A(1 - t) + Bt$
- ❑ float lerp(float a, float b, float t)
 { return a + (b - a) * t; }
- ❑ Point2 Canvas::Tween(Point2 A, Point2 B, float t)
- ❑ Sử dụng tweening trong nghệ thuật, hoạt hình
 - $P_i(t) = (1 - t)A_i + tB_i$.



BIỂU DIỄN ĐỐI TƯỢNG HÌNH HỌC

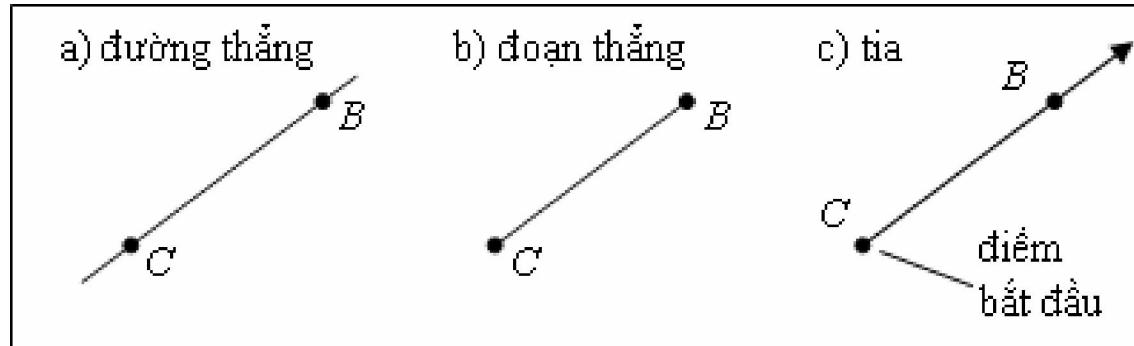
☐ Nội suy bậc 2, bậc 3

$$- P(t) = (1 - t)^2 A + 2(1 - t)tB + t^2 C \quad 1 = ((1 - t) + t)^2$$



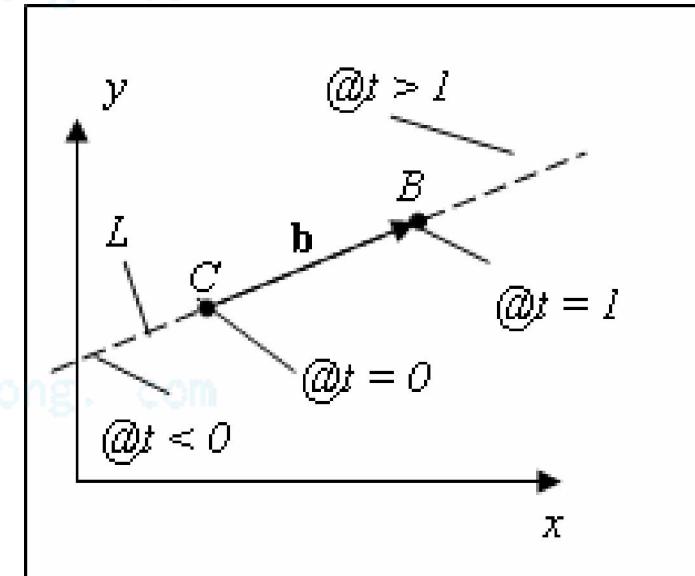
BIỂU DIỄN ĐỐI TƯỢNG HÌNH HỌC

□ Biểu diễn đường thẳng: đoạn thẳng, tia, đường thẳng



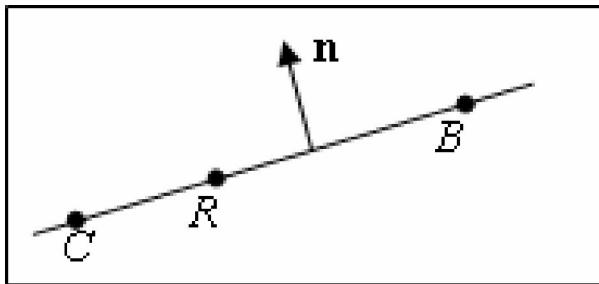
□ Biểu diễn tham số

- $L(t) = C + bt$
- Đoạn thẳng, $0 \leq t \leq 1$
- Tia, $0 \leq t \leq \infty$
- Đường thẳng, $-\infty \leq t \leq \infty$

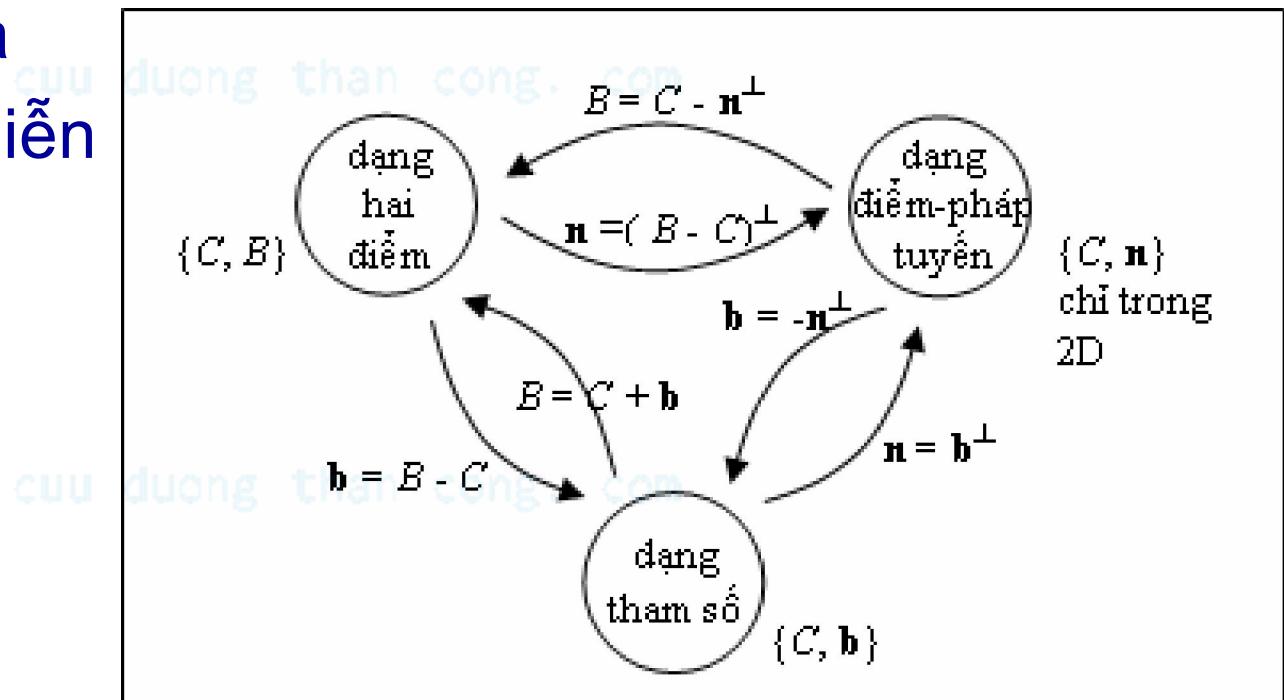


BIỂU DIỄN ĐỔI TƯỢNG HÌNH HỌC

- Dạng biểu diễn điểm pháp tuyến: $\mathbf{n} \bullet (\mathbf{R} - \mathbf{C}) = 0$



- Chuyển đổi giữa những cách biểu diễn khác nhau

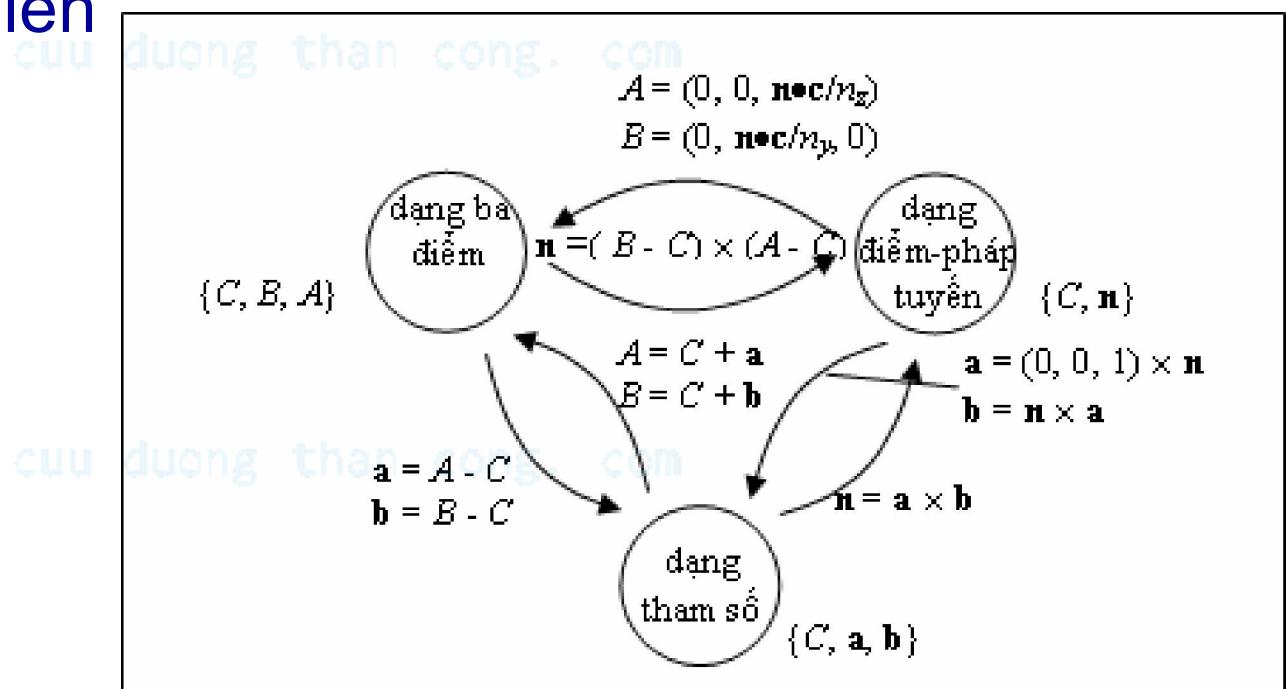
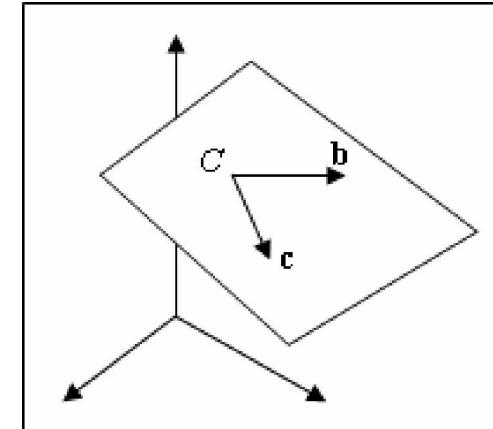


BIỂU DIỄN ĐỔI TƯỢNG HÌNH HỌC

- Biểu diễn mặt phẳng dưới dạng tham số:

$$P(s, t) = C + sa + tb$$

- Chuyển đổi giữa những cách biểu diễn khác nhau

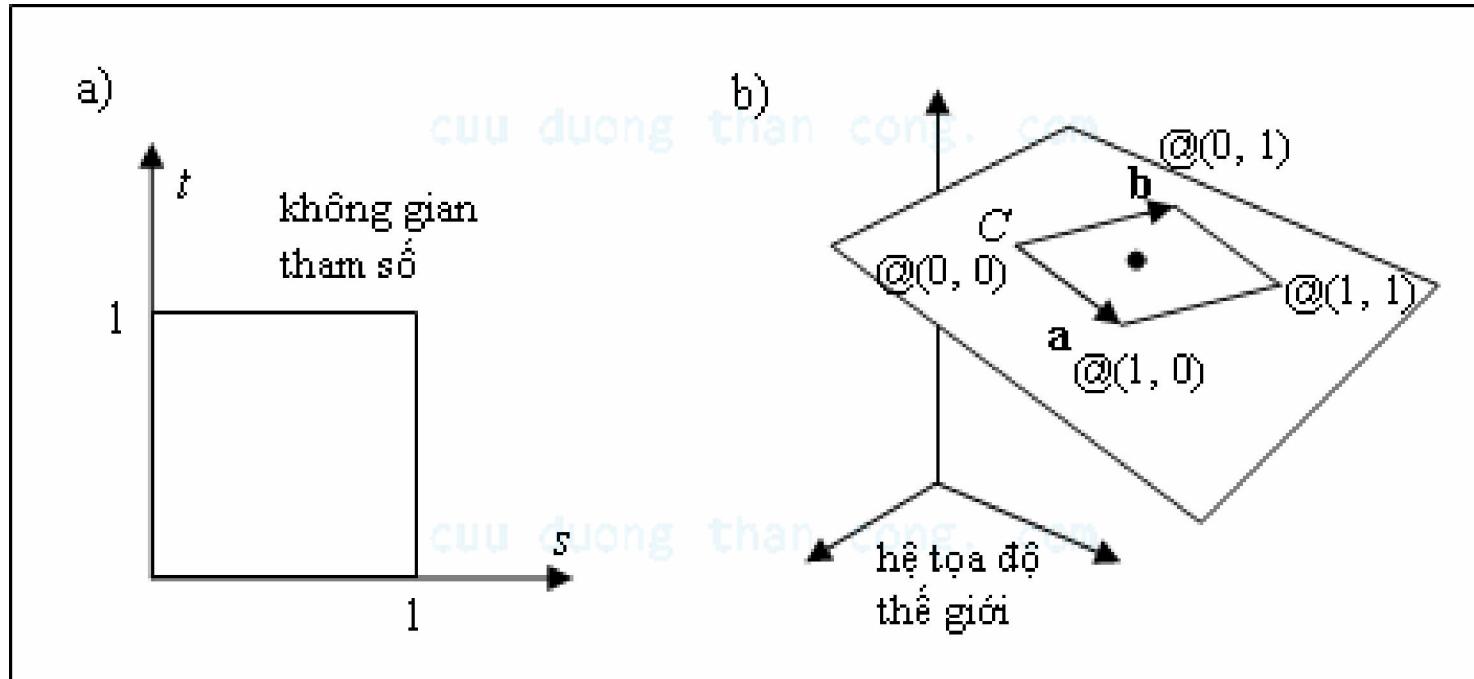


BIỂU DIỄN ĐỐI TƯỢNG HÌNH HỌC

☐ Mảnh phẳng: $P(s, t) = C + \mathbf{a}s + \mathbf{b}t$

$$P(0, 0) = C; \quad P(1, 0) = C + \mathbf{a}$$

$$P(0, 1) = C + \mathbf{b} \quad P(1, 1) = C + \mathbf{a} + \mathbf{b}$$



GIAO ĐIỂM CỦA 2 ĐOẠN THẲNG

$$AB(t) = A + \mathbf{b}t; CD(u) = C + \mathbf{d}u$$

Giao điểm: tìm t và u sao cho $A + \mathbf{b}t = C + \mathbf{d}u$

$$\mathbf{b}t = \mathbf{c} + \mathbf{d}u \text{ với } \mathbf{c} = C - A$$

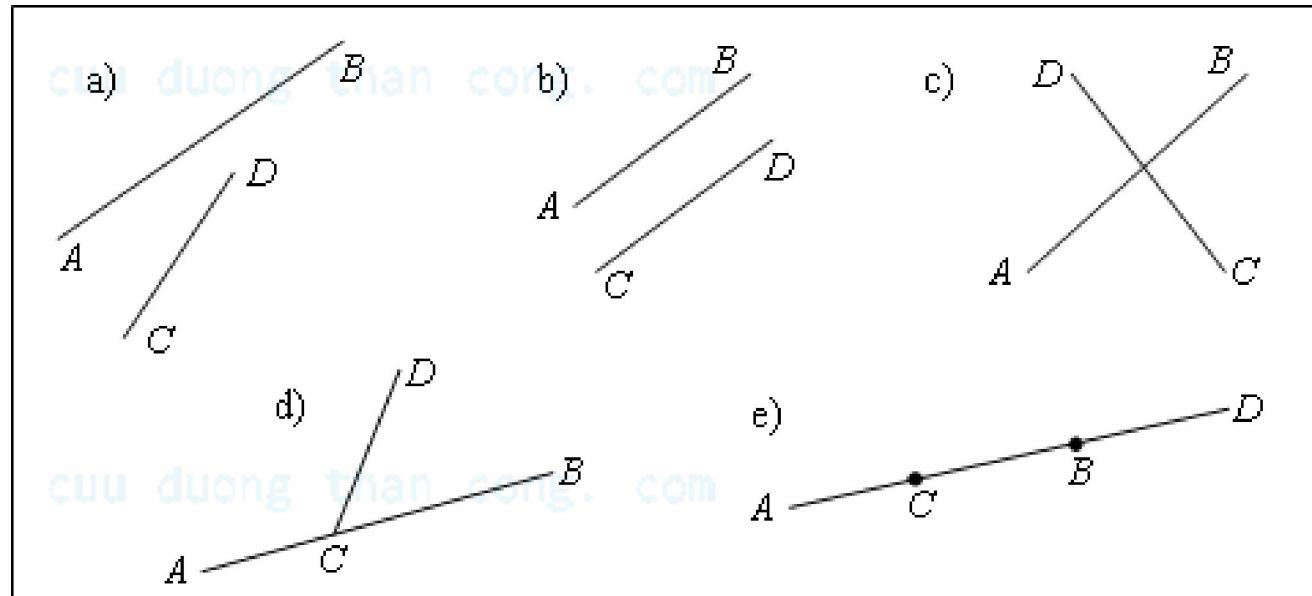
$$\mathbf{d}^\perp \bullet \mathbf{b}t = \mathbf{d}^\perp \bullet \mathbf{c}$$

✓ $\mathbf{d}^\perp \bullet \mathbf{b} \neq 0.$

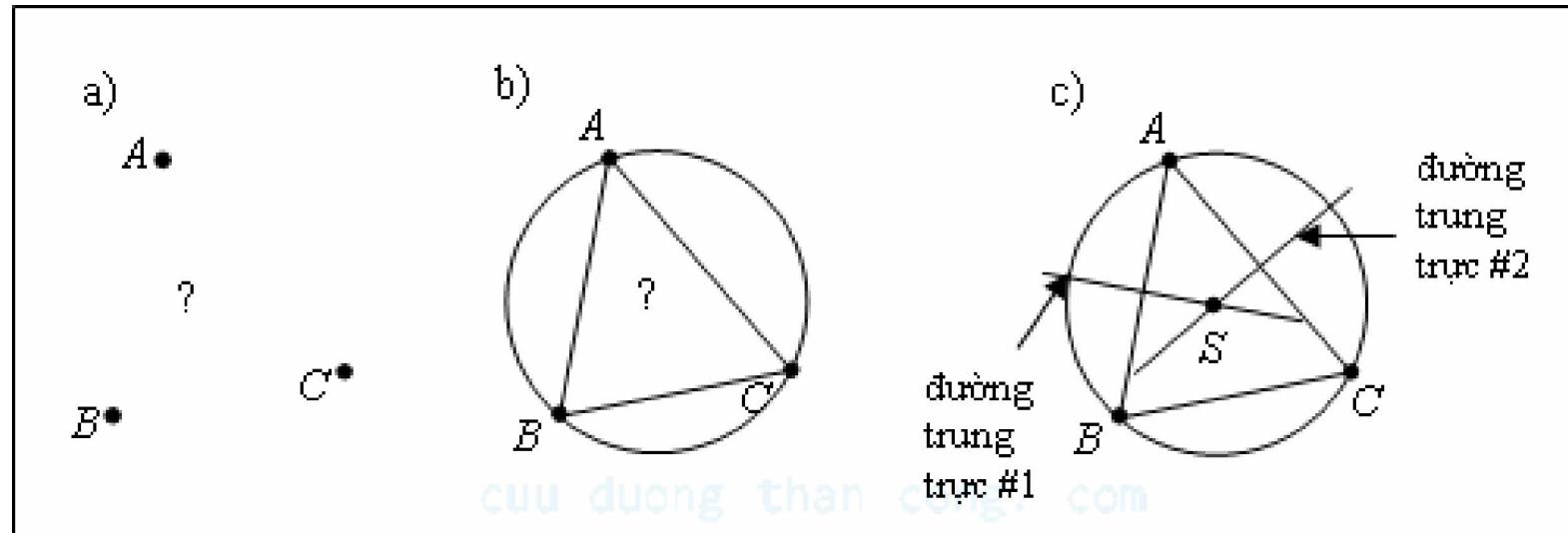
$$t = \frac{\mathbf{d}^\perp \bullet \mathbf{c}}{\mathbf{d}^\perp \bullet \mathbf{b}}$$

$$u = \frac{\mathbf{b}^\perp \bullet \mathbf{c}}{\mathbf{d}^\perp \bullet \mathbf{b}}$$

✓ $\mathbf{d}^\perp \bullet \mathbf{b} = 0$



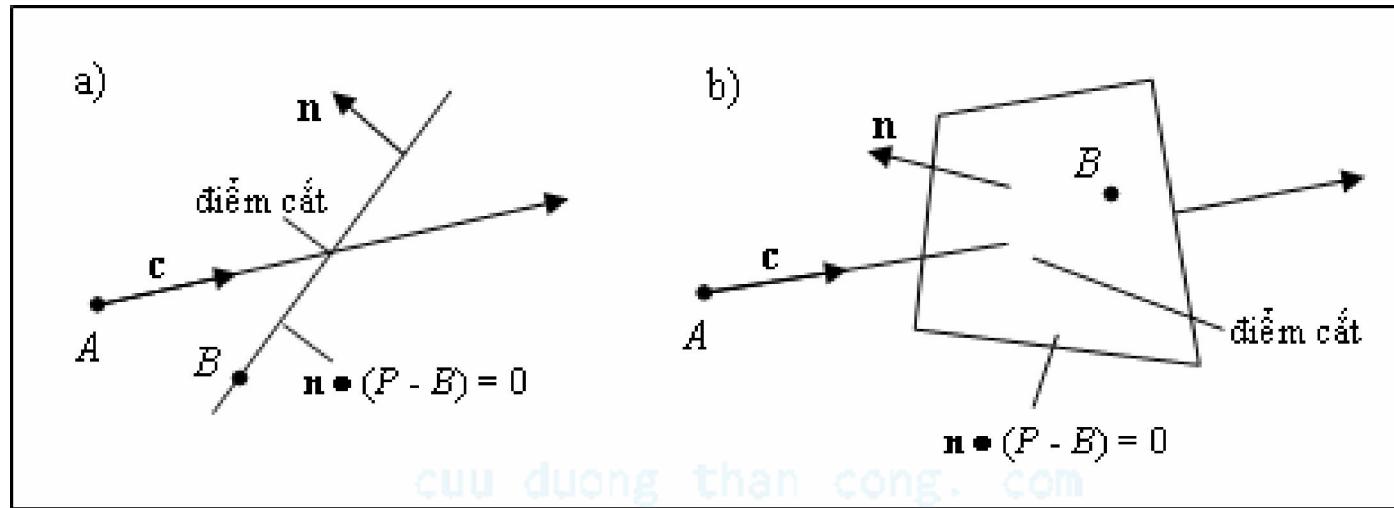
ĐƯỜNG TRÒN ĐI QUA 3 ĐIỂM



- Đường trung trực $L(t) = \frac{1}{2}(A + B) + (B - A)^{\perp}t$
- Đặt $\mathbf{a} = B - A$; $\mathbf{b} = C - B$; $\mathbf{c} = A - C$;
- Trung trực của AB; $A + \mathbf{a}/2 + \mathbf{a}^{\perp}t$; của CD; $A - \mathbf{c}/2 + \mathbf{c}^{\perp}u$
- $\mathbf{a}^{\perp}t = \mathbf{b}/2 + \mathbf{c}^{\perp}u \rightarrow$ *cuu duong than cong. com*

$$t = \frac{1}{2} \frac{\mathbf{b} \bullet \mathbf{c}}{\mathbf{a}^{\perp} \bullet \mathbf{c}}$$
$$S = A + \frac{1}{2} \left(\mathbf{a} + \frac{\mathbf{b} \bullet \mathbf{c}}{\mathbf{a}^{\perp} \bullet \mathbf{c}} \mathbf{a}^{\perp} \right)$$

GIAO CỦA ĐƯỜNG THẲNG VÀ MẶT PHẲNG



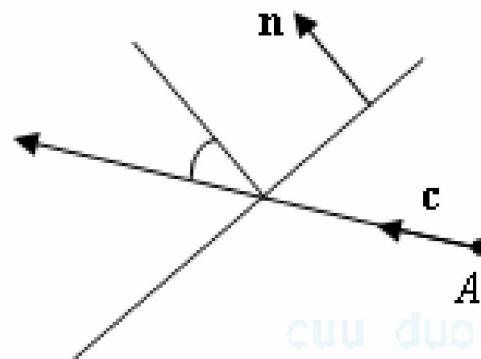
□ $\mathbf{n} \bullet (\mathbf{A} + \mathbf{c}t_{hit} - \mathbf{B}) = 0 \rightarrow \mathbf{n} \bullet (\mathbf{A} - \mathbf{B}) + \mathbf{n} \bullet \mathbf{c}t_{hit} = 0$

$$t_{hit} = \frac{\mathbf{n} \bullet (\mathbf{B} - \mathbf{A})}{\mathbf{n} \bullet \mathbf{c}}$$

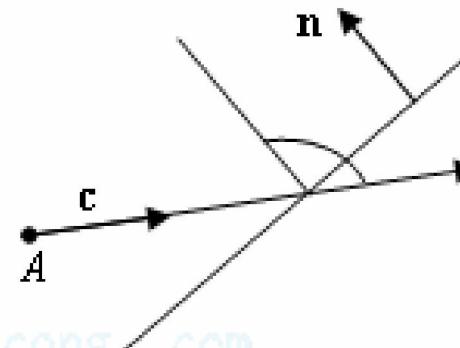
□ $\text{điểm cắt } P_{hit} = \mathbf{A} + \mathbf{c}t_{hit}$

GIAO CỦA ĐƯỜNG THẲNG VÀ MẶT PHẲNG

a) tia "nằm dọc" theo n

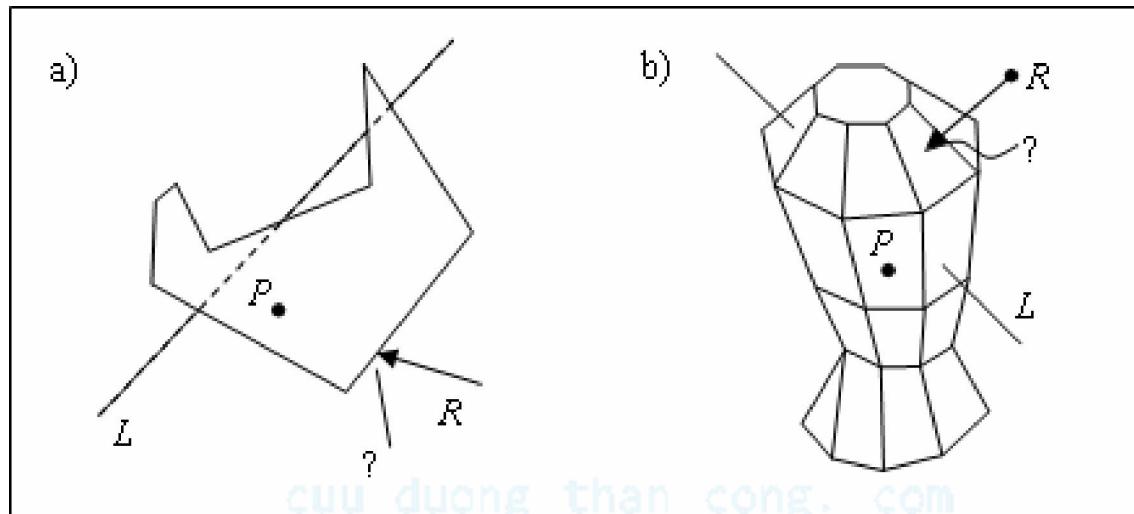


b) tia "nằm ngược" với n



- $n \cdot c > 0$, tia đi "dọc theo" hướng pháp tuyến của đường thẳng
- $n \cdot c = 0$, tia song song với đường thẳng
- $n \cdot c < 0$, tia đi "ngược với" hướng pháp tuyến của đường thẳng

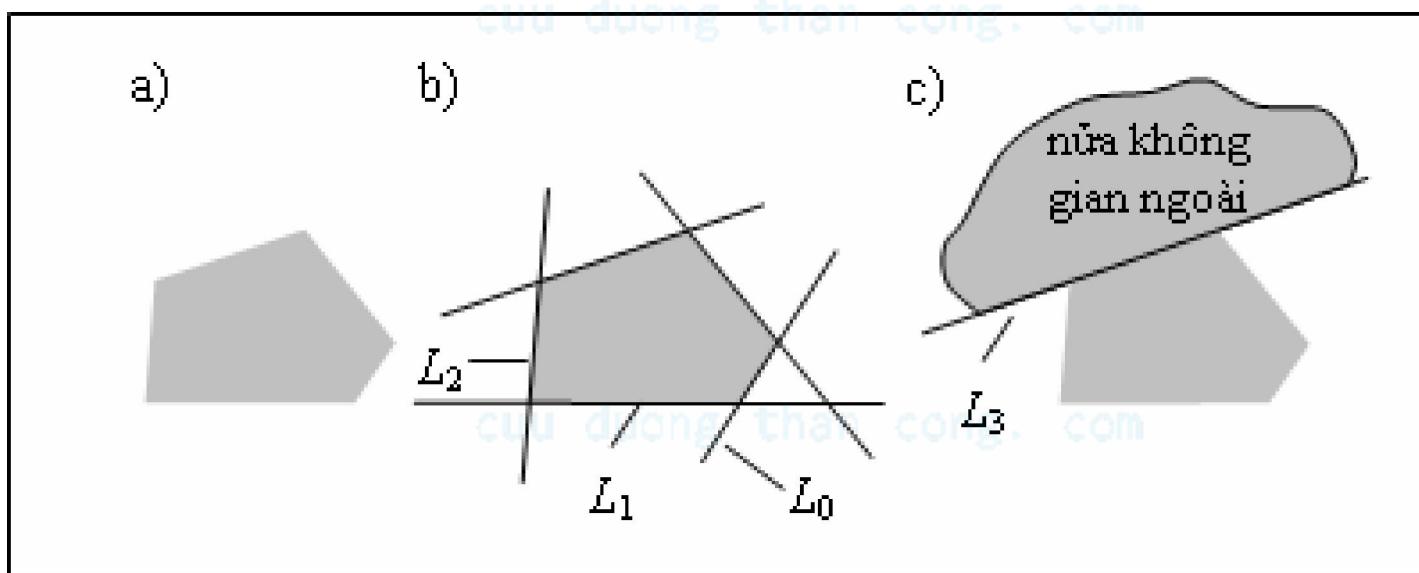
BÀI TOÁN LIÊN QUAN ĐẾN ĐA GIÁC



- Định nghĩa đa giác và đa diện
- Các bài toán liên quan
 - Điểm P cho trước nằm trong hay nằm ngoài đa giác (hoặc khối đa diện)
 - Giao điểm đầu tiên tia R với đa giác (hoặc khối đa diện)
 - Phần nào của đường thẳng L sẽ nằm trong đa giác (hoặc khối đa diện), phần nào nằm ngoài.

BÀI TOÁN LIÊN QUAN ĐẾN ĐA GIÁC

- Bài toán tìm giao điểm
- Đa giác (đa diện) lồi được mô tả bởi các đường thẳng (mặt phẳng) bao → tìm giao điểm với đường thẳng chằng qua tìm giao điểm của đường thẳng với tập các đường thẳng (mặt phẳng) bao

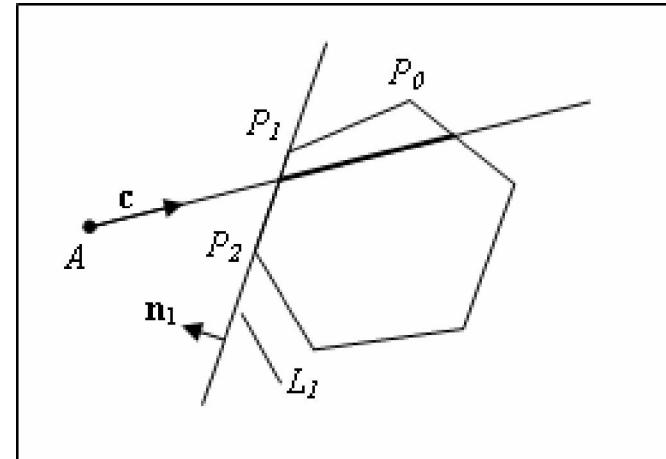


BÀI TOÁN LIÊN QUAN ĐẾN ĐA GIÁC

☐ Tìm giao điểm của tia với đa giác

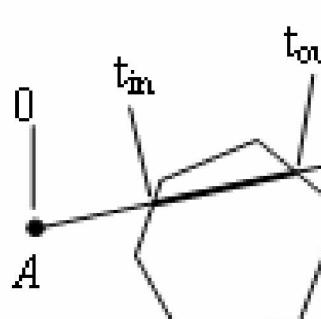
- Điểm cắt vào = $A + ct_{in}$;
- Điểm cắt ra = $A + ct_{out}$;
- Nằm trong đa giác $[t_{in}, t_{out}]$.

☐ Cắt xén tia

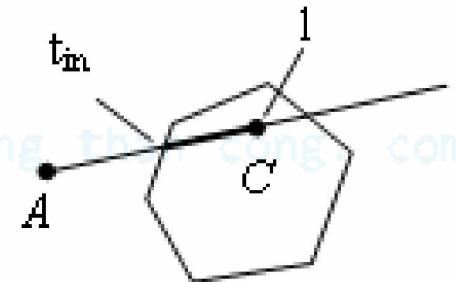


$$A' = A + \mathbf{c} \max(0, t_{in}) \quad C' = A + \mathbf{c} \min(1, t_{out})$$

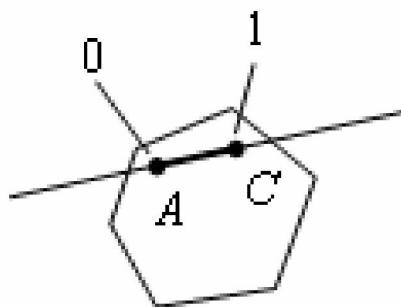
a)



b)



c)



BÀI TOÁN LIÊN QUAN ĐẾN ĐA GIÁC

❑ Xác định tia đi vào hay đi ra khỏi đa giác

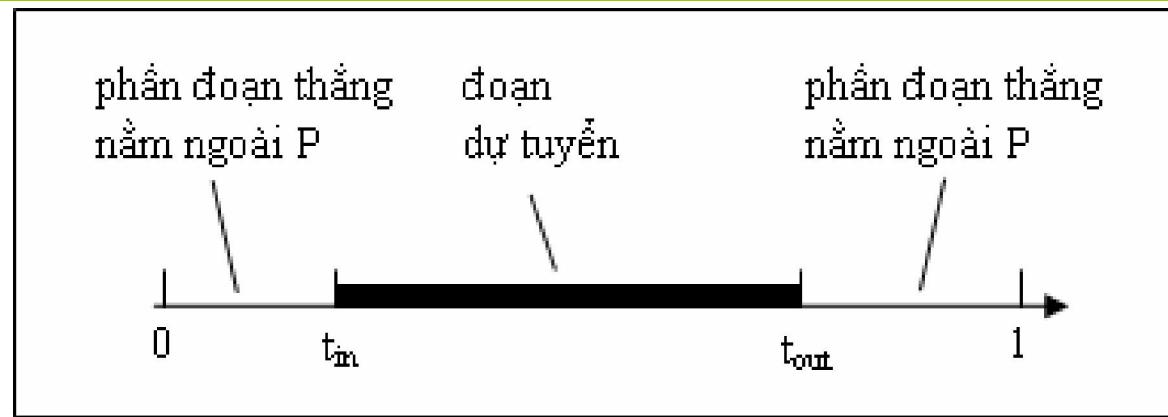
- Nếu $n \bullet c > 0$, tia đi ra khỏi P
- Nếu $n \bullet c = 0$, tia song song với đường thẳng
- Nếu $n \bullet c < 0$, tia đi vào P

❑ Với mỗi đường bao, chúng ta sẽ phải xác định:

- Thời gian cắt của tia với đường bao.
- Tia đi vào hay đi ra khỏi đa giác ở đường bao.

cuu duong than cong. com

BÀI TOÁN LIÊN QUAN ĐẾN ĐA GIÁC



Gọi đoạn $[t_{in}, t_{out}]$ là **đoạn dự tuyển**

- Khởi gán giá trị ban đầu $[0, 1]$ cho đoạn dự tuyển
- Với mỗi đường bao, xác định t_{hit} và xác định tia đi vào hay đi ra khỏi đa giác:
 - Nếu tia đi vào đa giác, thì $t_{in} = \max(t_{in}, t_{hit})$.
 - Nếu tia đi ra khỏi đa giác, thì $t_{out} = \min(t_{out}, t_{hit})$.
- Nếu $t_{in} > t_{out}$ thì tia không cắt đa giác, và chương trình kết thúc.
- Nếu đoạn dự tuyển không trống, thì đoạn thẳng từ $A + ct_{in}$ đến $A + ct_{out}$ nằm trong P .

Trường Đại Học Bách Khoa TP Hồ Chí Minh
Khoa Khoa học & Kỹ thuật Máy tính



ĐỒ HỌA MÁY TÍNH

CHƯƠNG 5:

BIẾN ĐỔI HÌNH

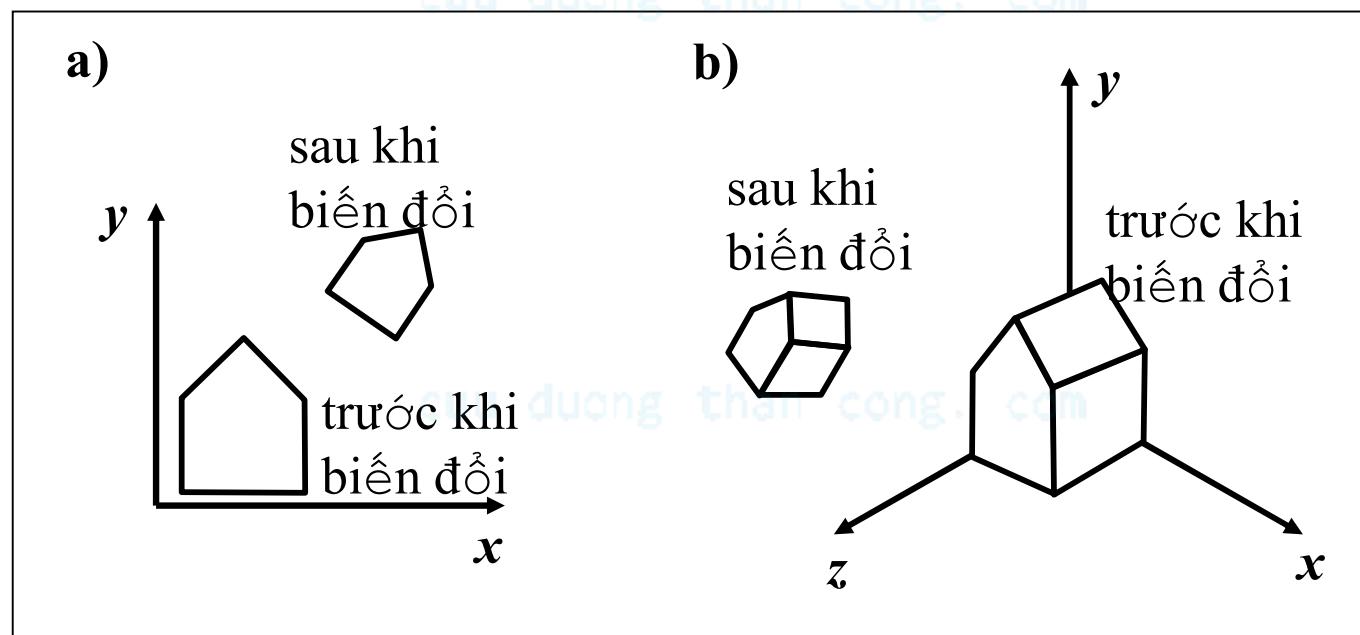
NỘI DUNG TRÌNH BÀY

- ❑ Giới thiệu những khái niệm cơ bản của phép biến đổi affine.
- ❑ Phép biến đổi hình 2D
- ❑ Phép biến đổi hình 3D
- ❑ Biến đổi hệ trực tọa độ
- ❑ Sử dụng phép biến đổi affine trong chương trình
- ❑ Vẽ khung cảnh 3D với OpenGL

cuuduongthancong.com

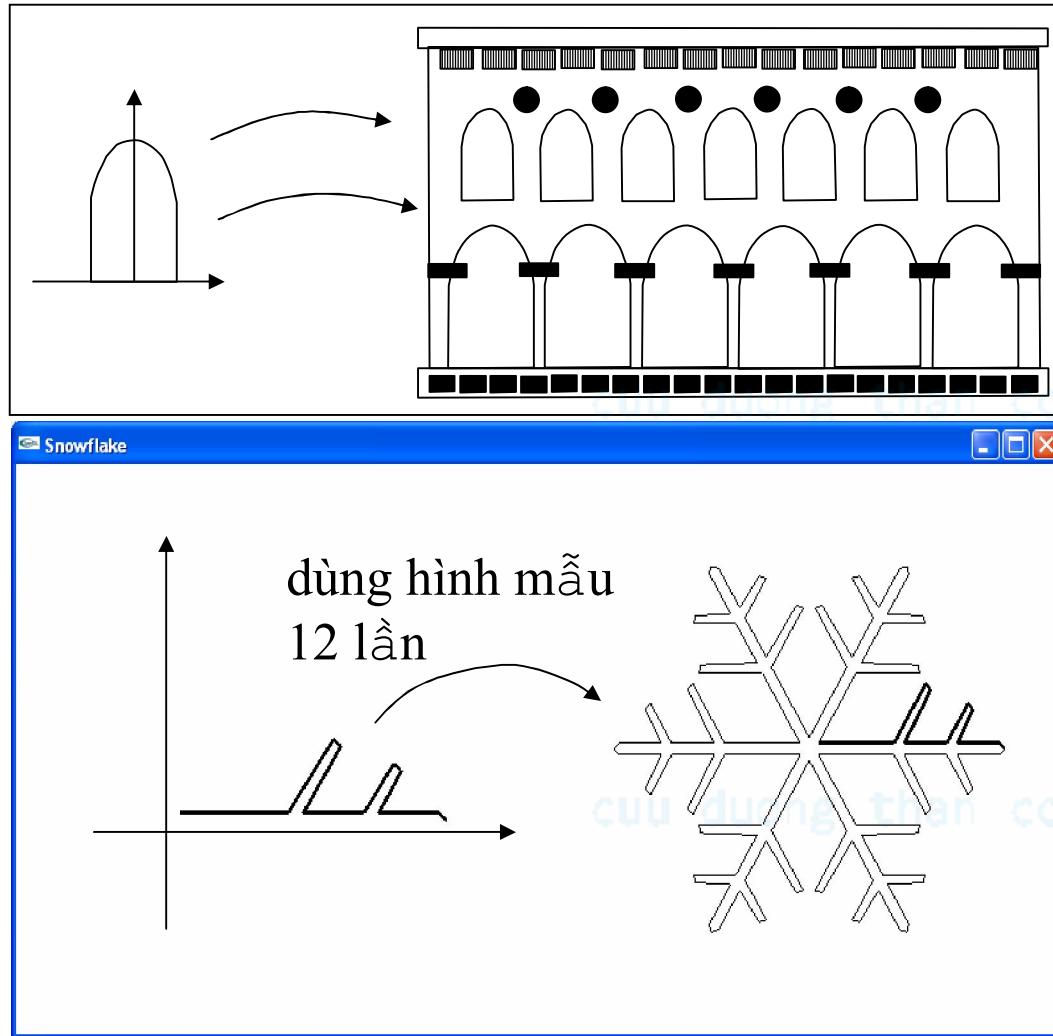
CÁC KHÁI NIỆM CƠ BẢN

- **Phép biến đổi affine** là khái niệm cơ bản nhất của đồ họa máy tính, là trọng tâm của OpenGL.
- Dùng khung tọa độ và hệ tọa độ đồng nhất.
- Phép biến đổi affine tổng thể là hợp của 3 phép biến đổi affine cơ bản: phép biến đổi tỷ lệ, phép quay và phép tịnh tiến.



CÁC KHÁI NIỆM CƠ BẢN

❑ Ứng dụng của phép biến đổi

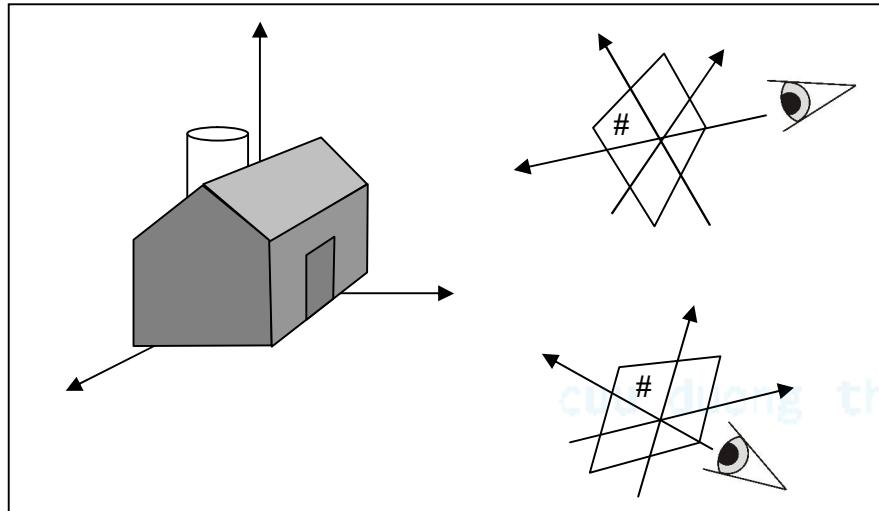


✓ Tạo khung cảnh 3D từ những đối tượng đơn giản

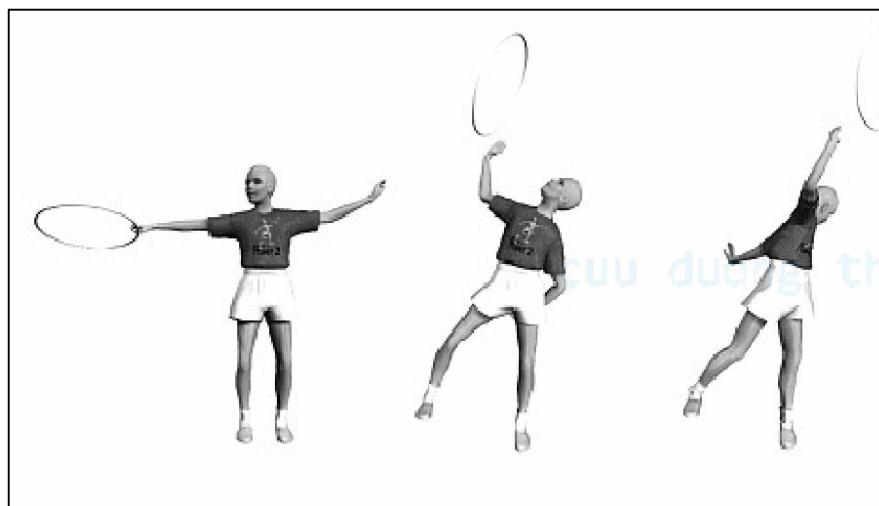
✓ Tạo đối tượng phức tạp từ đối tượng đơn giản

CÁC KHÁI NIỆM CƠ BẢN

❑ Ứng dụng của phép biến đổi



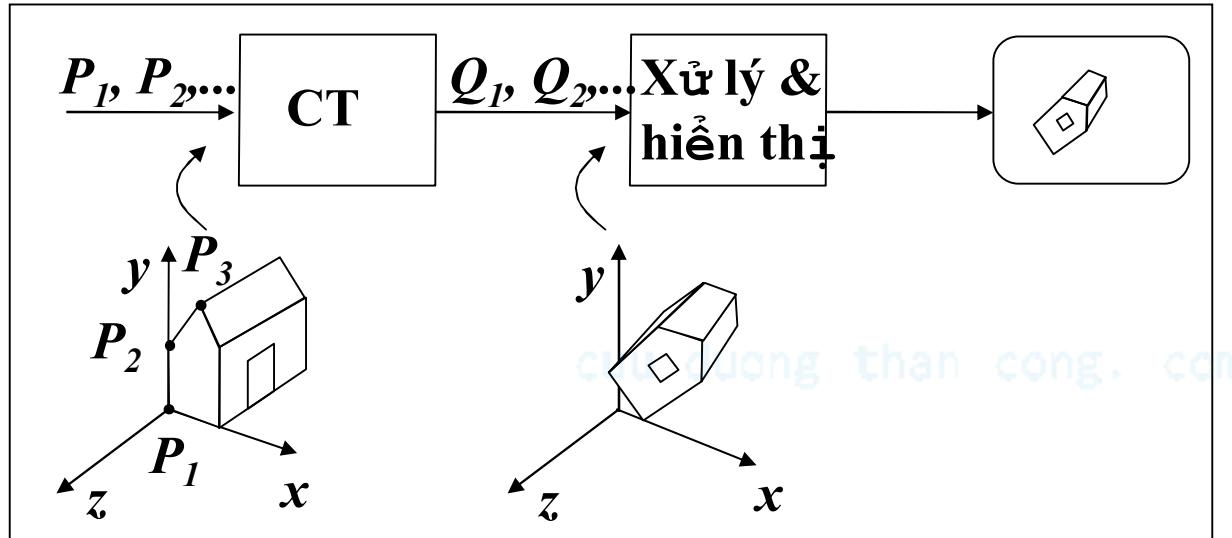
✓ Nhìn khung cảnh ở những góc nhìn khác nhau bằng cách thay đổi hướng, vị trí của camera



✓ Tạo hoạt hình bằng cách di chuyển và quay hệ tọa độ riêng của từng đối tượng

CÁC KHÁI NIỆM CƠ BẢN

□ Sử dụng phép biến đổi trong OpenGL



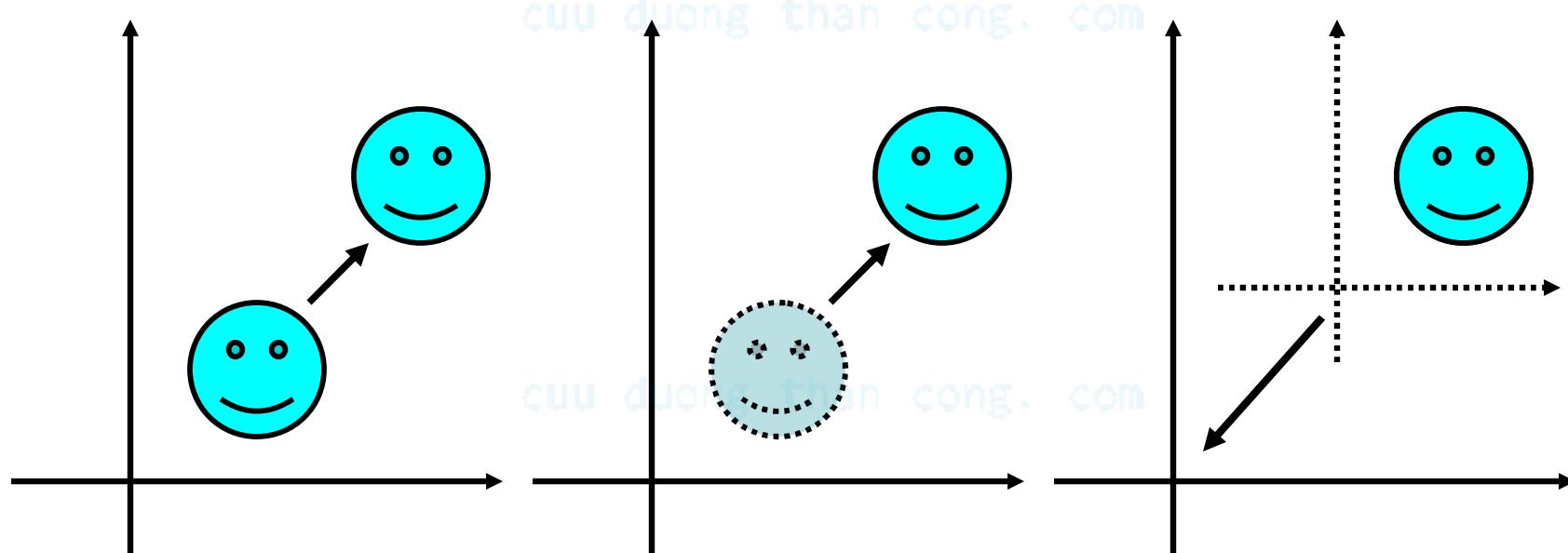
Đường ống đồ họa: là một loạt những thao tác được áp dụng cho các điểm gửi xuống đường ống.

```
glBegin(GL_LINES) ;  
    glVertex3f( . . . ) ; //gửi P1 đến đường ống  
    glVertex3f( . . . ) ; //gửi P2 đến đường ống  
    glVertex3f( . . . ) ; //gửi P3 đến đường ống  
glEnd() ;
```

CÁC KHÁI NIỆM CƠ BẢN

☐ Biến đổi đối tượng và biến đổi hệ trục tọa độ

- Biến đổi đối tượng: hệ trục tọa độ giữ nguyên, biến đổi đối tượng.
- Biến đổi hệ trục tọa độ: biến đổi hệ trục tọa độ, biểu diễn đối tượng trong hệ trục mới.



CÁC KHÁI NIỆM CƠ BẢN

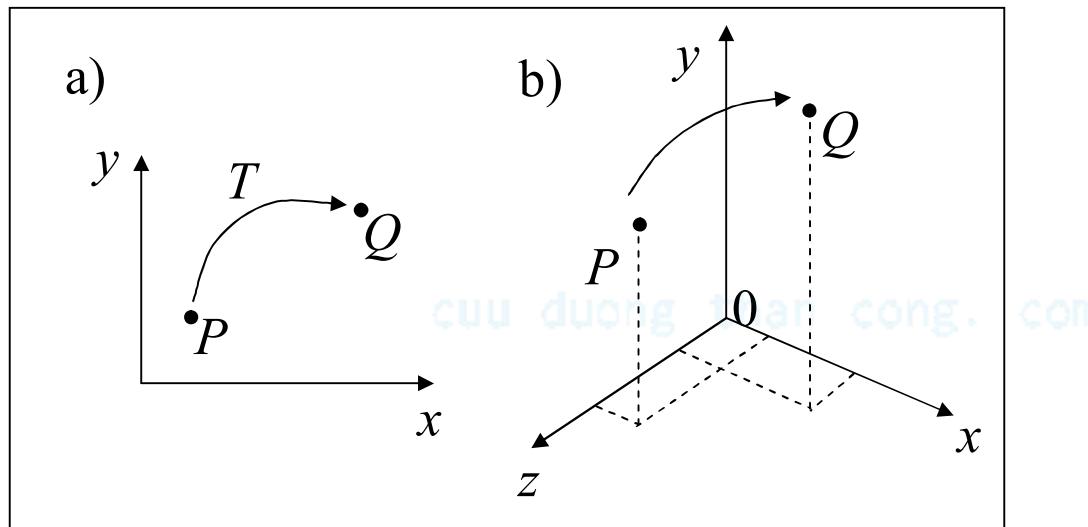
□ Định nghĩa phép biến đổi tổng quát

- Phép biến đổi làm thay đổi điểm P trong không gian 2D (hoặc 3D) thành một điểm Q bằng một công thức hay thuật toán T nào đó.

$$P = (Px, Py, 1); Q = (Qx, Qy, 1) \quad (Q - ảnh)$$

$$(Qx, Qy, 1) = T(Px, Py, 1) \quad (T - phép biến đổi)$$

$$Q = T(P).$$



CÁC KHÁI NIỆM CƠ BẢN

□ Phép biến đổi affine

- Là phép biến đổi thông dụng trong đồ họa máy tính
- Có dạng đơn giản: tọa độ của Q là tổ hợp tuyến tính các tọa độ của P.

$$Q_x = m_{11}P_x + m_{12}P_y + m_{13}$$

$$Q_y = m_{21}P_x + m_{22}P_y + m_{23}$$

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

T

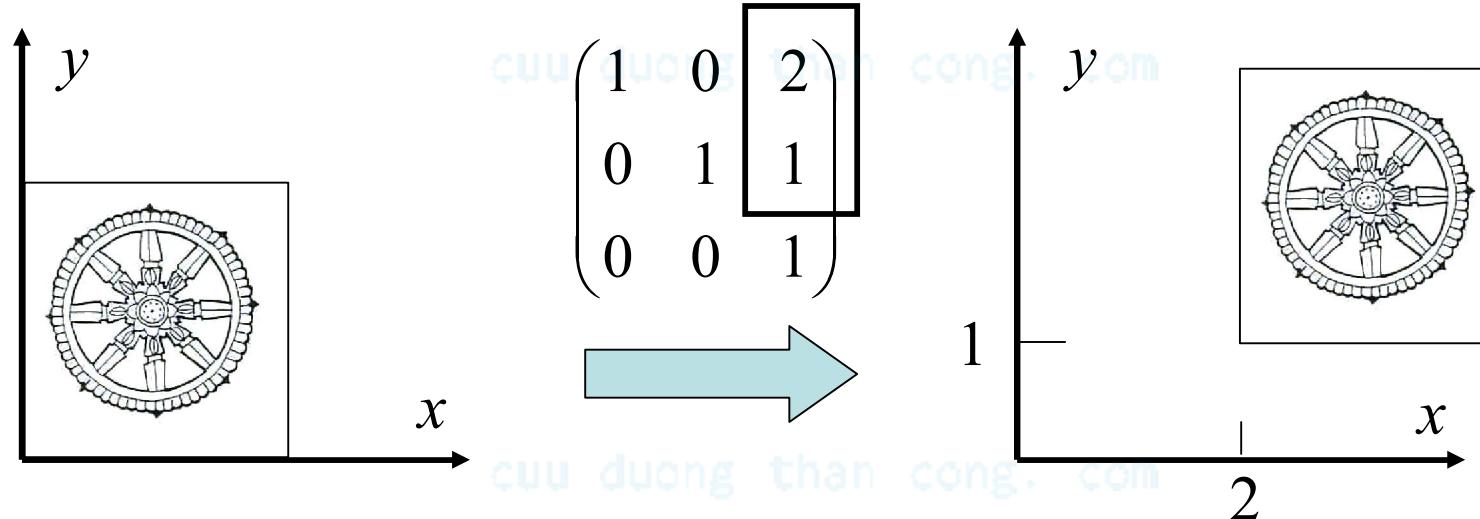
hàng thứ 3 luôn
là (0, 0, 1)

PHÉP BIẾN ĐỔI HÌNH 2D

□ Phép tịnh tiến

$$\begin{aligned} Q_x &= P_x + m_{13} \\ Q_y &= P_y + m_{23} \end{aligned}$$

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & m_{13} \\ 0 & 1 & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

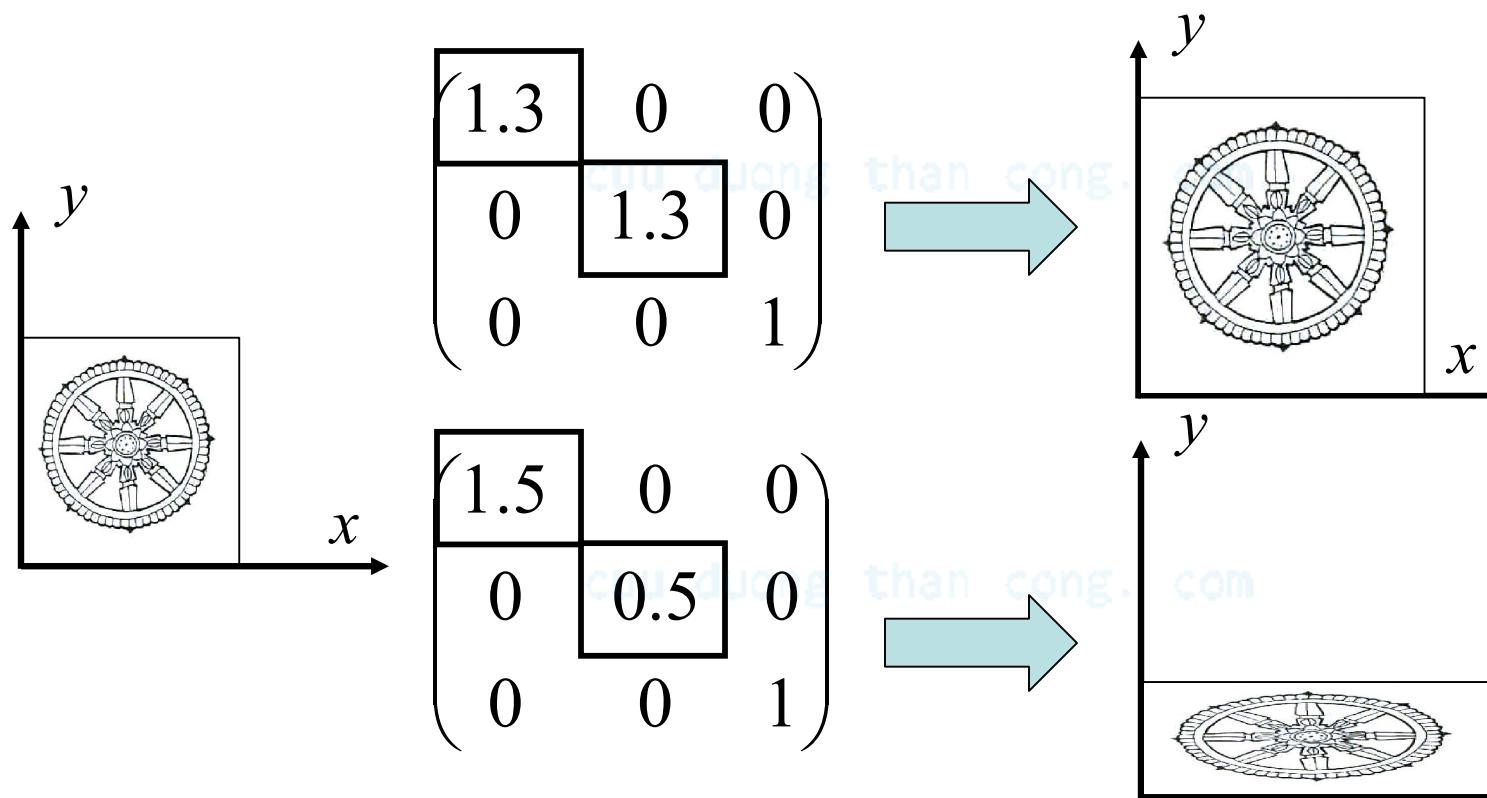


PHÉP BIẾN ĐỔI HÌNH 2D

□ Phép biến đổi tỷ lệ

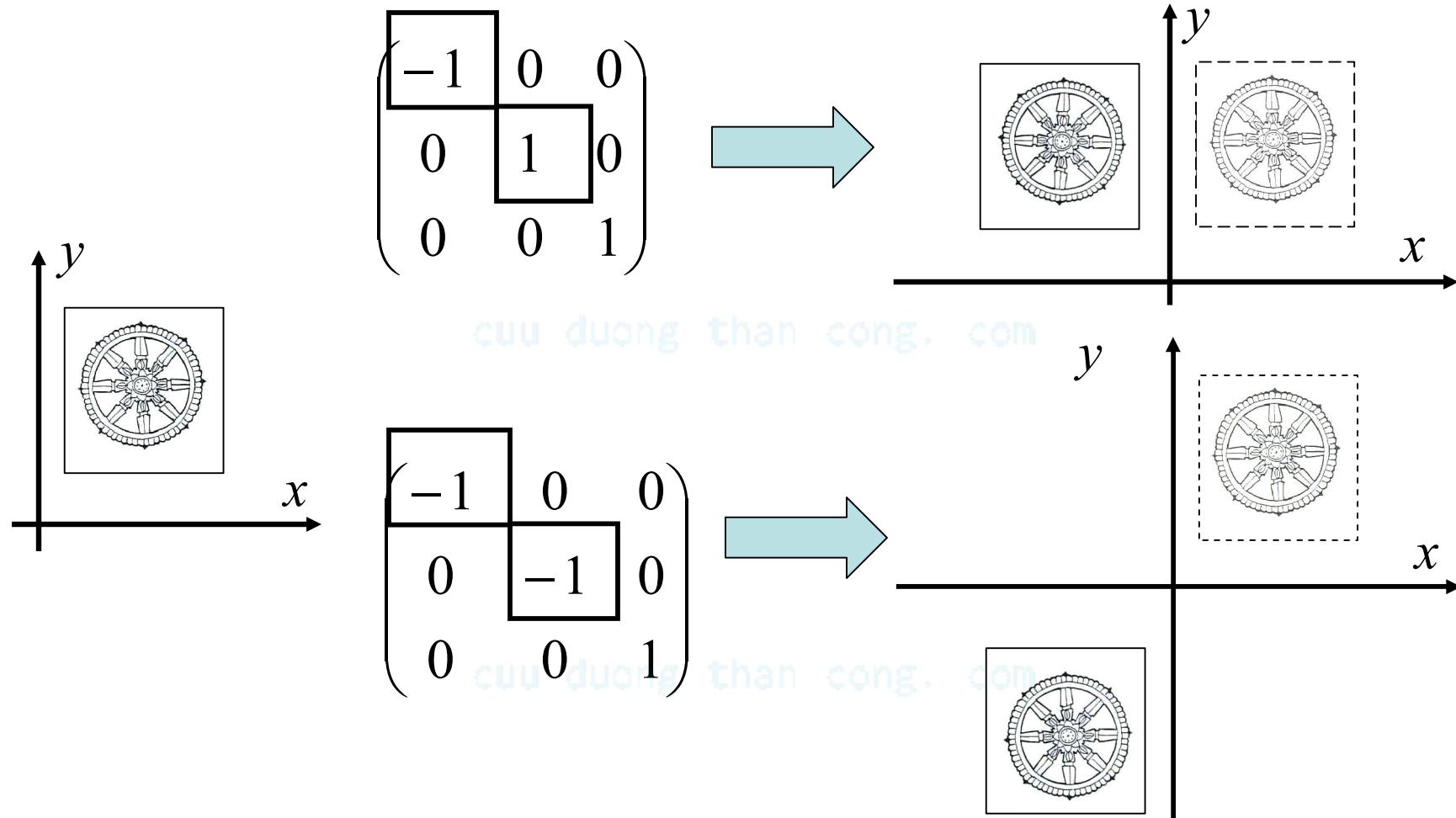
$$Q_x = S_x P_x$$
$$Q_y = S_y P_y$$

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$



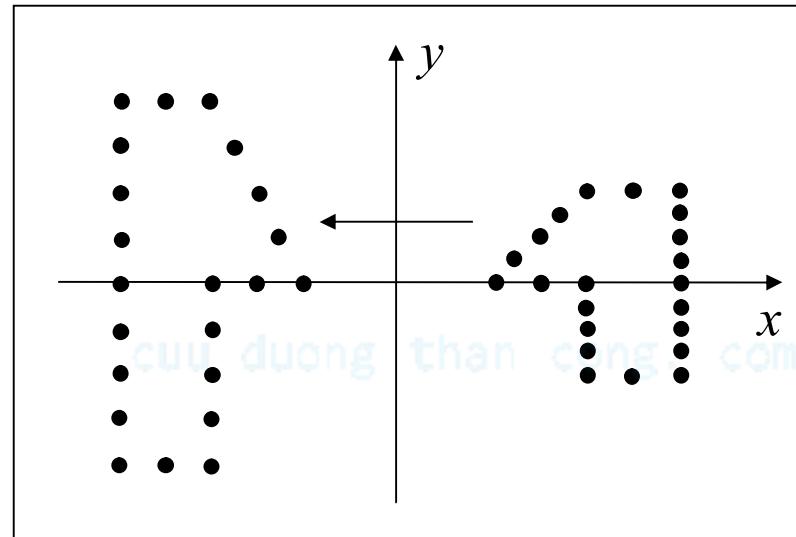
PHÉP BIẾN ĐỔI HÌNH 2D

□ Phép biến đổi tỷ lệ (phép đối xứng)



PHÉP BIẾN ĐỔI HÌNH 2D

□ Phép biến đổi tỷ lệ (phép đối xứng)



$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

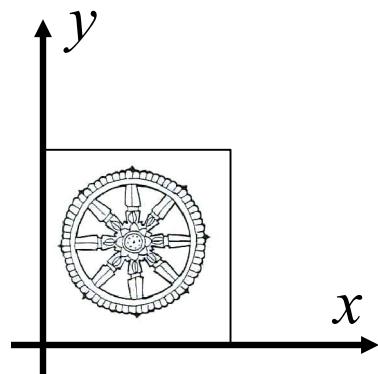
PHÉP BIẾN ĐỔI HÌNH 2D

□ Phép quay

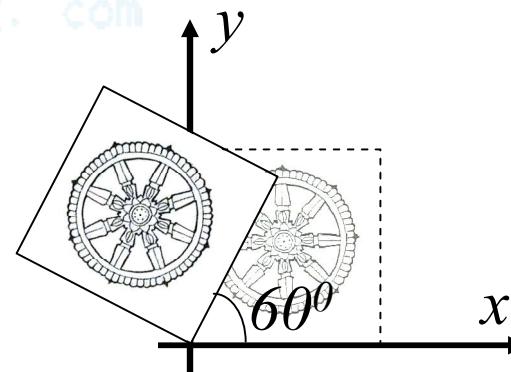
$$Q_x = P_x \cos(\theta) - P_y \sin(\theta)$$

$$Q_y = P_x \sin(\theta) + P_y \cos(\theta)$$

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

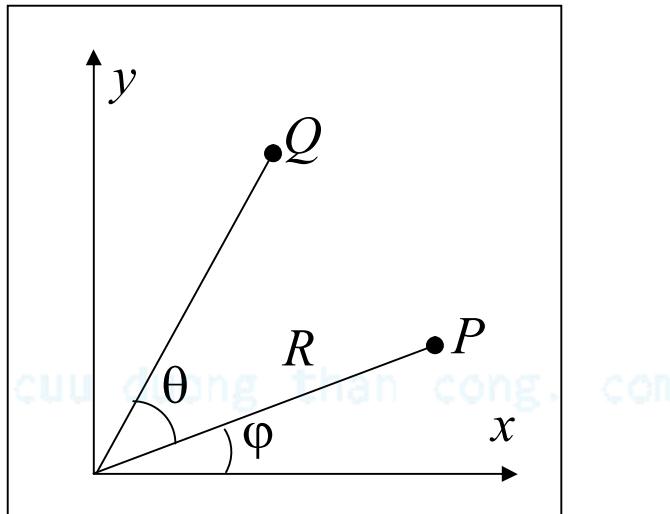


$$\begin{pmatrix} 0.5 & -\sqrt{3}/2 & 0 \\ \sqrt{3}/2 & 0.5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



PHÉP BIẾN ĐỔI HÌNH 2D

□ Phép quay (chứng minh)



$$Q_x = R \cos(\theta + \varphi)$$

$$Q_y = R \sin(\theta + \varphi)$$

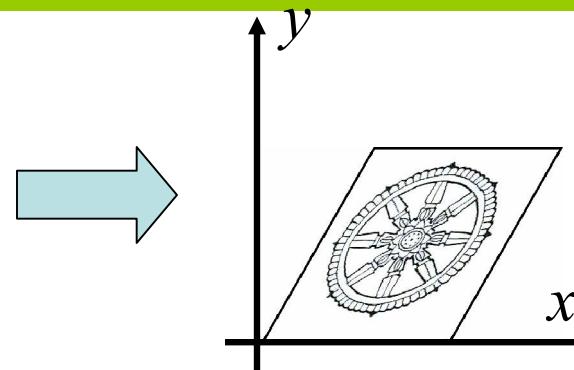
$$Q_x = R \cos \theta \cos \varphi - R \sin \theta \sin \varphi = P_x \cos \theta - P_y \sin \theta$$

$$Q_y = R \sin \theta \cos \varphi + R \cos \theta \sin \varphi = P_x \sin \theta + P_y \cos \theta$$

PHÉP BIẾN ĐỔI HÌNH 2D

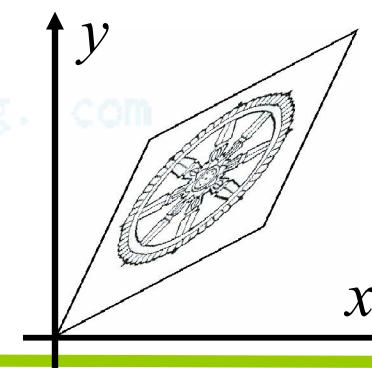
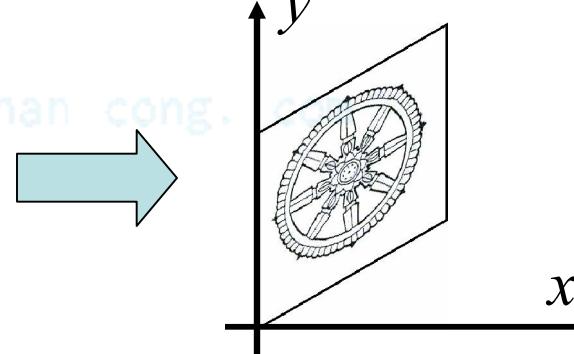
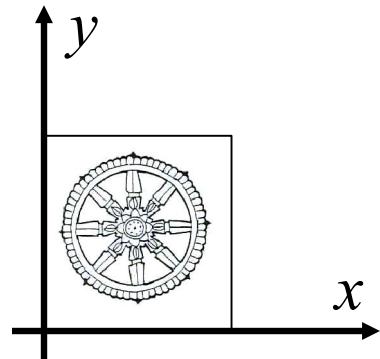
□ Phép trượt

$$\begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



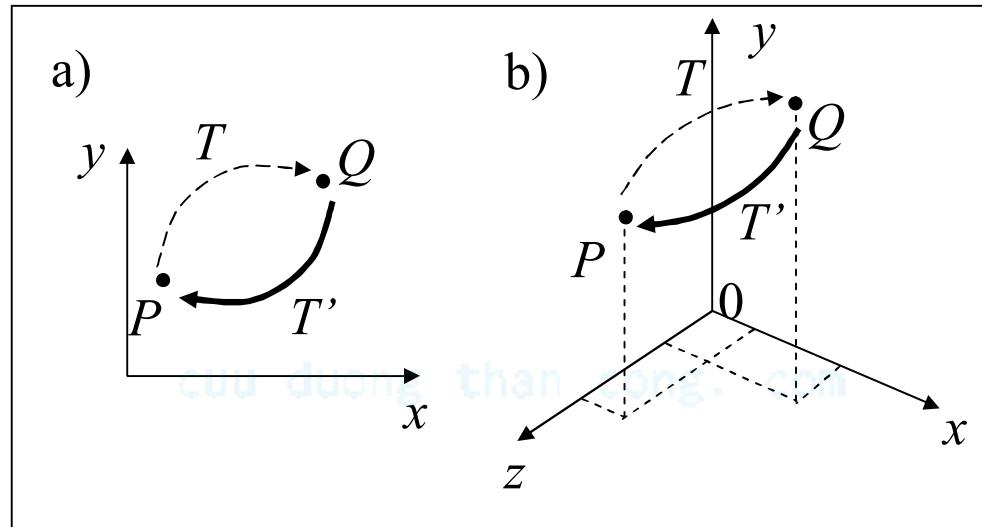
$$\begin{pmatrix} 1 & 0 & 0 \\ g & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & h & 0 \\ g & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



PHÉP BIẾN ĐỔI HÌNH 2D

☐ Nghịch đảo của phép biến đổi affine



$$P = T'Q = M^{-1}Q$$

$$M^{-1} = \frac{1}{\det M} \begin{pmatrix} m_{22} & -m_{12} \\ -m_{21} & m_{11} \end{pmatrix}$$

đa số các trường hợp
cột thứ 3 là (0, 0, 1)

$$\det M = m_{11} m_{22} - m_{12} m_{21}$$

PHÉP BIẾN ĐỔI HÌNH 2D

☐ Nghịch đảo của phép biến đổi affine

Phép biến đổi tỷ lệ

$$M^{-1} = \begin{pmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Phép quay

$$M^{-1} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Phép trượt

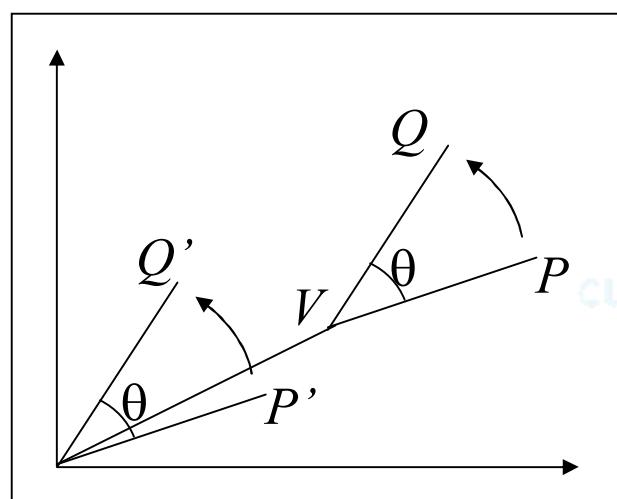
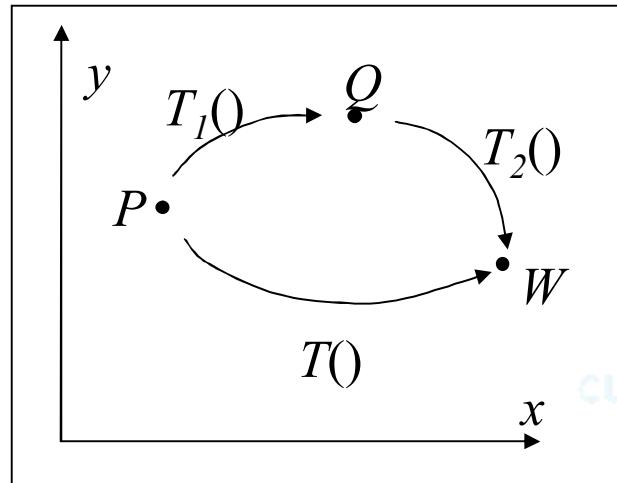
$$M^{-1} = \begin{pmatrix} 1 & -h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Phép tinh tiến

$$M^{-1} = \begin{pmatrix} 1 & 0 & -m_{13} \\ 0 & 1 & -m_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

PHÉP BIẾN ĐỔI HÌNH 2D

□ Hợp các phép biến đổi



$$T_2(T_1 P) = (T_2 T_1) P$$

- Tịnh tiến điểm P với vector $v = (-V_x, -V_y)$
- Quay xung quanh gốc tọa độ góc θ .
- Tịnh tiến điểm P về vị trí cũ.

$$\begin{pmatrix} 1 & 0 & V_x \\ 0 & 1 & V_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -V_x \\ 0 & 1 & -V_y \\ 0 & 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} \cos(\theta) & -\sin(\theta) & d_x \\ \sin(\theta) & \cos(\theta) & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

PHÉP BIẾN ĐỔI HÌNH 2D

Tính chất của phép biến đổi affine

- ☐ Bảo toàn tổ hợp affine của các điểm

$$T(a_1 P_1 + a_2 P_2) = a_1 T(P_1) + a_2 T(P_2) \text{ với } a_1 + a_2 = 1$$

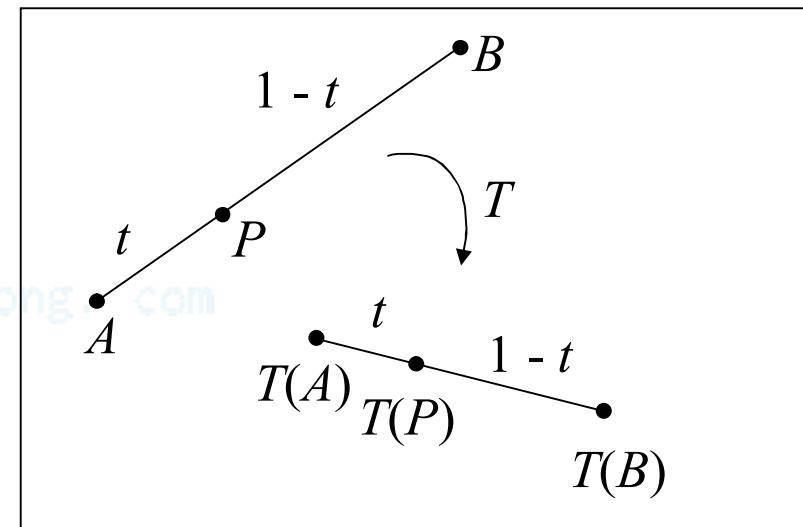
- ☐ Bảo toàn đường thẳng và mặt phẳng

- ☐ Bảo toàn tính song song

- ☐ Ảnh hưởng đến diện tích :

$$\frac{\text{area after transformation}}{\text{area before transformation}} = |\det M|$$

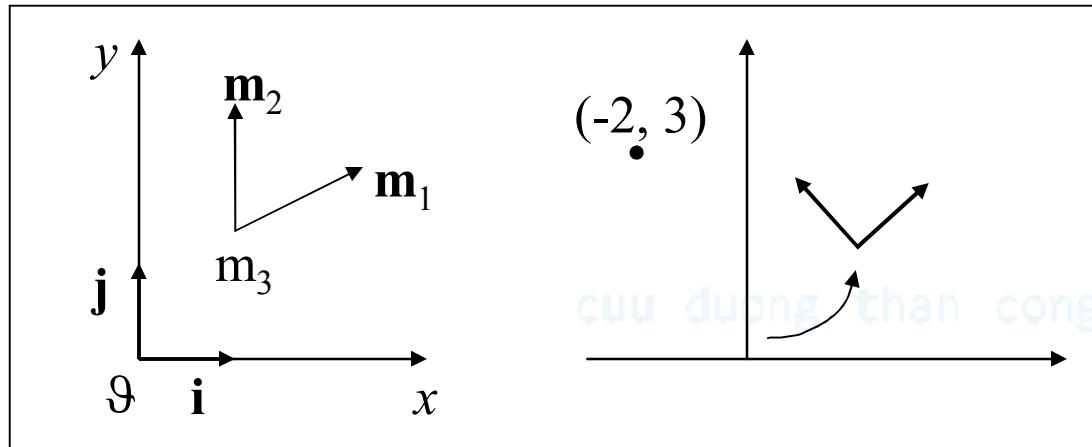
- ☐ Bảo toàn tỷ lệ khoảng cách



PHÉP BIẾN ĐỔI HÌNH 2D

Tính chất của phép biến đổi affine

- ☐ Cột của ma trận M là khung tọa độ sau khi biến đổi



Phép quay xung
quanh điểm (-2, 3)
với góc quay 30^0

$$\begin{pmatrix} 0.866 & -0.5 & 1.232 \\ 0.5 & 0.866 & 1.402 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} = (\mathbf{m}_1 \quad \mathbf{m}_2 \quad \mathbf{m}_3) \quad \mathbf{m}_1 = Mi, \mathbf{m}_2 = Mj, \mathbf{m}_3 = M\vartheta$$

- ☐ Mọi PBD affine là hợp của những PBD affine đơn giản
 $M = (\text{tịnh tiến})(\text{trượt})(\text{tỷ lệ})(\text{quay})$

PHÉP BIẾN ĐỔI HÌNH 3D

□ Công thức tổng quát

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

cuu duong than cong. com

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = M \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

cuu duong than cong. com

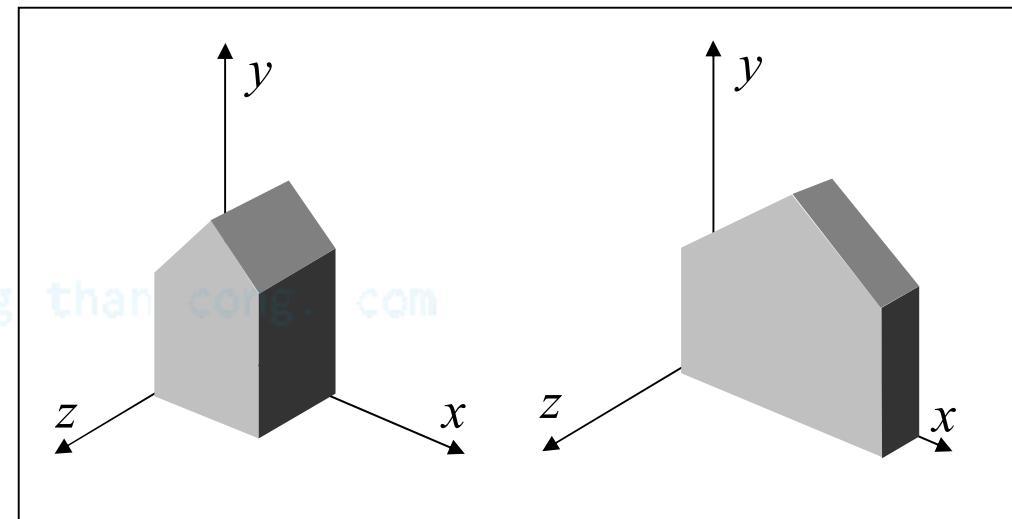
PHÉP BIẾN ĐỔI HÌNH 3D

□ Phép tịnh tiến

$$\begin{pmatrix} 1 & 0 & 0 & m_{14} \\ 0 & 1 & 0 & m_{24} \\ 0 & 0 & 1 & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

□ Phép biến đổi tỷ lệ

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



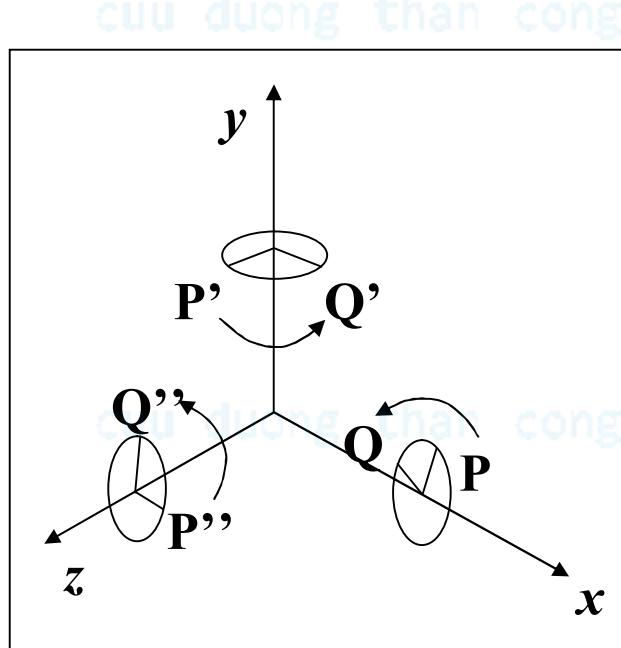
PHÉP BIẾN ĐỔI HÌNH 3D

□ Phép trượt

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ f & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Q = (P_x, fP_x + P_y, P_z)$$

□ Phép quay



- ✓ x-roll, y-roll, z-roll
- ✓ khi góc quay là 90^0 :
 - z-roll: $x \rightarrow y$
 - x-roll: $y \rightarrow z$
 - y-roll: $z \rightarrow x$

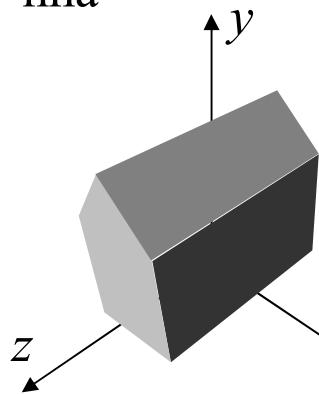
PHÉP BIẾN ĐỔI HÌNH 3D

□ Phép quay

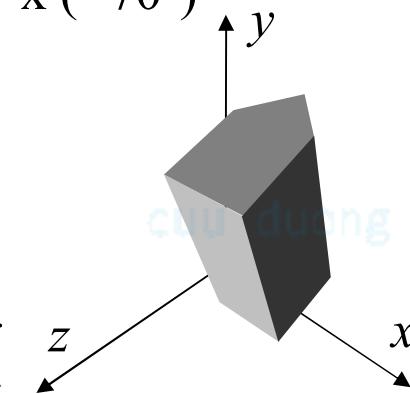
$$R_x(\beta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} R_y(\beta) = \begin{pmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} R_z(\beta) = \begin{pmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

cuuduongthancong.com

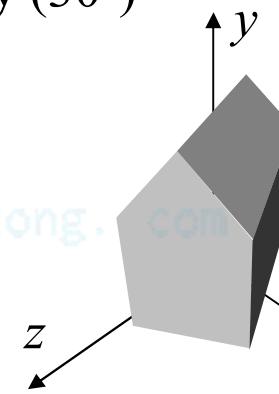
a) ngôi
nhà



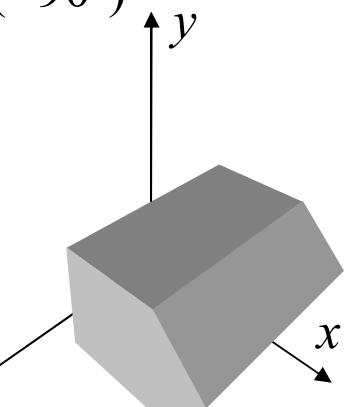
b) quay quanh trục
x (- 70°)



c) quay quanh trục
y (30°)



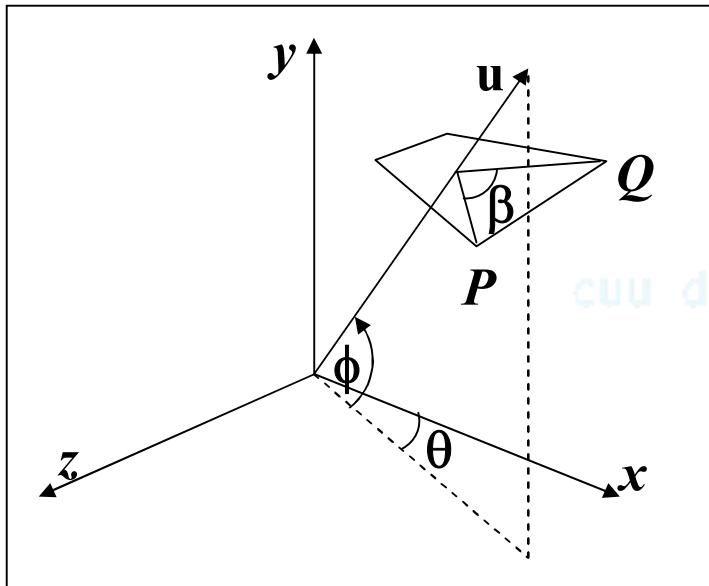
d) quay quanh trục
z (- 90°)



PHÉP BIẾN ĐỔI HÌNH 3D

❑ Hợp các phép biến đổi

– Phép quay xung quanh một trục bất kỳ

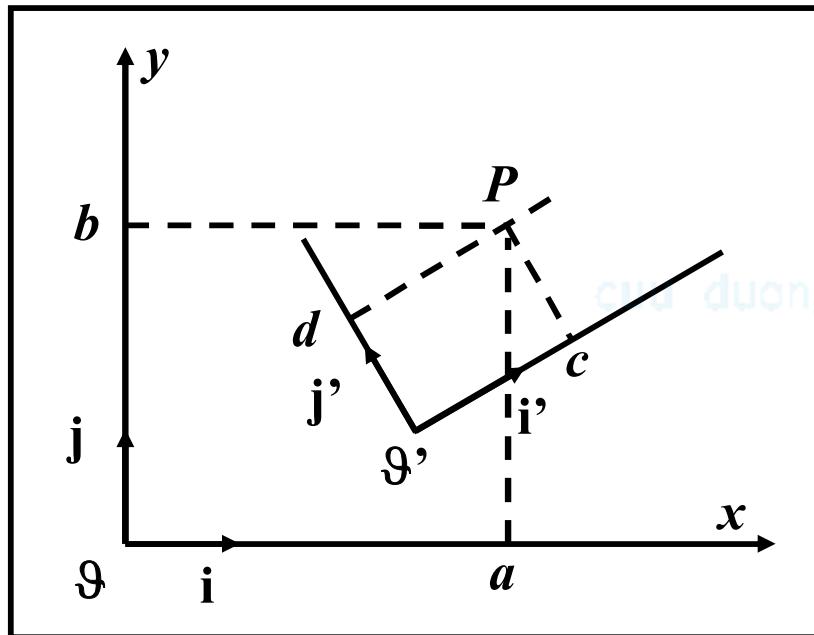


- ✓ $u \rightarrow x$.
- ✓ Quay xung quanh trục x với góc quay β .
- ✓ Khôi phục u .

$$R_u(\beta) = R_y(-\theta)R_z(\phi)R_x(\beta)R_z(-\phi)R_y(\theta)$$

$$\begin{pmatrix} c + (1 - c)u_x^2 & (1 - c)u_yu_x - su_z & (1 - c)u_zu_x + su_y & 0 \\ (1 - c)u_xu_y + su_z & c + (1 - c)u_y^2 & (1 - c)u_zu_y - su_x & 0 \\ (1 - c)u_xu_z - su_y & (1 - c)u_yu_z + su_x & c + (1 - c)u_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

BIẾN ĐỔI HỆ TRỤC TỌA ĐỘ

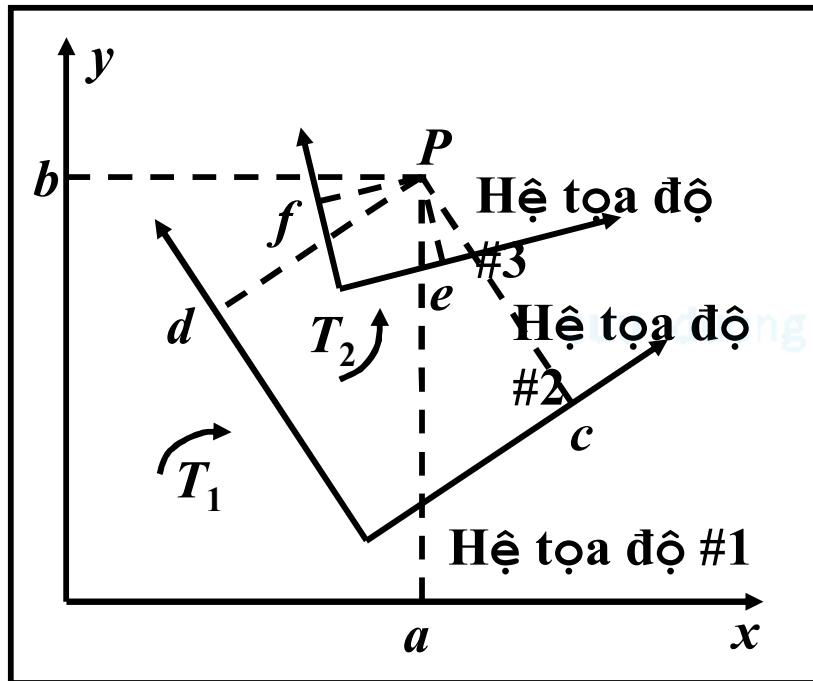


$$(P_x, P_y, 1)^T \rightarrow \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} a \\ b \\ 1 \end{pmatrix} = M \begin{pmatrix} c \\ d \\ 1 \end{pmatrix}$$

BIẾN ĐỔI HỆ TRỤC TỌA ĐỘ

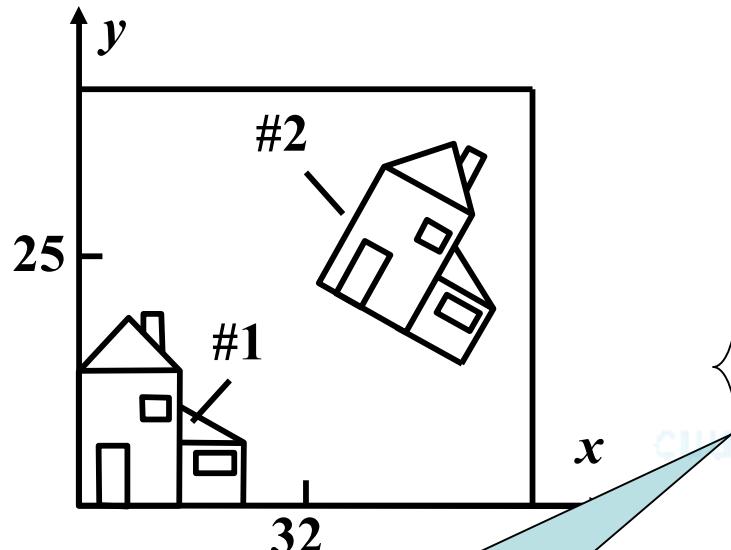
- ☐ Hợp của nhiều phép biến đổi hệ trục tọa độ



$$\begin{pmatrix} a \\ b \\ 1 \end{pmatrix} = M_1 \begin{pmatrix} c \\ d \\ 1 \end{pmatrix} = M_1 M_2 \begin{pmatrix} e \\ f \\ 1 \end{pmatrix}$$

cuuduongthancong.com

SỬ DỤNG PHÉP BIẾN ĐỔI TRONG CHƯƠNG TRÌNH



Có 2 cách vẽ
ngôi nhà #1

Vẽ ngôi nhà
#2 ???

```
glBegin(GL_LINES);
    glVertex2d(V[0].x, V[0].y);
    glVertex2d(V[1].x, V[1].y);
    glVertex2d(V[2].x, V[2].y);
    ...// những điểm còn lại
glEnd();

cvs.moveTo(V[0]);
cvs.lineTo(V[1]);
cvs.lineTo(V[2]);
...// những điểm còn lại
```

```
cvs.setWindow(...);
cvs.setViewport(...);
```

SỬ DỤNG PHÉP BIẾN ĐỔI TRONG CHƯƠNG TRÌNH

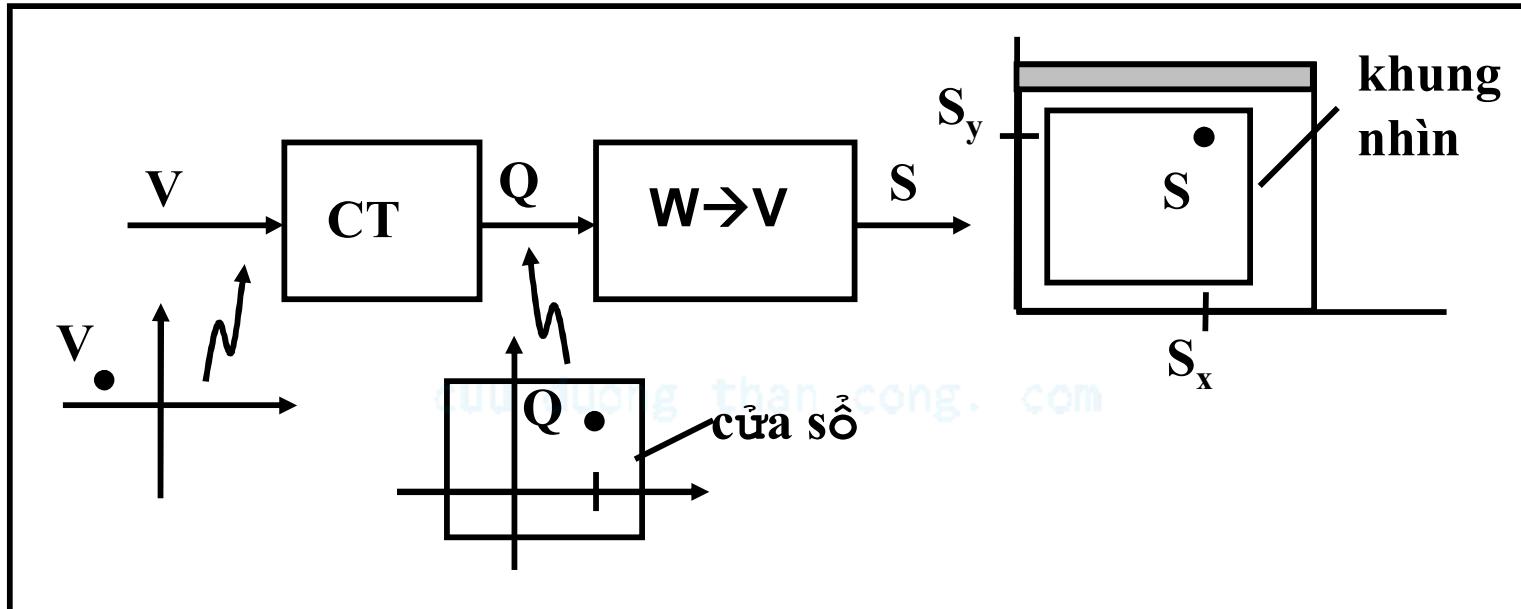
❑ Cách làm phức tạp

- $Q = \text{transform2D}(M, P);$
- `cvs.moveTo(transform2D(M, v[0]));`
`cvs.lineTo(transform2D(M, v[1]));`
`cvs.lineTo(transform2D(M, v[2]));`
...// những điểm còn lại
- Vấn đề là khó tìm ra ma trận M

cuuduongthancong.com

SỬ DỤNG PHÉP BIẾN ĐỔI TRONG CHƯƠNG TRÌNH

❑ Cách làm đơn giản



CT được khởi gán là ma trận
đơn vị:

$$CT = CT^*M$$

```
glScaled(sx, sy, 1.0)  
glTranslated(dx, dy, 0)  
glRotated(angle, 0, 0, 1)
```

SỬ DỤNG PHÉP BIẾN ĐỔI TRONG CHƯƠNG TRÌNH

❑ Cách làm đơn giản

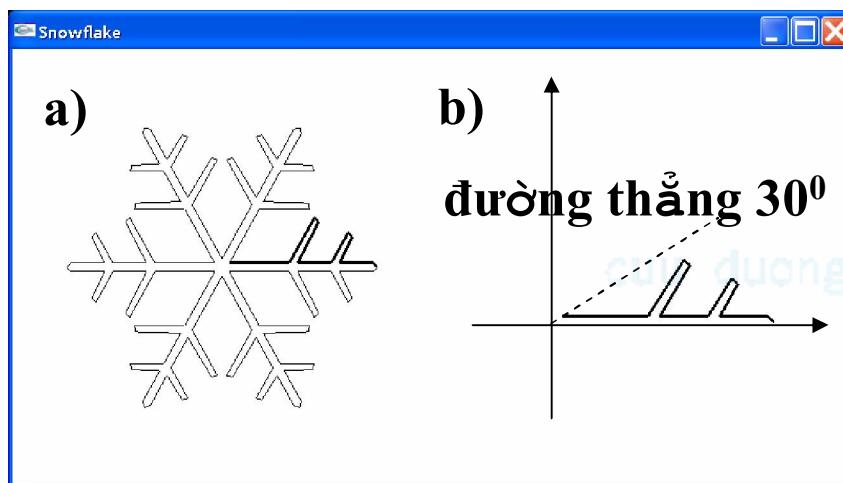
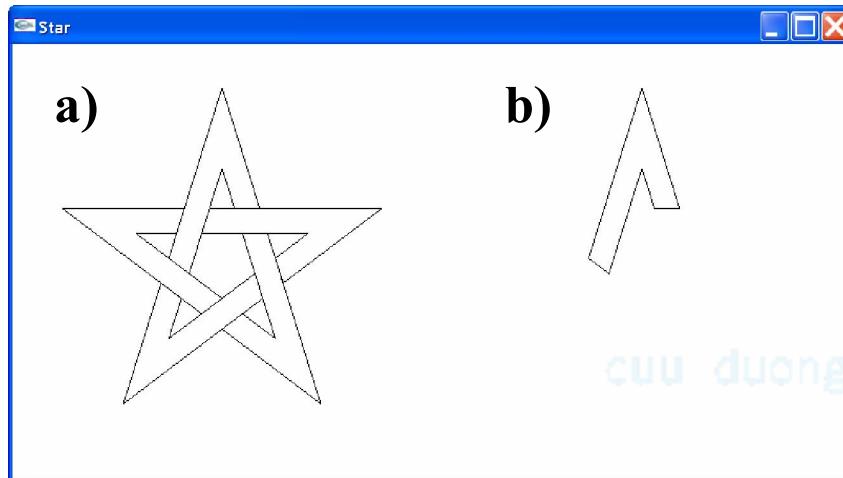
```
cvs.setWindow(...); // thiết lập cửa sổ  
cvs.setViewport(...); // thiết lập khung nhìn  
cvs.initCT(); // bắt đầu với biến đổi đồng nhất  
  
house(); // vẽ ngôi nhà số #1  
cvs.translate2D(32, 25); // CT bây giờ là phép biến đổi tịnh tiến  
cvs.rotate2D(-30); // CT bây giờ bao gồm phép tịnh tiến và phép quay  
house(); // vẽ ngôi nhà số #2
```

SỬ DỤNG PHÉP BIẾN ĐỔI TRONG CHƯƠNG TRÌNH

```
void Canvas::initCT(void) {  
    glMatrixMode(GL_MODELVIEW);  
    glLoadIdentity();  
}  
  
void Canvas::scale2D(double sx, double sy) {  
    glMatrixMode(GL_MODELVIEW);  
    glScaled(sx, sy, 1.0);  
}  
  
void Canvas::translate2D(double dx, double dy) {  
    glMatrixMode(GL_MODELVIEW);  
    glTranslated(dx, dy, 0);  
}  
  
void Canvas::rotate2D(double angle) {  
    glMatrixMode(GL_MODELVIEW);  
    glRotated(angle, 0, 0, 1.0);  
}
```

SỬ DỤNG PHÉP BIẾN ĐỔI TRONG CHƯƠNG TRÌNH

❑ Ví dụ

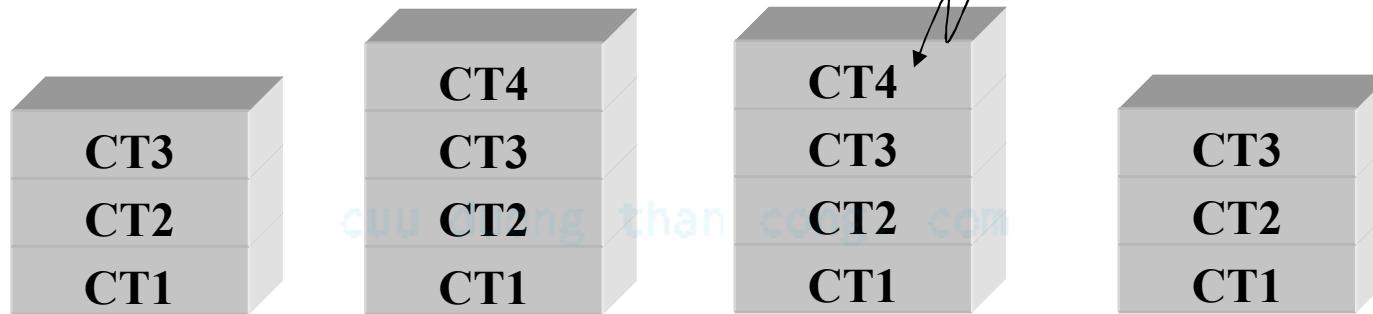


```
for(count=0; count<5; count++)  
{  
    starMotif();  
    cvs.rotate2D(72.0);  
}  
cứu đường thanh công. com  
void drawFlake(){  
    for(count=0; count<6;count++)  
    {  
        flakeMotif();  
        cvs.scale2D(1.0, -1.0);  
        flakeMotif();  
        cvs.scale2D(1.0, -1.0);  
        cvs.rotate2D(60.0);  
    }  
}
```

SỬ DỤNG PHÉP BIẾN ĐỔI TRONG CHƯƠNG TRÌNH

☐ Lưu giữ CT để sau này dùng đến

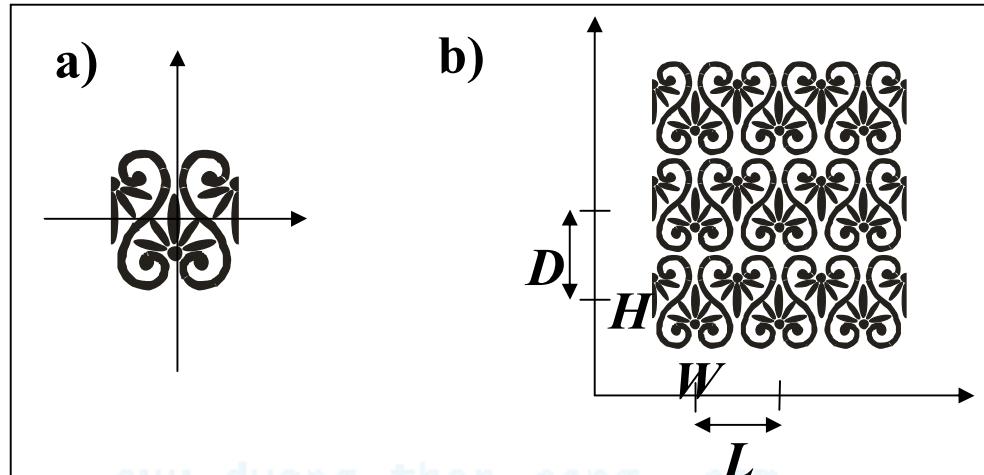
- a) trước
- b) sau khi pushCT()
- c) sau khi rotate2D()
- d) sau khi popCT()



```
void Canvas::pushCT() {  
    glMatrixMode(GL_MODELVIEW);  
    glPushMatrix();  
}  
void Canvas::popCT() {  
    glMatrixMode(GL_MODELVIEW);  
    glPopMatrix();  
}
```

SỬ DỤNG PHÉP BIẾN ĐỔI TRONG CHƯƠNG TRÌNH

❑ Ví dụ



```
cvs.pushCT();           // so we can return here
cvs.translate2D(W, H);  // position for the first motif
for(row = 0; row < 3; row++) { // draw each row
    pushCT();
    for(col = 0 ; col < 3; col++) {
        motif();
        cvs.translate2D(L, 0); } //move to the right
    cvs.popCT(); // back to the start of this row
    cvs.translate2D(0, D); } //move up to the next row
cvs.popCT(); //back to where we started
```

Trường Đại Học Bách Khoa TP Hồ Chí Minh
Khoa Khoa học & Kỹ thuật Máy tính



ĐỒ HỌA MÁY TÍNH

CHƯƠNG 6:

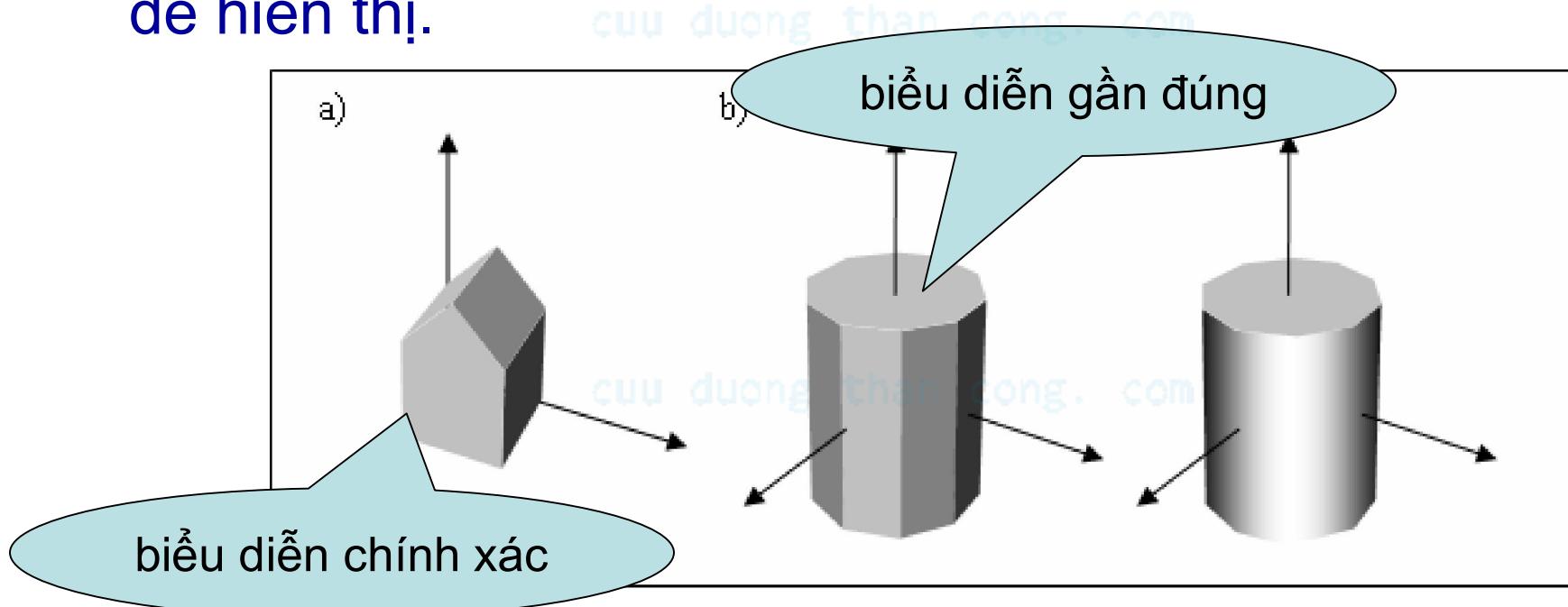
**MÔ HÌNH HÓA ĐỐI TƯỢNG
3D BẰNG LƯỚI ĐA GIÁC**

NỘI DUNG TRÌNH BÀY

- Lưới đa giác
- Khối đa diện
- Khối quét
- Lưới xấp xỉ mặt cong
- Mặt chứa cạnh thẳng
- Mặt tròn xoay
- Mặt bậc hai
- Mặt siêu bậc hai
- Mặt biểu diễn bởi hàm tường minh

LƯỚI ĐA GIÁC

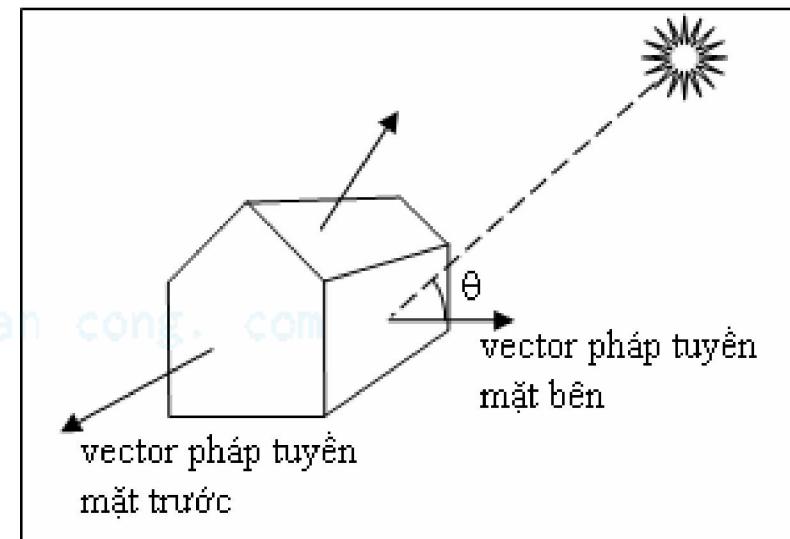
- ❑ Lưới đa giác là tập hợp các đa giác phẳng (các mặt) tạo nên bề mặt đối tượng, là phương pháp chuẩn để biểu diễn đối tượng.
- ❑ Lý do sử dụng lưới đa giác: dễ biểu diễn (tập hợp các đỉnh), ít thuộc tính (đỉnh, vector pháp tuyến), dễ biến đổi, dễ hiển thị.



MÔ HÌNH HÓA KHỐI RẮN BẰNG LƯỚI

- ❑ Khối rắn:các mặt xếp khít với nhau đóng kín một phần không gian
- ❑ “Bề mặt” mỏng: các mặt không đóng kín một phần kg
- ❑ Lưới đa giác:
 - là tập hợp các đa giác
 - được biểu diễn bởi danh sách các đa giác và thông tin hướng

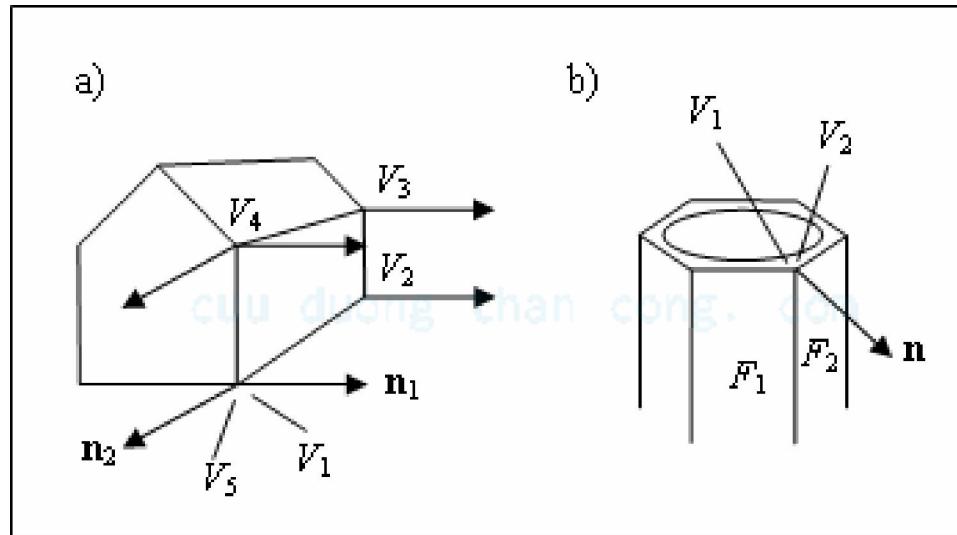
hướng cho biết mặt nhận được bao nhiêu ánh sáng và thường được dùng trong quá trình tô màu



MÔ HÌNH HÓA KHỐI RĂN BẰNG LƯỚI

□ Pháp tuyến đỉnh và pháp tuyến mặt

- gán mỗi đỉnh thuộc mặt một vector pháp tuyến
- V_1 và V_5 tuy cùng 1 điểm nhưng dùng pháp tuyến khác nhau (tô màu phẳng)
- V_1 và V_5 dùng pháp tuyến giống nhau (tô màu trơn). Vector pháp tuyến này vuông góc với mặt cong tại điểm đang xét



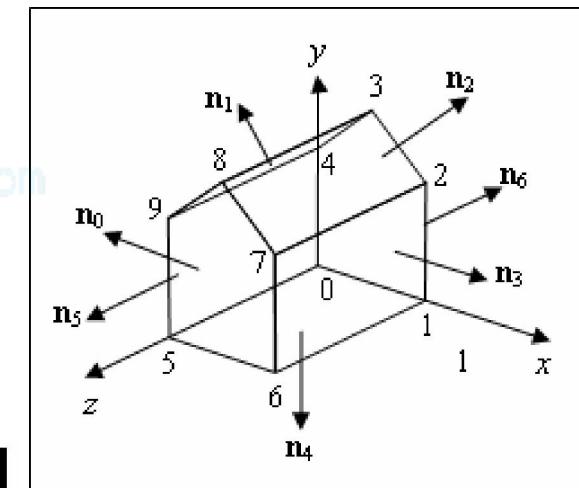
ĐỊNH NGHĨA LƯỚI ĐA GIÁC

- ❑ Lưới đa giác là tập hợp các đa giác mà mỗi đỉnh của từng mặt được gán một vector pháp tuyến
- ❑ Cách làm hiệu quả: tổ chức thành ba danh sách. Danh sách đỉnh (thông tin vị trí), danh sách pháp tuyến (thông tin hướng), danh sách mặt (thông tin liên kết)

đỉnh	x	y	z
0	0	0	0
1	1	0	0
2	1	1	0
3	0.5	1.5	0
4	0	1	0
5	0	0	1
6	1	0	1
7	1	1	1
8	0.5	1.5	1
9	0	1	1

pháp tuyến	n_x	n_y	n_z
0	-1	0	0
1	-0.707	0.707	0
2	0.707	0.707	0
3	1	0	0
4	0	-1	0
5	0	0	1
6	0	0	-1

mặt	các đỉnh	các pháp tuyến
0 (trái)	0, 5, 9, 4	0, 0, 0, 0
1 (mái nhà trái)	3, 4, 9, 8	1, 1, 1, 1
2 (mái nhà phải)	2, 3, 8, 7	2, 2, 2, 2
3 (phải)	1, 2, 7, 6	3, 3, 3, 3
4 (đáy)	0, 1, 6, 5	4, 4, 4, 4
5 (trước)	5, 6, 7, 8, 9	5, 5, 5, 5, 5
6 (sau)	0, 4, 3, 2, 1	6, 6, 6, 6, 6



ĐỊNH NGHĨA LƯỚI ĐA GIÁC

□ Tìm vector pháp tuyến:

- nếu mặt là phẳng \rightarrow pháp tuyến mặt là pháp tuyến đỉnh.
- dùng tích vô hướng tính pháp tuyến $m = (V1 - V2) \times (V3 - V4)$.
- hai vấn đề: (1) khi hai vector gần song song với nhau (2) đa giác không thực sự phẳng

$$m_x = \sum_{i=0}^{N-1} (y_i - y_{next(i)})(z_i + z_{next(i)})$$

- $next(j) = (j + 1) \bmod N$

$$m_y = \sum_{i=0}^{N-1} (z_i - z_{next(i)})(x_i + x_{next(i)})$$

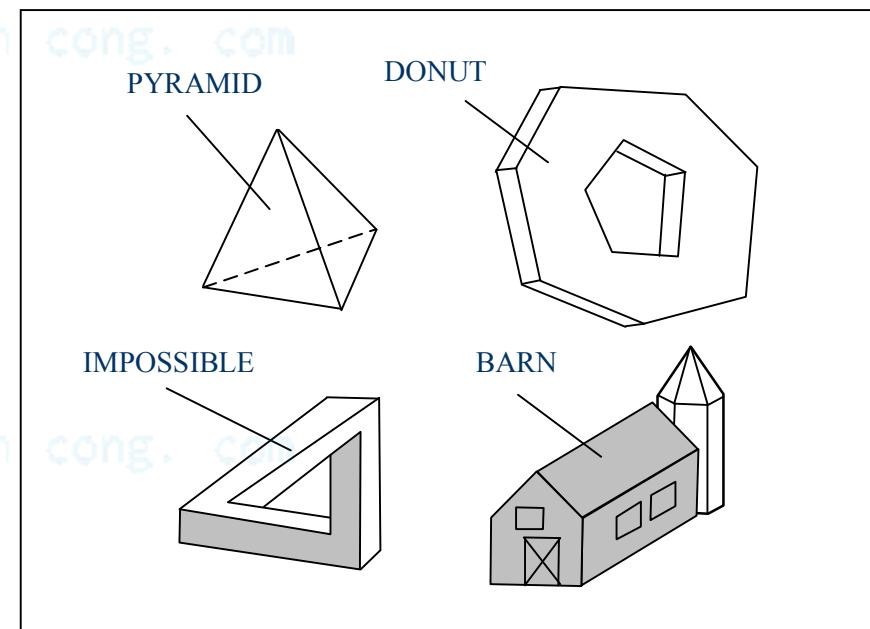
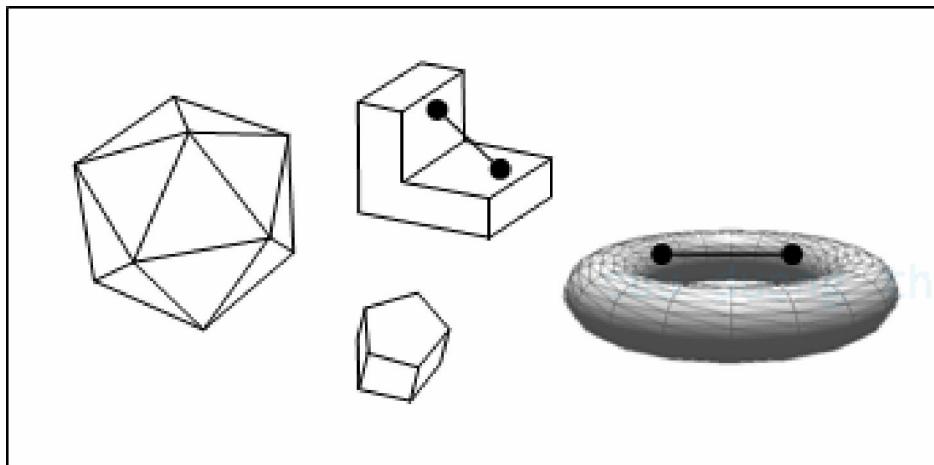
- các đỉnh duyệt theo CCW

$$m_z = \sum_{i=0}^{N-1} (x_i - x_{next(i)})(y_i + y_{next(i)})$$

- m chỉ ra phía ngoài

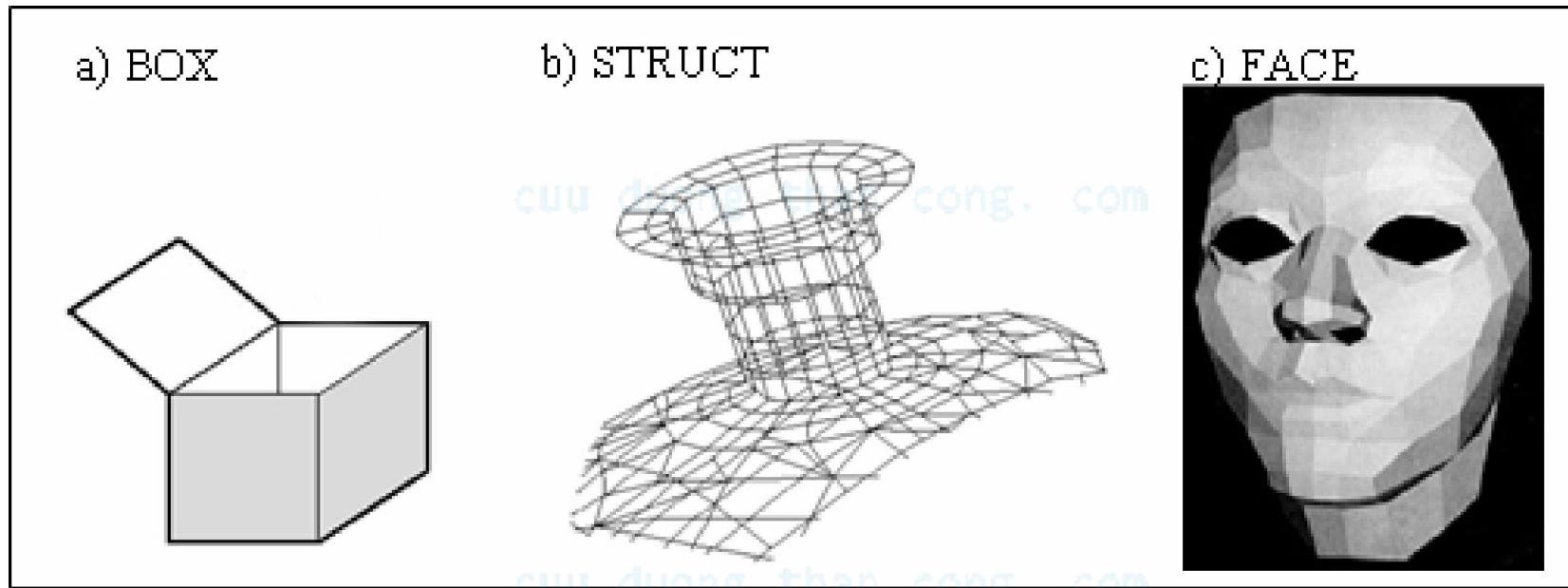
TÍNH CHẤT CỦA LƯỚI ĐA GIÁC

- ❑ Tính đặc; nếu đóng kín một phần không gian
- ❑ Tính liên thông: giữa hai đỉnh bất kỳ có 1 đường dẫn
- ❑ Tính đơn giản: không có lỗ hổng
- ❑ Tính phẳng: các mặt đều là đa giác phẳng (e.g tam giác)
- ❑ Tính lồi



MÔ HÌNH HÓA ĐT KHÔNG ĐẶC

- ☐ Đối tượng không đặc được coi là “cái vỏ” có chiều dày vô cùng bé.



LƯỚI ĐA GIÁC TRONG CT

```
class VertexID{  
public:  
    int vertIndex; //index of this vertex in the vertex list  
    int normIndex; // index of this vertex's normal  
};  
class Face{  
public:  
    int nVerts; // number of vertice in this face  
    VertexID* vert; // the list of vertex and normal index  
    Face() { nVerts = 0; vert = NULL; }  
    ~Face() { delete[] vert; nVerts = 0; }  
};
```

LƯỚI ĐA GIÁC TRONG CT

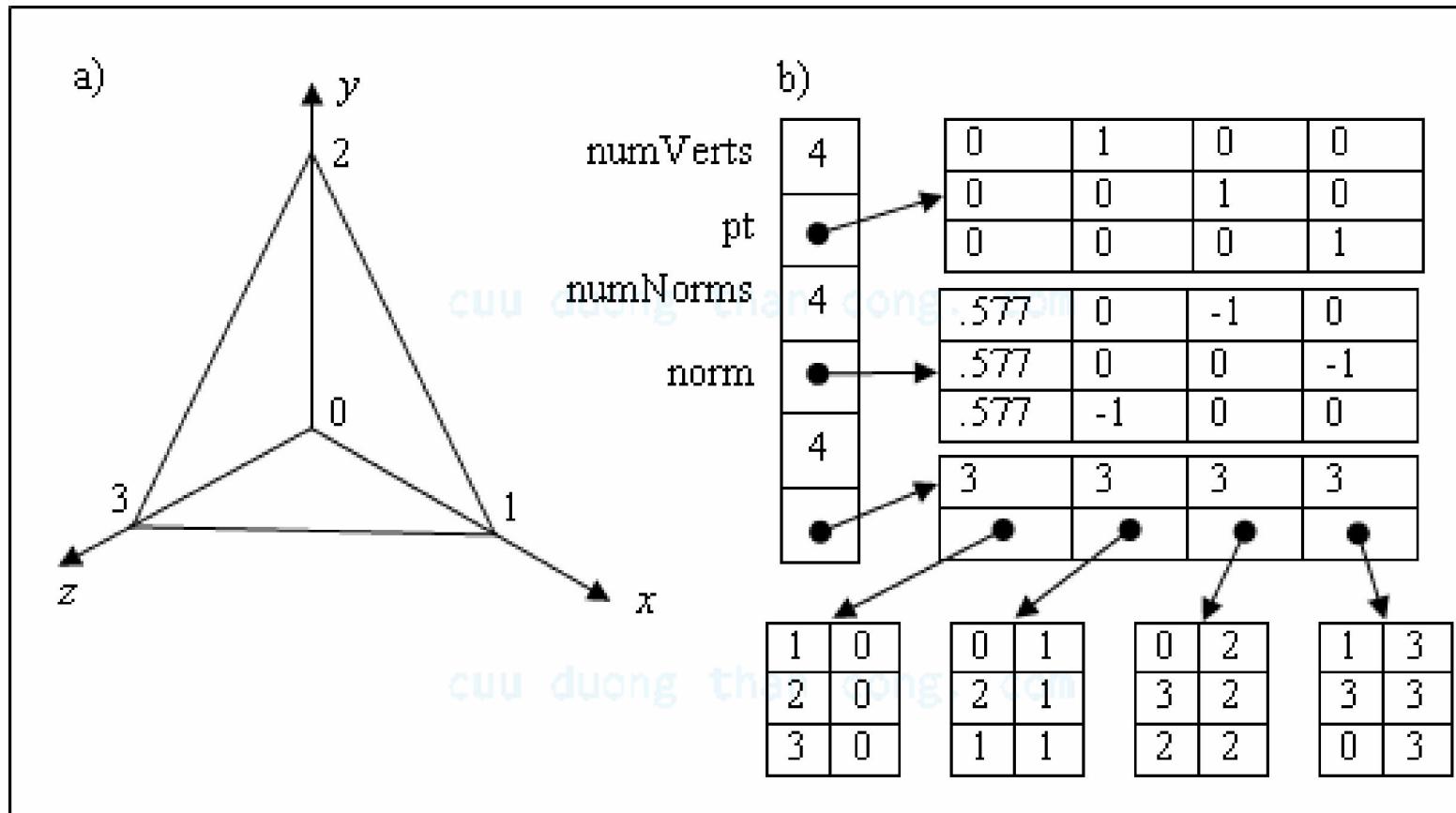
```
class Mesh {  
private:  
    int      numVerts; // number of vertices in the mesh  
    Point3*   pt; // array of 3D vertices  
    int      numNormals; // number of normal vectors for the mesh  
    Vector3*  norm; // array of normals  
    int      numFaces; // number of faces in the mesh  
    Face*    face; // array of face data  
    // ... others to be added later  
public:  
    Mesh();  
    ~Mesh();  
    int  readFile(char* fileName);  
    // ... others  
};
```

LƯỚI ĐA GIÁC TRONG CT

```
void Mesh::draw()
{
    for (int f = 0; f < numFaces; f++)
    {
        glBegin(GL_POLYGON);
        for (int v = 0; v < face[f].nVerts; v++)
        {
            int    in = face[f].vert[v].normIndex;
            int    iv = face[f].vert[v].vertIndex;
            glNormal3f(norm[in].x, norm[in].y, norm[in].z);
            glVertex3f(pt[iv].x, pt[iv].y, pt[iv].z);
        }
        glEnd();
    }
}
```

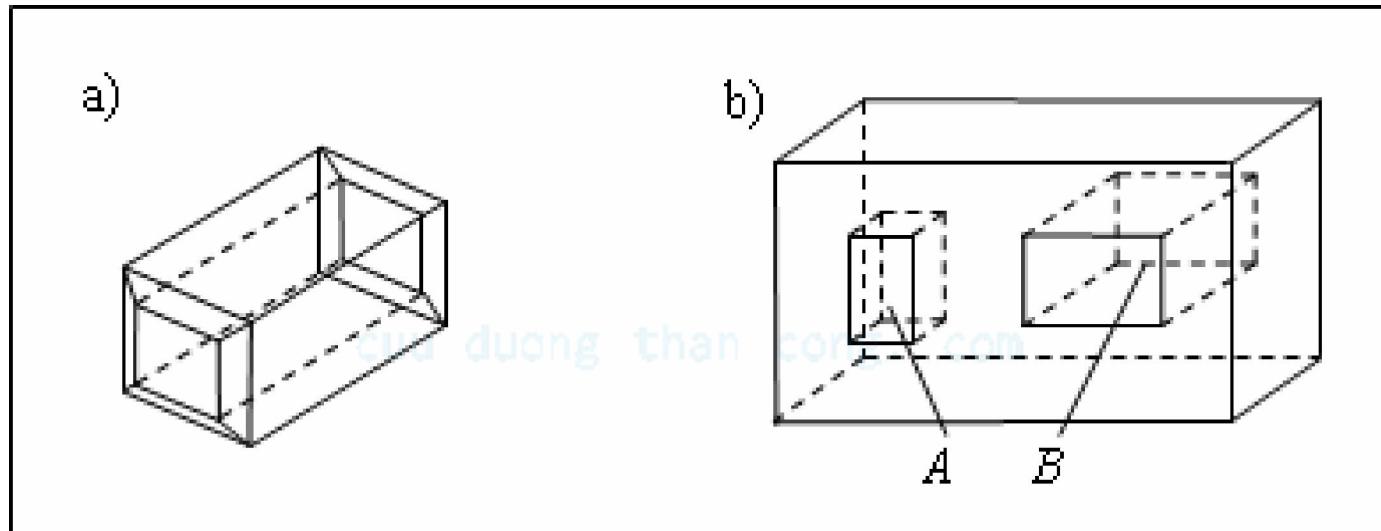
LƯỚI ĐA GIÁC TRONG CT

❑ Ví dụ



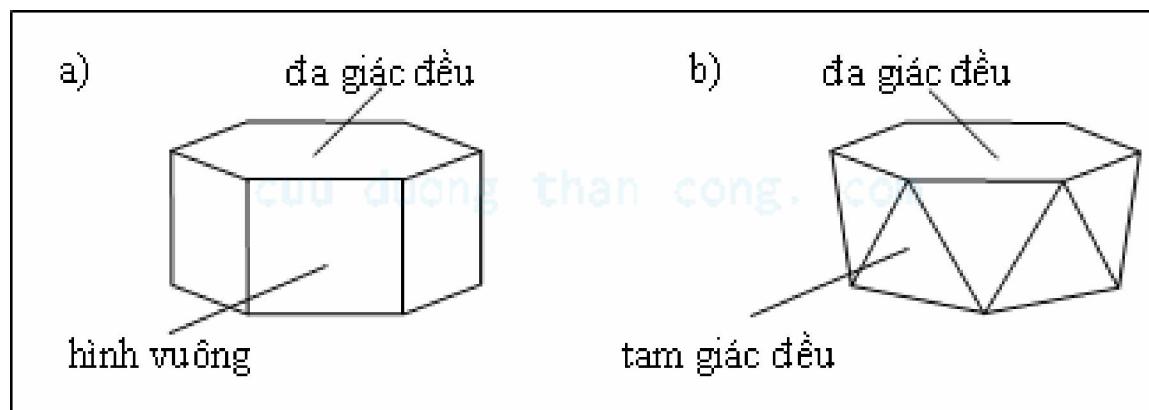
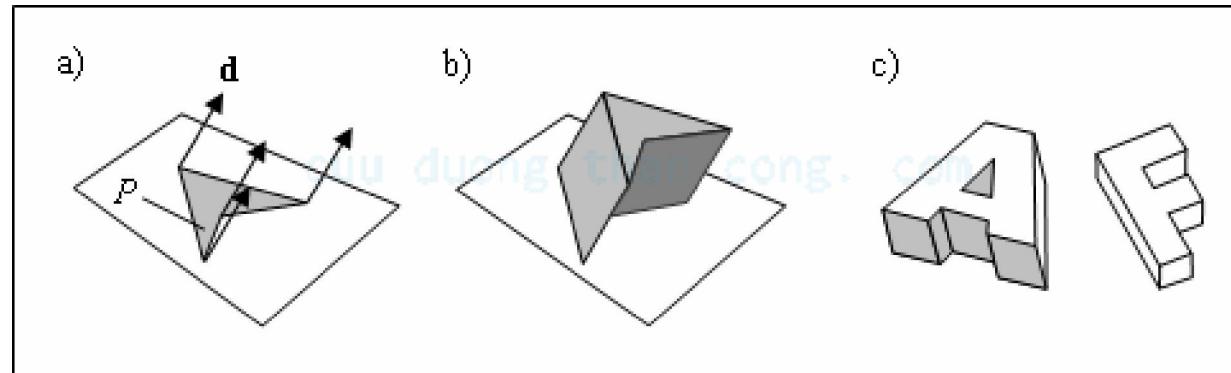
KHỐI ĐA DIỆN

- ❑ Định nghĩa: là lưới liên thông của các đa giác phẳng đơn giản. Các đa giác này bao bọc một kg giới hạn.
- ❑ Tính chất: (1) mỗi cạnh thuộc 2 mặt, (2) đỉnh là giao điểm của ít nhất 3 cạnh, (3) các mặt không xuyên qua nhau.
- ❑ Công thức Euler: (1) $V + F - E = 2$ (cube $V=8, F=6, E=12$), (2) $V + F - E = 2 + H - 2G$ (H : tổng lỗ hổng nằm trên các mặt, G : tổng lỗ hổng xuyên qua đa diện)



HÌNH LĂNG TRỤ VÀ PHẢN LĂNG TRỤ

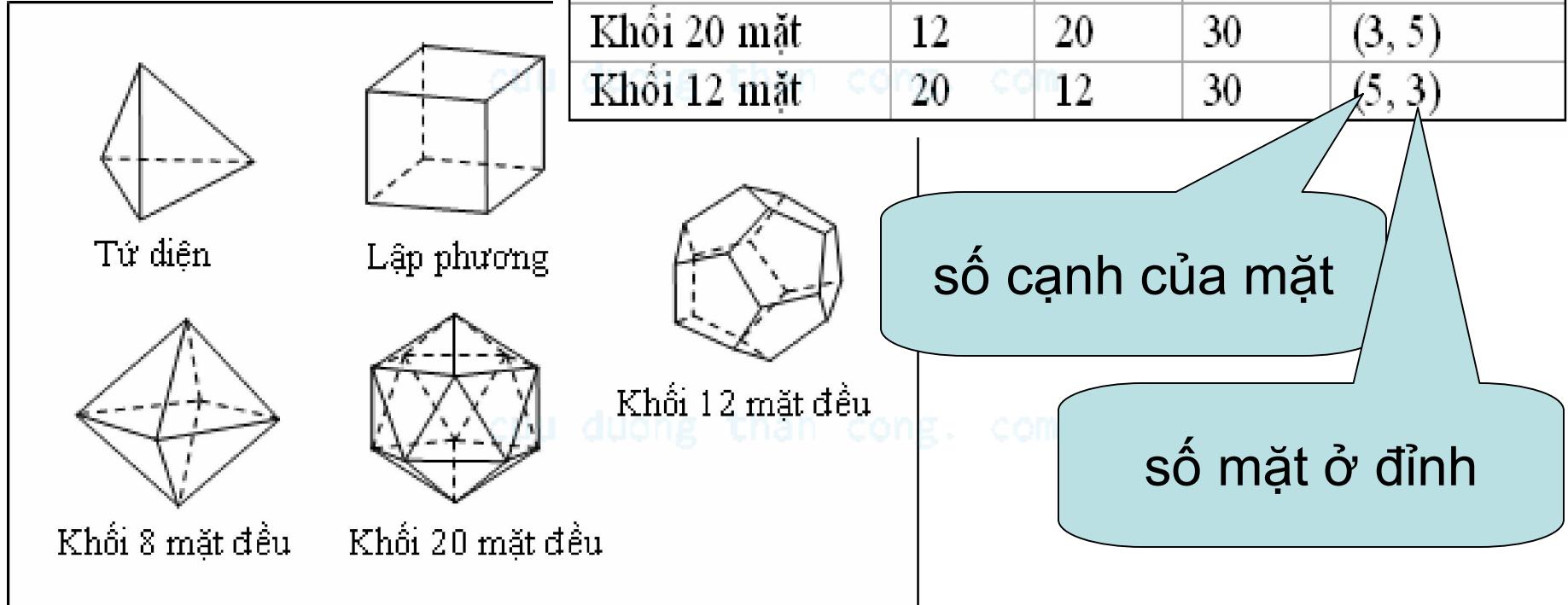
- Hình lăng trụ: quét đa giác dọc theo đoạn thẳng. Khi d vuông góc với đa giác thì lăng trụ là lăng trụ đứng
- Lăng trụ đều: đa giác là đa giác đều
- Phản lăng trụ: đa giác đỉnh quay $180/n$ độ so với đáy



KHỐI ĐA DIỆN ĐỀU (PLATONIC)

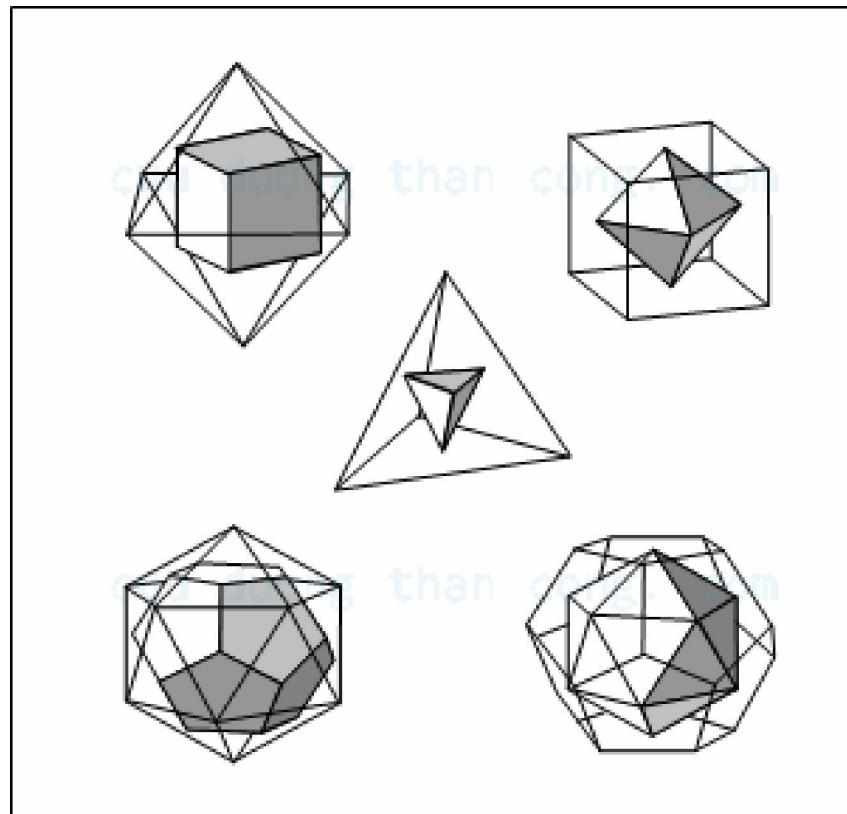
□ Định nghĩa: các mặt bằng nhau và đều là đa giác đều

Khối	V	F	E	Schlafli
Tứ diện	4	4	6	(3, 3)
Lập phương	8	6	12	(4, 3)
Khối 8 mặt	6	8	12	(3, 4)
Khối 20 mặt	12	20	30	(3, 5)
Khối 12 mặt	20	12	30	(5, 3)



KHỐI ĐA DIỆN ĐỀU (PLATONIC)

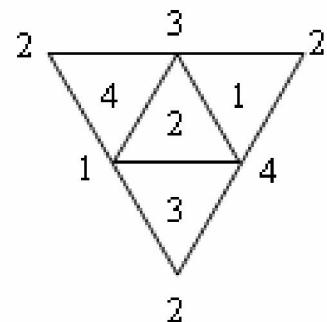
- ❑ Khối đa diện đều đối ngẫu: D là đối ngẫu của P thì đỉnh của D là tâm của P \rightarrow D nối tâm hai mặt kề nhau của P.
- ❑ D có số cạnh bằng số đỉnh của P và số đỉnh bằng số cạnh của P. Nếu P có giá trị Schläfli là (p, q) thì D có giá trị Schläfli là (q, p)



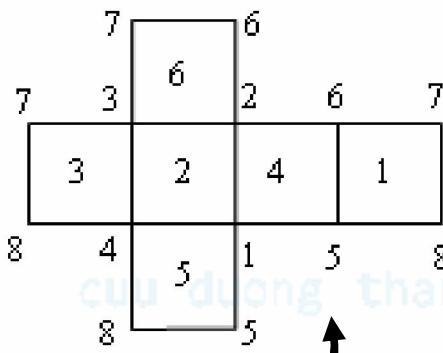
KHỐI ĐA DIỆN ĐỀU (PLATONIC)

□ Mô hình của khối đa diện đều

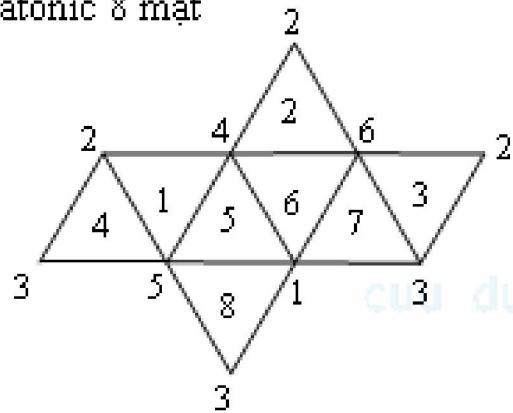
a) Khối tứ diện



b) Khối lập phương



c) Khối Platonic 8 mặt



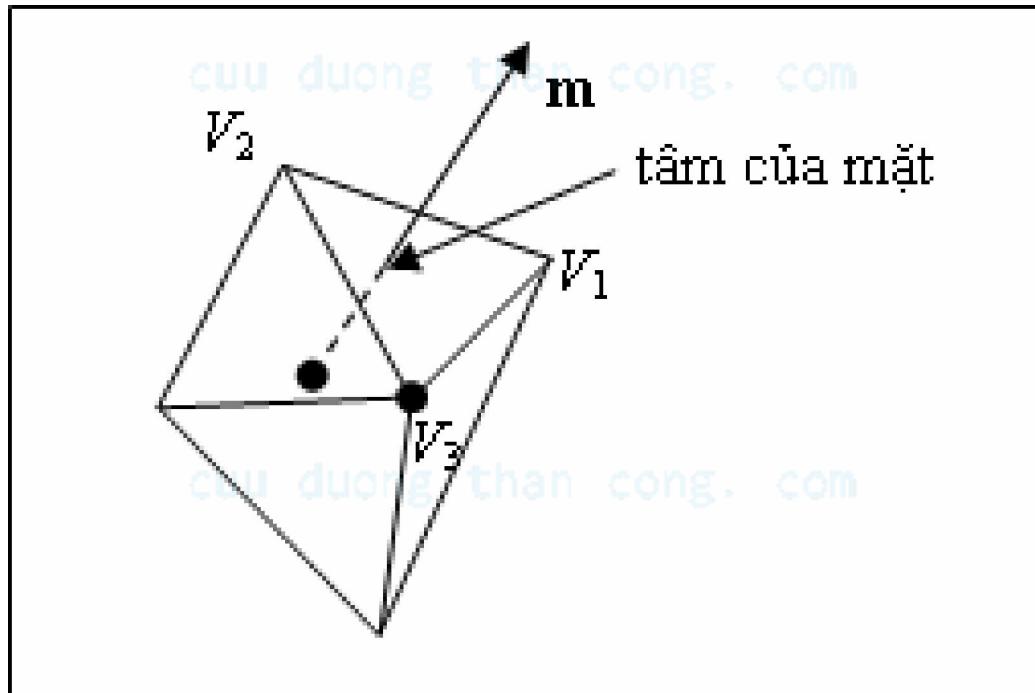
-mặt 4 của lập phương gồm 1,5,6,2 → đỉnh 4 của (c) là giao của các mặt 1,5,6,2

-tứ diện tự đối ngẫu → danh sách các đỉnh của mặt k sẽ trùng với danh sách các mặt giao nhau tại k.

KHỐI ĐA DIỆN ĐỀU (PLATONIC)

□ Pháp tuyến của khối Platonic

$$m = \frac{V_1 + V_2 + V_3}{3}$$



KHỐI ĐA DIỆN ĐỀU (PLATONIC)

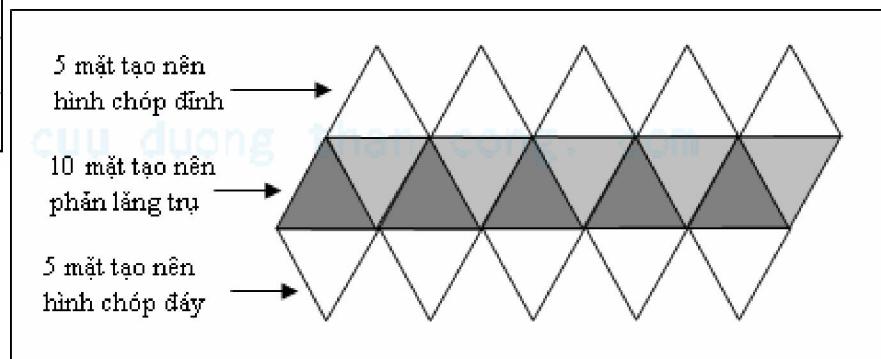
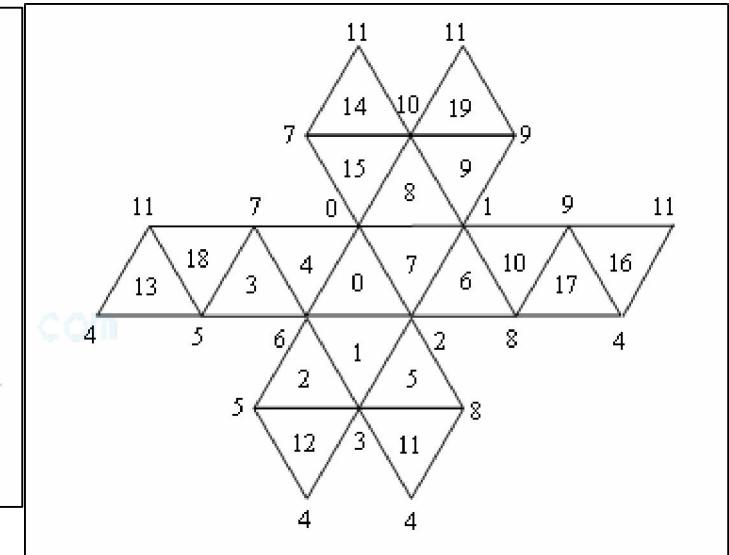
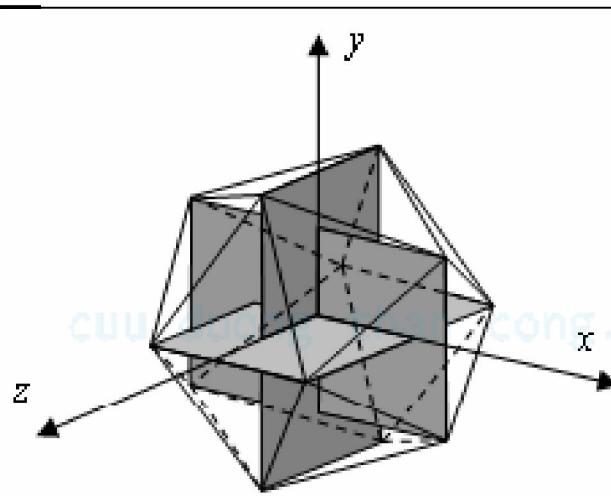
☐ **Khối tứ diện đều**: có thể nội tiếp trong khối lập phương sao cho các đỉnh trùng với các đỉnh của lập phương, các cạnh nằm trên các mặt. Giả sử khối lập phương có các đỉnh là $(\pm 1, \pm 1, \pm 1)$, một đỉnh của tứ diện là $(1, 1, 1)$

Danh sách đỉnh				Danh sách mặt	
<u>Đỉnh</u>	<u>x</u>	<u>y</u>	<u>z</u>	<u>Mặt</u>	<u>Các đỉnh</u>
0	1	1	1	0	1, 2, 3
1	1	-1	-1	1	0, 3, 2
2	-1	-1	1	2	0, 1, 3
3	-1	1	-1	3	0, 2, 1

KHỐI ĐA DIỆN ĐỀU (PLATONIC)

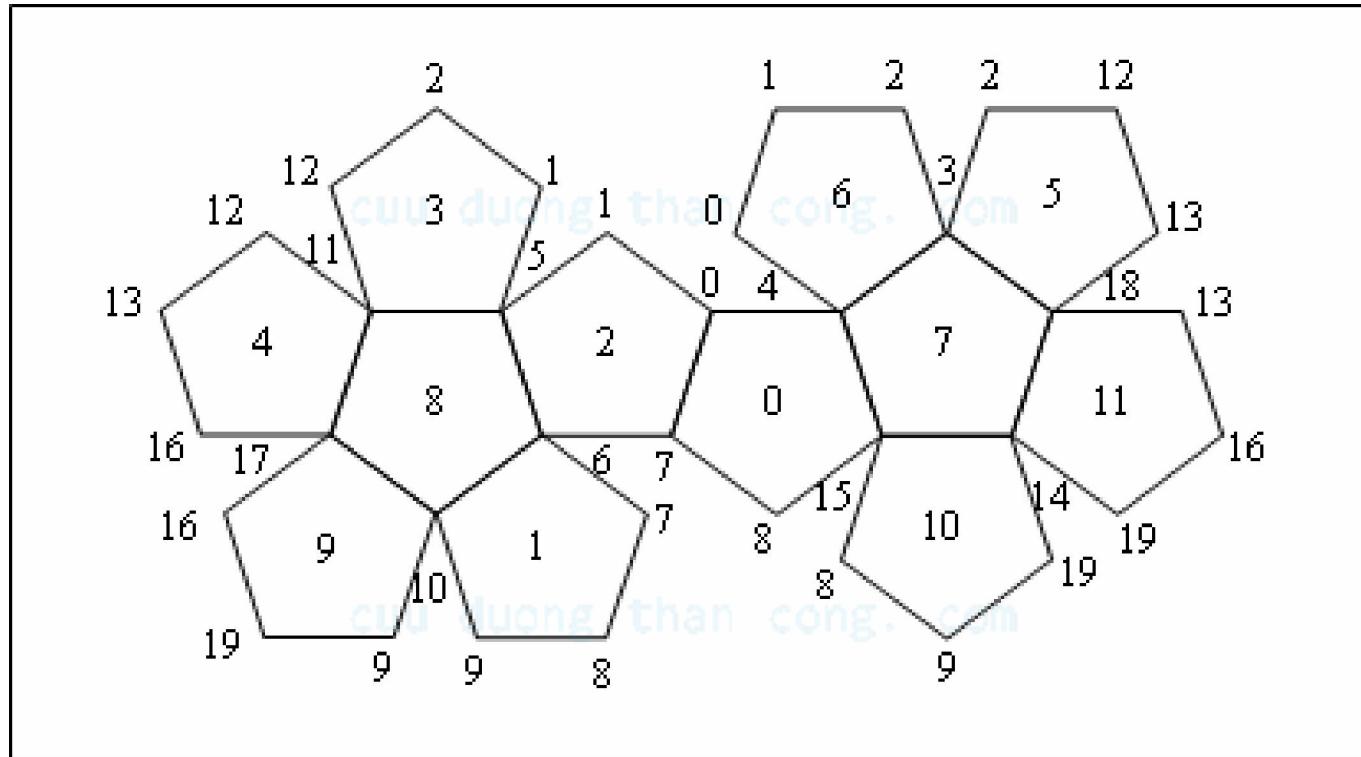
- **Khối Platonic 20 mặt:** xây dựng bằng dùng 3 hình chữ nhật vàng có cạnh dài bằng 1, cạnh ngắn là $\tau = (\sqrt{5} - 1)/2 = 0.618$

Định	x	y	z
0	0	1	τ
1	0	1	$-\tau$
2	1	τ	0
3	1	$-\tau$	0
4	0	-1	$-\tau$
5	0	-1	τ
6	τ	0	1
7	$-\tau$	0	1
8	τ	0	-1
9	$-\tau$	0	-1
10	-1	τ	0
11	-1	$-\tau$	0



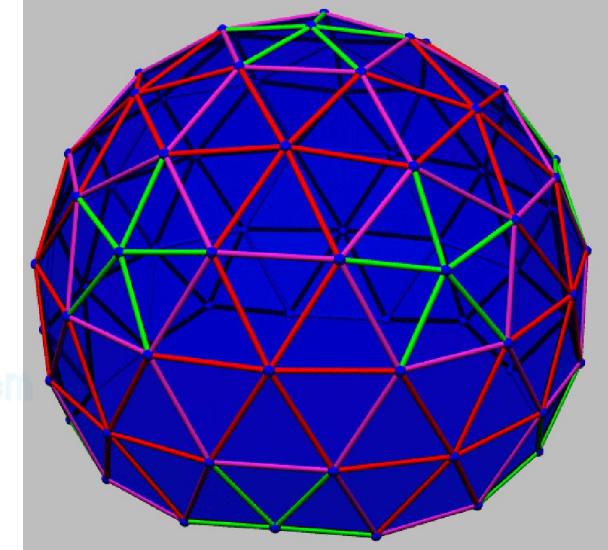
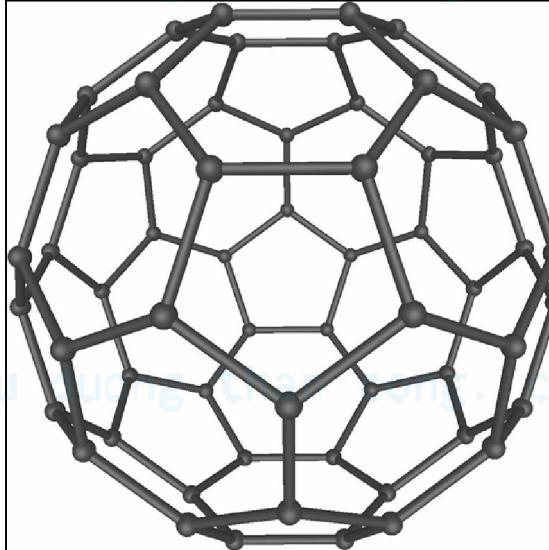
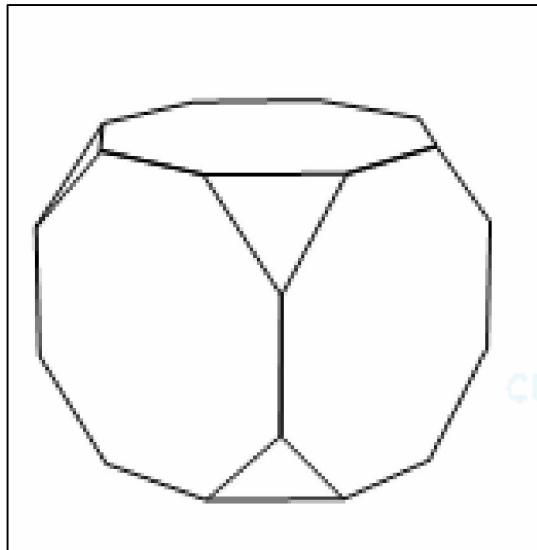
KHỐI ĐA DIỆN ĐỀU (PLATONIC)

- Khối Platonic 12 mặt: là khối đối ngẫu của khối Platonic 20 mặt → đỉnh k của khối này nằm ở tâm mặt k của khối 20 mặt



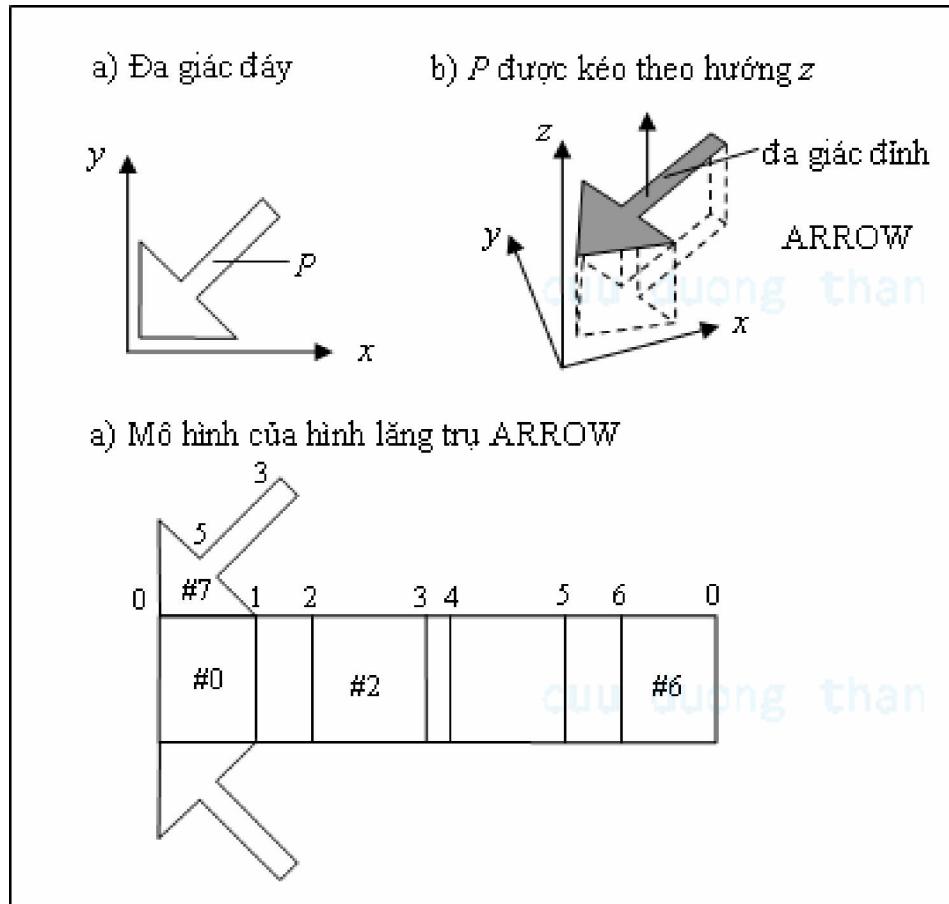
KHỐI ĐA DIỆN NỬA ĐỀU

- Các mặt vẫn là đa giác đều nhưng có nhiều loại mặt khác nhau:
 - hình lập phương vát góc
 - khối cầu bucky
 - khối vòm geodesic



KHỐI QUÉT

- ❑ Được tạo bằng cách quét một hình 2D trong không gian
- ❑ Hình lăng trụ: pháp tuyến đỉnh là pháp tuyến mặt



Xây dựng lưới đa giác:

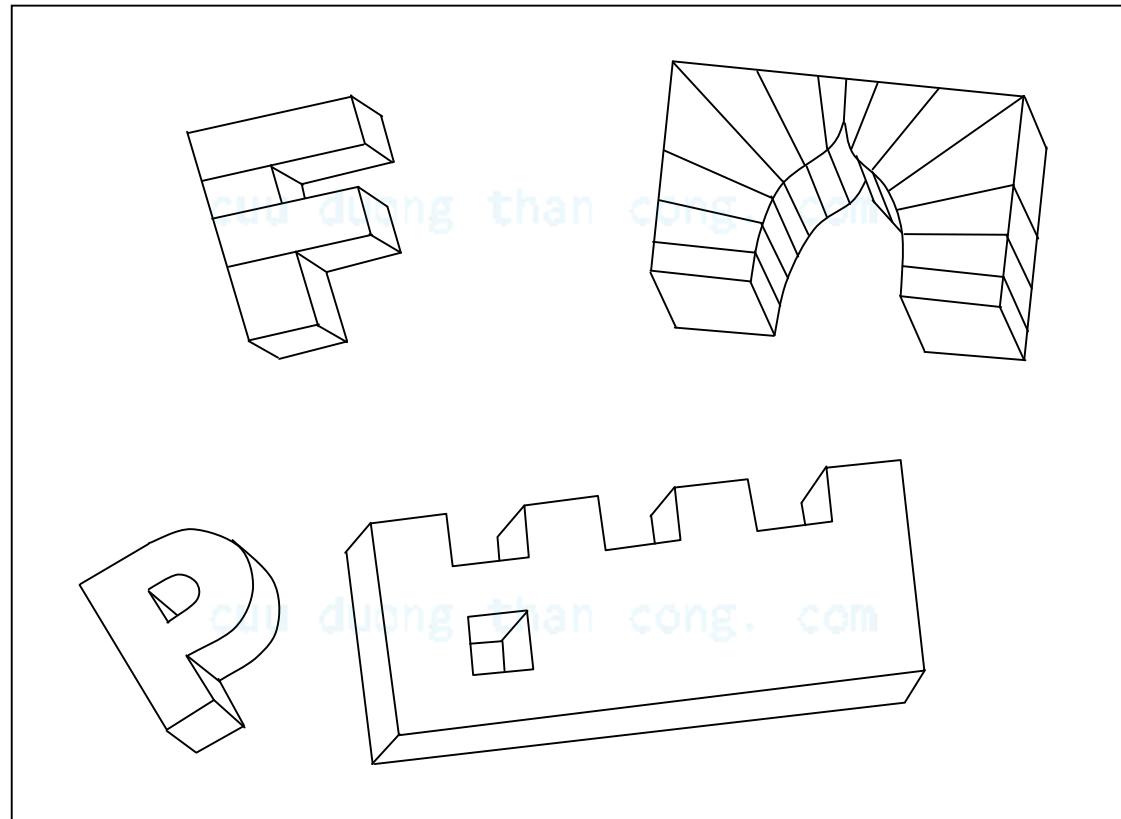
-xây dựng ds đỉnh

-xây dựng ds mặt

-pháp tuyến mặt được tính bằng phương pháp Newell

KHỐI QUÉT

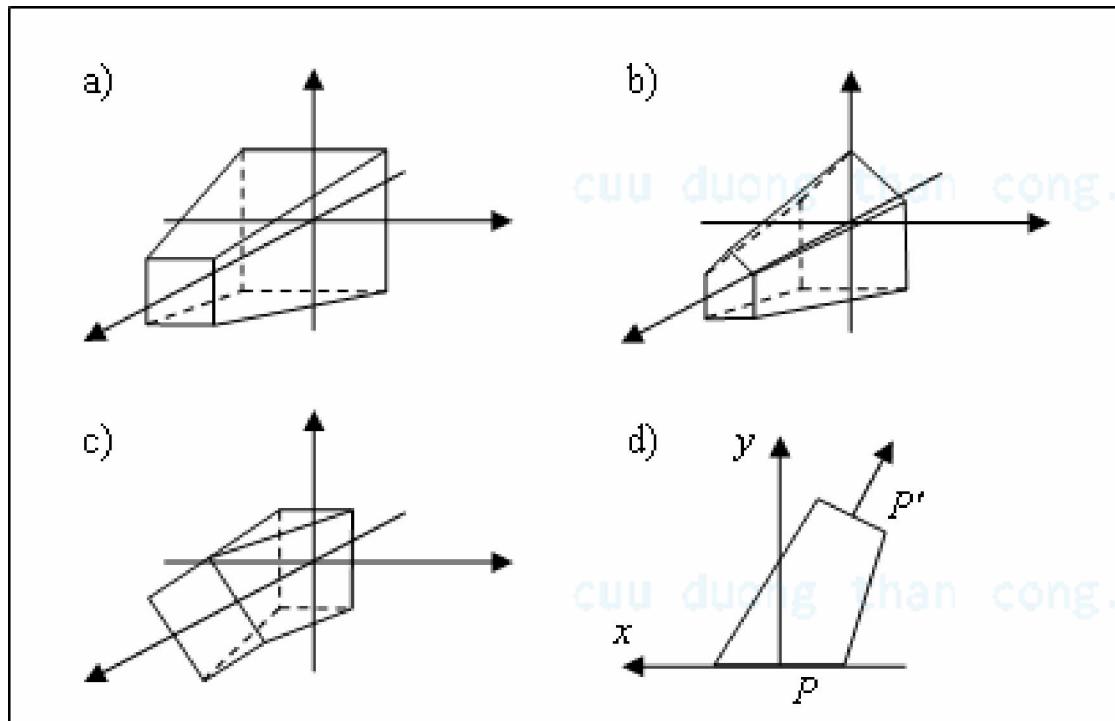
- ☐ Mảng các hình lăng trụ: một số thư viện đồ họa không vẽ được đa giác lõm → khi đa giác là lõm thì cần phải tách nó thành tập đa giác lồi



KHỐI QUÉT

□ Quét kết hợp với xoắn: $P = \{p_0, p_1, \dots, p_{N-1}\}$ (đáy)

$P' = \{Mp_0, Mp_1, \dots, Mp_{N-1}\}$ (đỉnh)



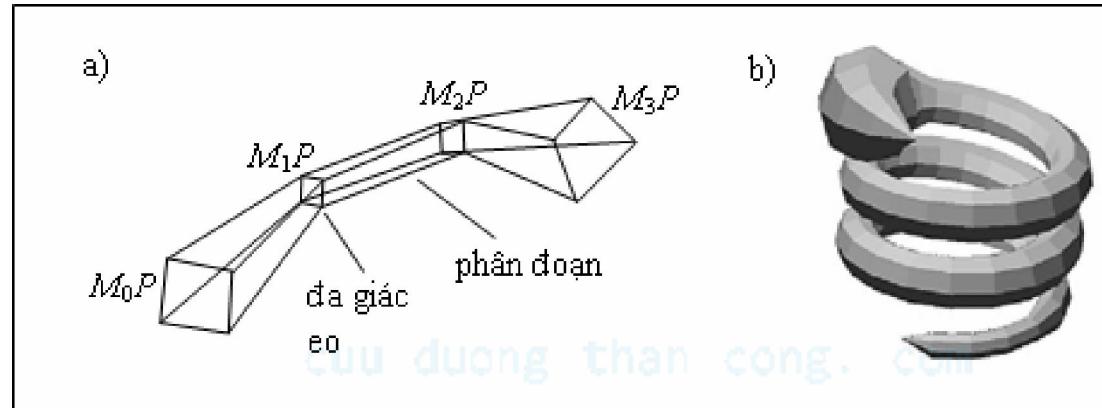
$$M = \begin{pmatrix} 0.7 & 0 & 0 & 0 \\ 0 & 0.7 & 0 & 0 \\ 0 & 0 & 1 & H \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & H \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

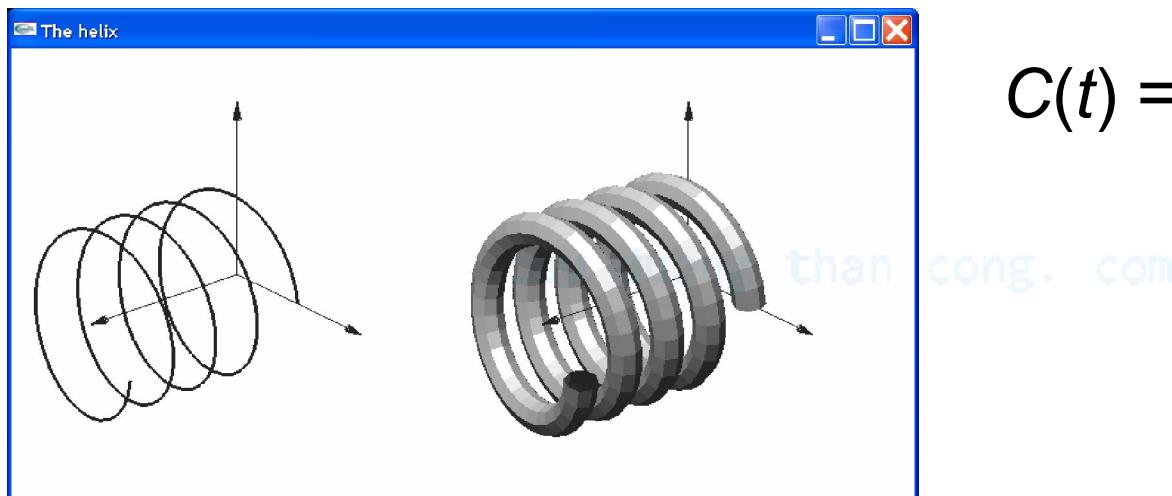
ds mặt giữ nguyên,
đỉnh và pháp tuyến
thay đổi

KHỐI QUÉT

- ❑ Xây dựng khối quét phân đoạn: gồm nhiều đoạn, mỗi đoạn có ma trận biến đổi riêng.



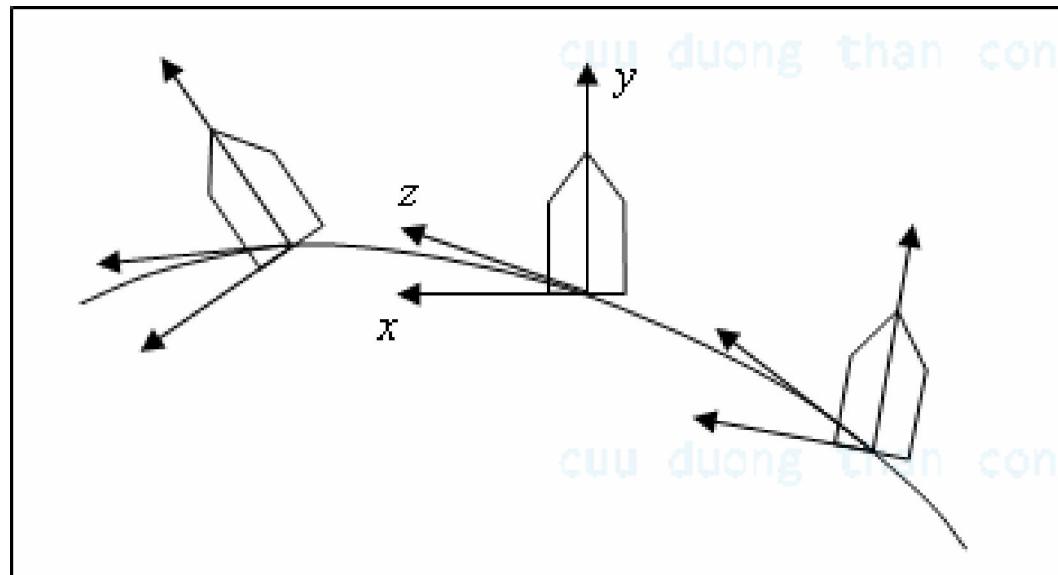
- ❑ Thiết kế hình ống dựa trên đường cong 3D



$$C(t) = (\cos(t), \sin(t), bt)$$

KHỐI QUÉT

- ❑ Lấy mẫu $C(t)$ ở $\{t_0, t_1, \dots\}$, xây dựng đa giác eo nằm trong mặt phẳng vuông góc với đường cong ở $C(t_i)$. Xây dựng hệ trục tọa độ sao cho trục z tiếp tuyến với đường cong
- ❑ Dùng $C(t)$ để xác định hệ tọa độ cục bộ



- $T(t_i)$ tiếp tuyến đc,
- $N(t_i)$, $B(t_i)$ vuông góc với $T(t_i)$
- Ma trận M_i biến đổi đa giác đáy thành đa giác eo
- $M_i = (\mathbf{N}(t_i) | \mathbf{B}(t_i) | \mathbf{T}(t_i) | C(t_i))$

KHỐI QUÉT

☐ Tạo khung Frenet

$$\mathbf{C}'(t) = (\mathbf{C}'_x(t), \mathbf{C}'_y(t), \mathbf{C}'_z(t))$$

$\mathbf{T}(t)$ là $\mathbf{C}'(t)$ sau khi chuẩn hóa

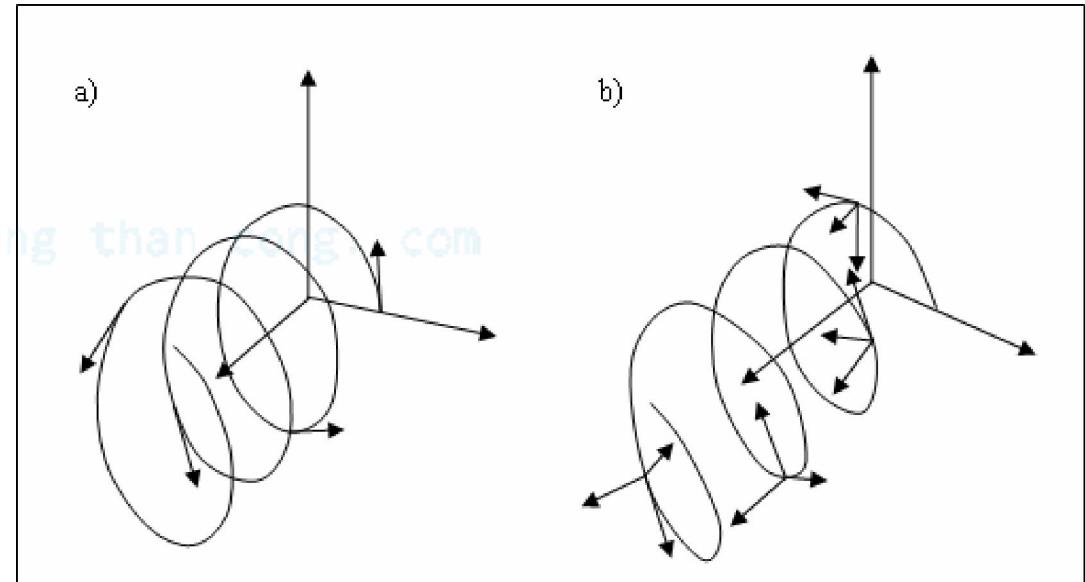
$$\mathbf{B}(t) = \frac{\mathbf{C}'(t) \times \mathbf{C}''(t)}{|\mathbf{C}'(t) \times \mathbf{C}''(t)|}$$

$$\mathbf{N}(t) = \mathbf{B}(t) \times \mathbf{T}(t)$$

$$\mathbf{T}(t) = \frac{1}{\sqrt{1+b^2}}(-\sin(t), \cos(t), b)$$

$$\mathbf{B}(t) = \frac{1}{\sqrt{1+b^2}}(b \sin(t), -b \cos(t), 1)$$

$$\mathbf{N}(t) = (-\cos(t), -\sin(t), 0)$$



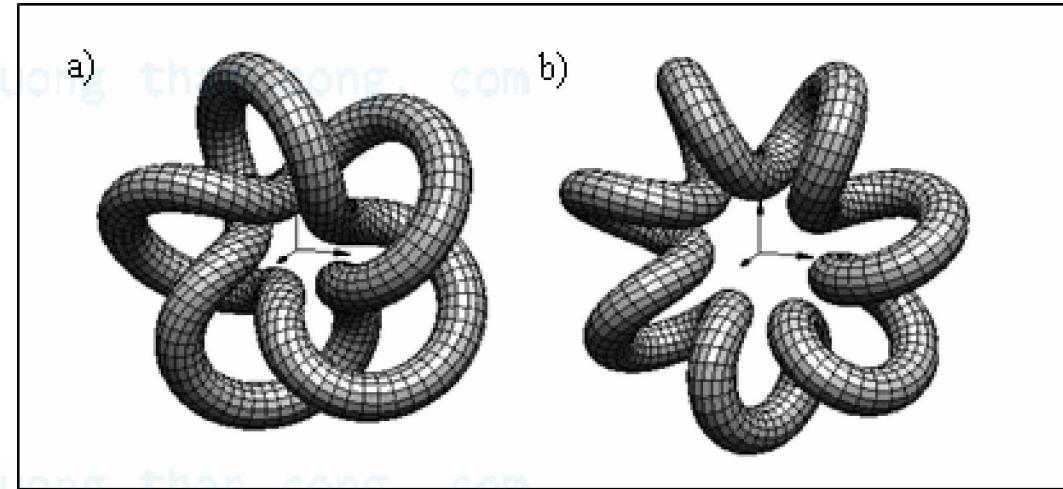
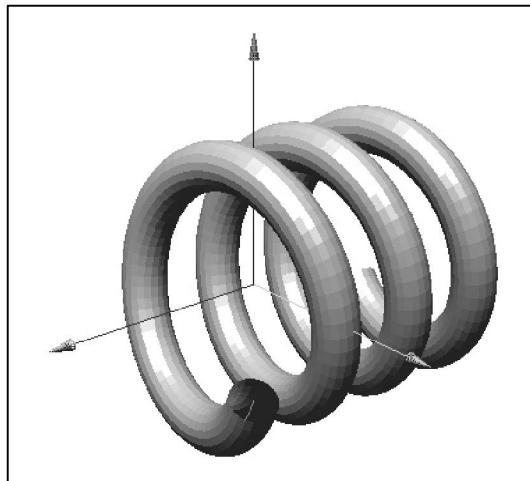
KHỐI QUÉT

- Tạo khung Frenet bằng phương pháp tính

$$C'(t) \approx \frac{C(t + \varepsilon) - C(t - \varepsilon)}{2\varepsilon}$$

$$C''(t) \approx \frac{C(t - \varepsilon) - 2C(t) + C(t + \varepsilon)}{\varepsilon^2}$$

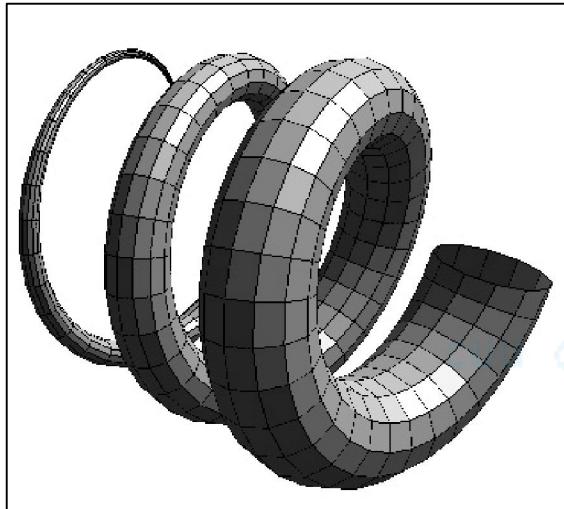
- VÍ DỤ



$$\begin{aligned} C(t) = & ((a + b\cos(qt))\cos(pt), \\ & (a + b\cos(qt))\sin(pt), c\sin(qt)) \end{aligned}$$

KHỐI QUÉT

❑ VÍ DỤ

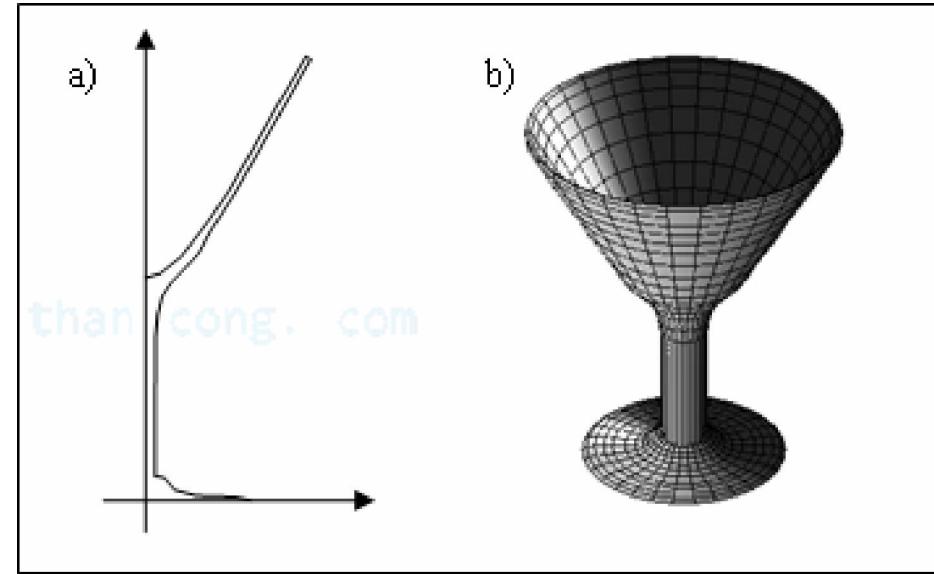
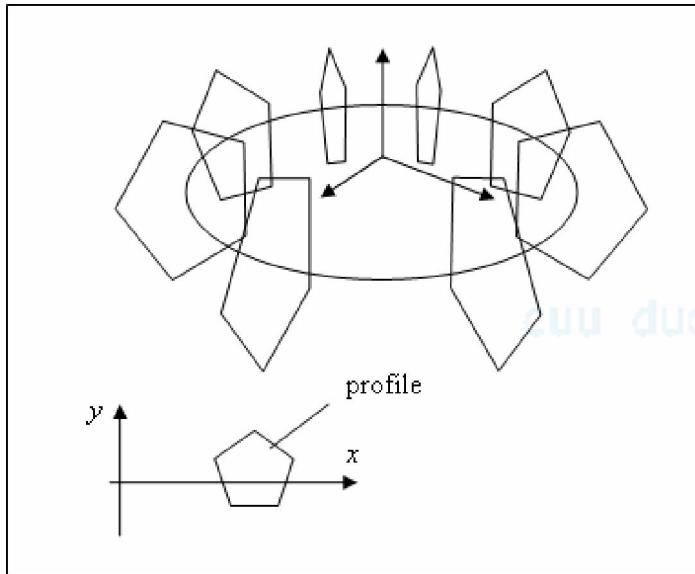


$$M' = M \begin{pmatrix} g(t) & 0 & 0 & 0 \\ 0 & g(t) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

cuu duong than cong. com

KHỐI QUÉT

- Dùng khối quét **xấp xỉ** mặt tròn xoay: phép biến đổi affine là phép quay, đc xương sống ở gốc tọa độ



$$M_i = \begin{pmatrix} \cos(\theta_i) & 0 & \sin(\theta_i) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_i) & 0 & \cos(\theta_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

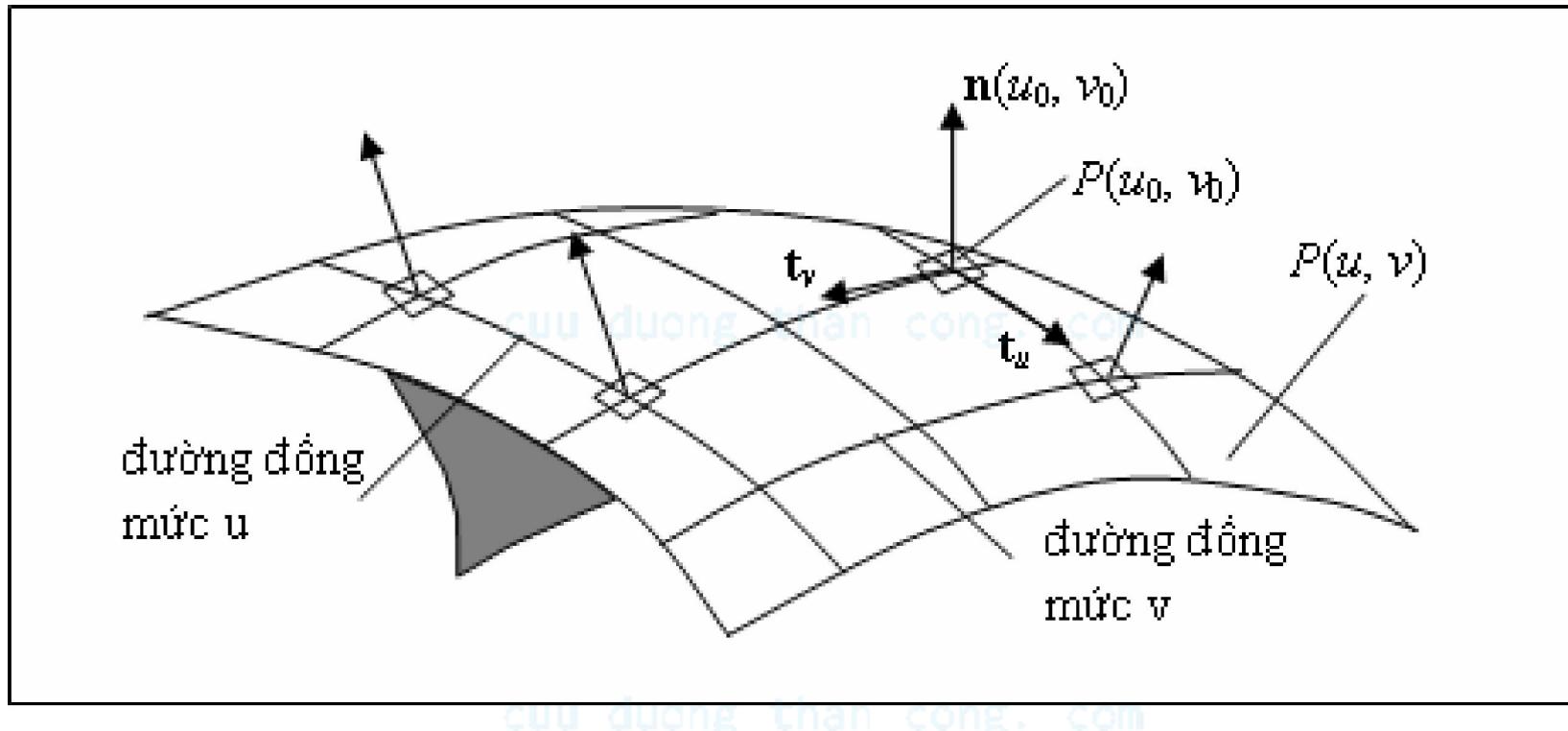
các điểm nằm trên đường
gấp khúc thứ i là
 $(x_j \cos(\theta_i), y_j, x_j \sin(\theta_i))$

LƯỚI XẤP XỈ MẶT CONG

- Phương pháp chung: “đa giác hóa” mặt cong, đỉnh và pháp tuyến được tính từ công thức của mặt cong.
- Biểu diễn mặt cong:
 - dạng tham số: $P(u, v) = (X(u, v), Y(u, v), Z(u, v))$,
 - dạng ẩn: $F(x, y, z) = 0$
 - hàm trong ngoài: điểm (x, y, z)
 - nằm trên mặt cong nếu $F(x, y, z) = 0$
 - nằm bên trong nếu $F(x, y, z) < 0$
 - nằm bên ngoài nếu $F(x, y, z) > 0$

LƯỚI XẤP XỈ MẶT CONG

□ Tìm vector pháp tuyến



LƯỚI XẤP XỈ MẶT CONG

- ☐ Mặt cong được biểu diễn dưới dạng tham số

$$n(u_0, v_0) = \left(\frac{\partial p}{\partial u} \times \frac{\partial p}{\partial v} \right) \Big|_{u=u_0, v=v_0}$$

(n có thể được chuẩn hóa nếu muốn)

Ví dụ: mặt phẳng $P(u, v) = C + \mathbf{a}u + \mathbf{b}v$

$$\mathbf{n}(u, v) = \mathbf{a} \times \mathbf{b}$$

cuu duong than cong. com

LƯỚI XẤP XỈ MẶT CÔNG

- ☐ Mặt cong được biểu diễn dưới dạng ẩn

$$\mathbf{n}(x_0, y_0, z_0) = \nabla F \Big|_{x=x_0, y=y_0, z=z_0} = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z} \right) \Big|_{x=x_0, y=y_0, z=z_0}$$

Ví dụ: dạng ẩn của mặt phẳng là:

$$F(x, y, z) = \mathbf{n} \bullet ((x, y, z) - A) = 0$$

hay: $n_x x + n_y y + n_z z - \mathbf{n} \bullet A = 0$

→ pháp tuyến là (n_x, n_y, n_z)

CÁC MẶT CƠ SỞ

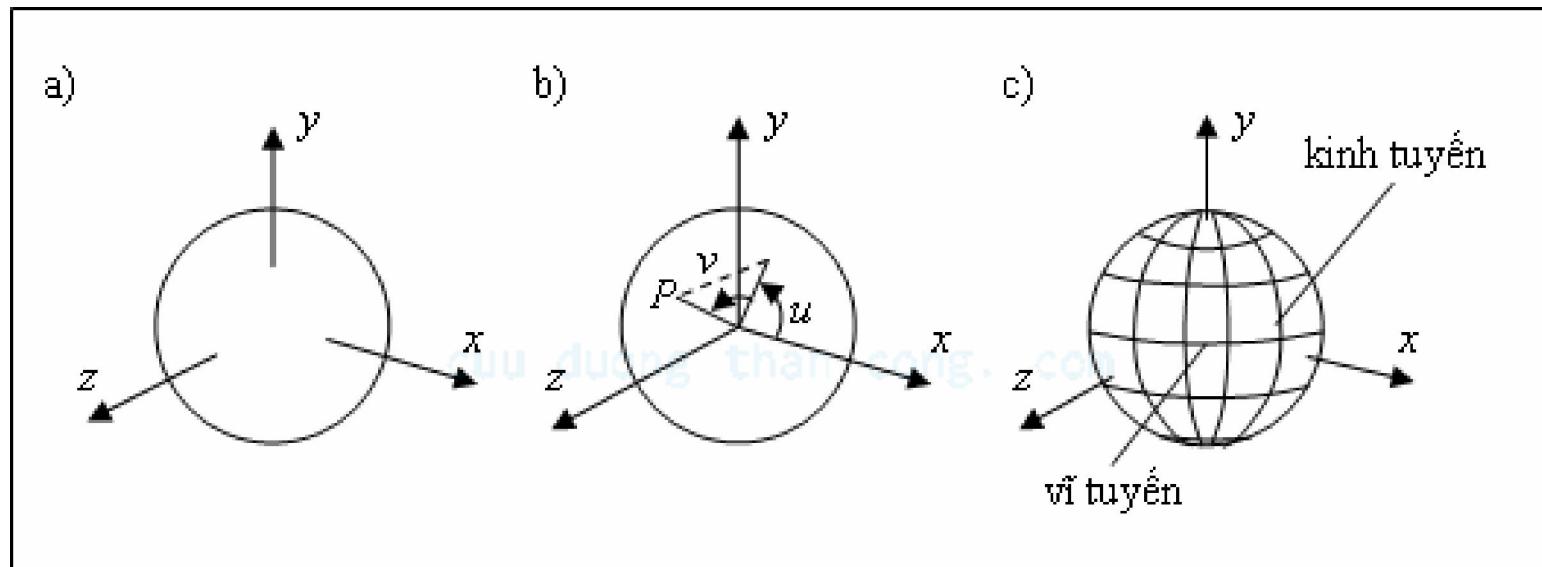
☐ Mặt cầu cơ sở:

✓ dạng ẩn $F(x, y, z) = x^2 + y^2 + z^2 - 1$

→ pháp tuyến $(2x, 2y, 2z)$

✓ dạng tham số $P(u, v) = (\cos(v)\cos(u), \cos(v)\sin(u), \sin(v))$

→ pháp tuyến $\mathbf{n}(u, v) = -\cos(u, v)\mathbf{p}(u, v)$



CÁC MẶT CƠ SỞ

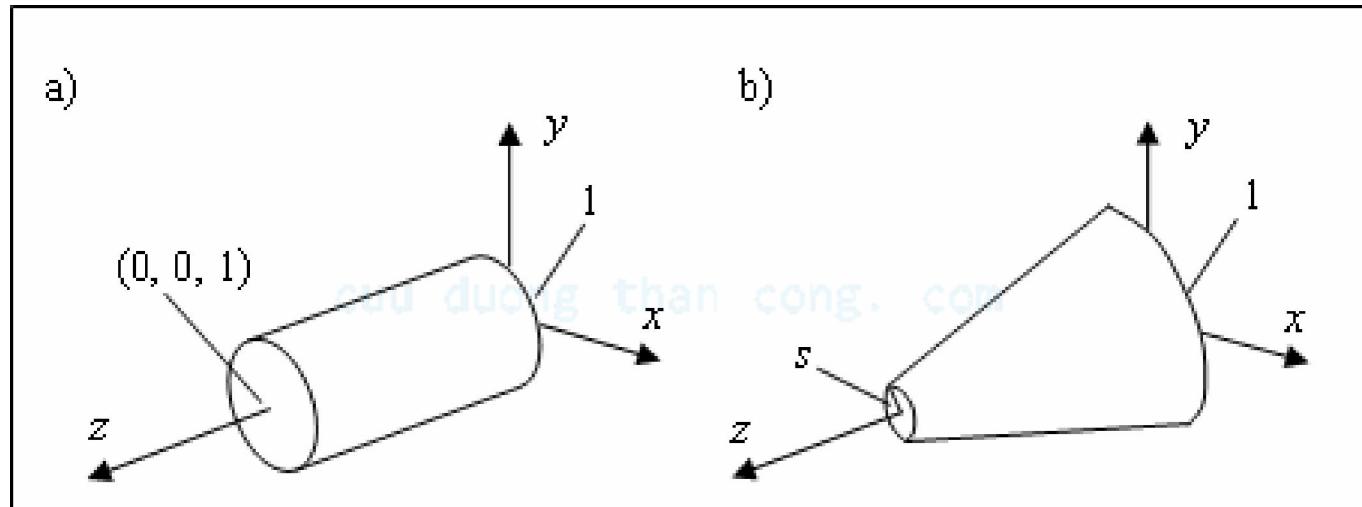
□ Mặt trụ cơ sở:

- dạng ẩn $F(x, y, z) = x^2 + y^2 - (1 + (s - 1)z)^2 \quad 0 < z < 1 \rightarrow$
 $\mathbf{n}(x, y, z) = (x, y, -(s - 1)(1 + (s - 1)z))$
- dạng tham số

$$P(u, v) = ((1 + (s - 1)v)\cos(u), (1 + (s - 1)v)\sin(u), v)$$

cuuduongthancong.com

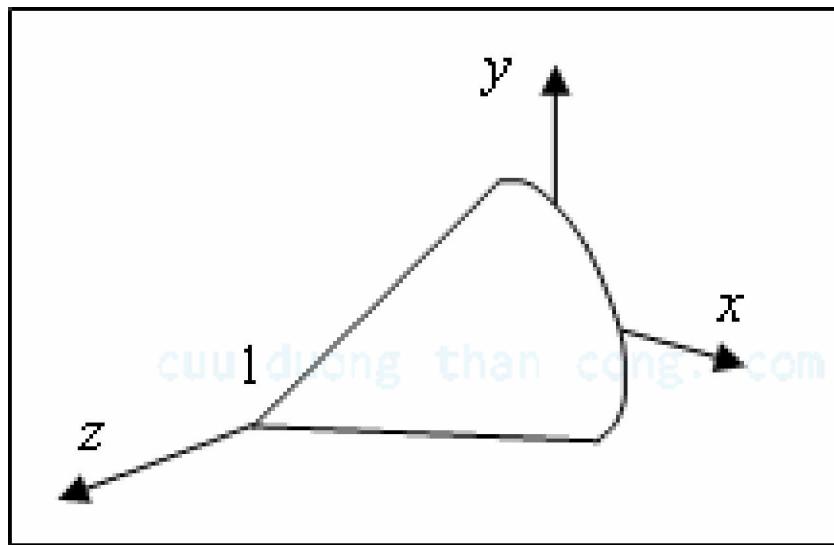
$$\rightarrow \mathbf{n}(u, v) = (\cos(u), \sin(u), 1-s)$$



CÁC MẶT CƠ SỞ

□ Mặt nón cơ sở

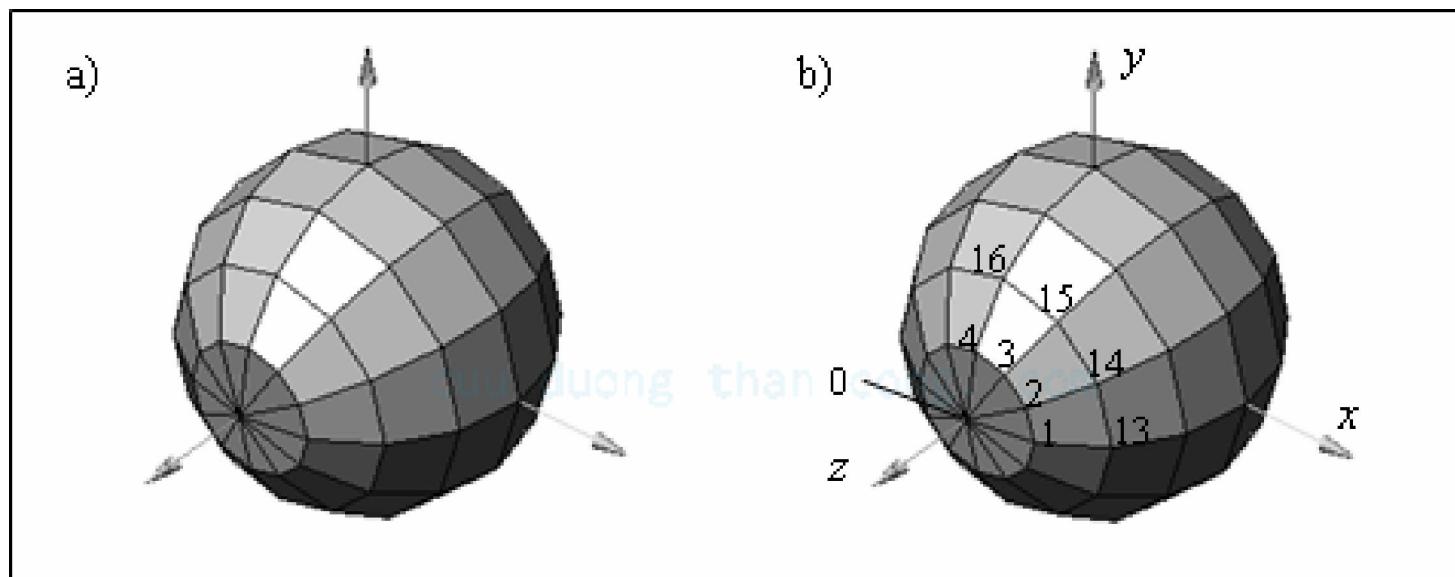
- dạng ẩn: $F(x, y, z) = x^2 + y^2 - (1 - z)^2 = 0 \quad 0 < z < 1$
 $\rightarrow n = (x, y, 1 - z)$
- dạng tham số $P(u, v) = ((1 - v) \cos(u), (1 - v)\sin(u), v)$
 $\rightarrow n(u, v) = (\cos(u), \sin(u), 1)$



XÂY DỰNG LƯỚI ĐA GIÁC

□ Xây dựng lưới đa giác cho hình cầu:

- chia thành nhiều lát(stack) và múi(slice)
- $u_i = 2\pi i/nSlices$ với $i = 0, 1, \dots, nSlices - 1$
- $v_j = \pi/2 - \pi j/nStacks$, với $j = 0, 1, \dots, nStacks$
- pháp tuyến norm[k] chính là pt[k]



MẶT CHÚA CẠNH THẲNG

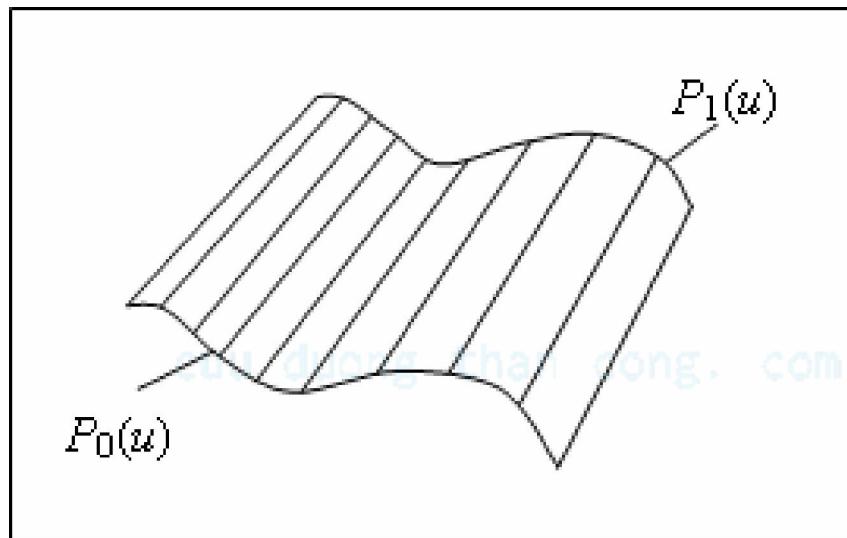
☐ Tồn tại ít nhất một đt thuộc mặt đi qua mỗi điểm

☐ Dạng biểu diễn tham số

$$P(u, v) = (1-v)P_0(u) + vP_1(u) \leftarrow P(u, v) = (1-v)P_0 + vP_1$$

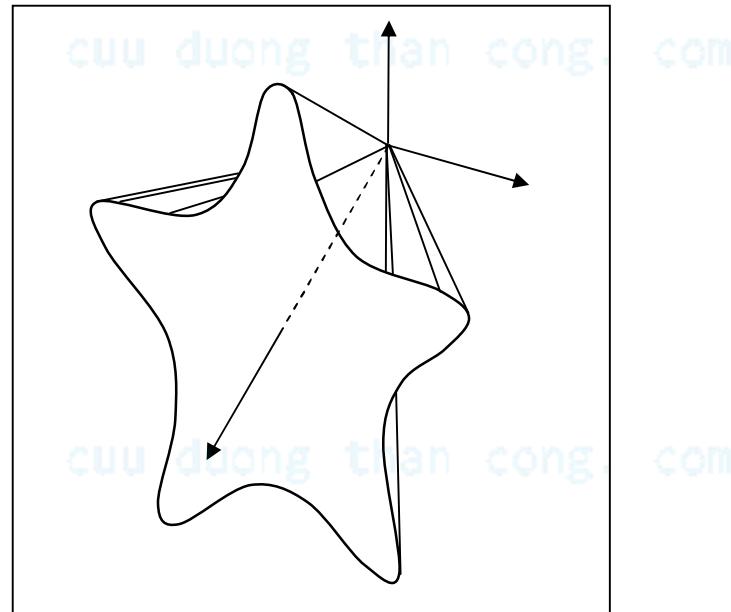
☐ Mỗi hàm cho bởi 3 thành phần

$$P_0(u) = (X_0(u), Y_0(u), Z_0(u))$$



MẶT CHÚA CẠNH THĂNG

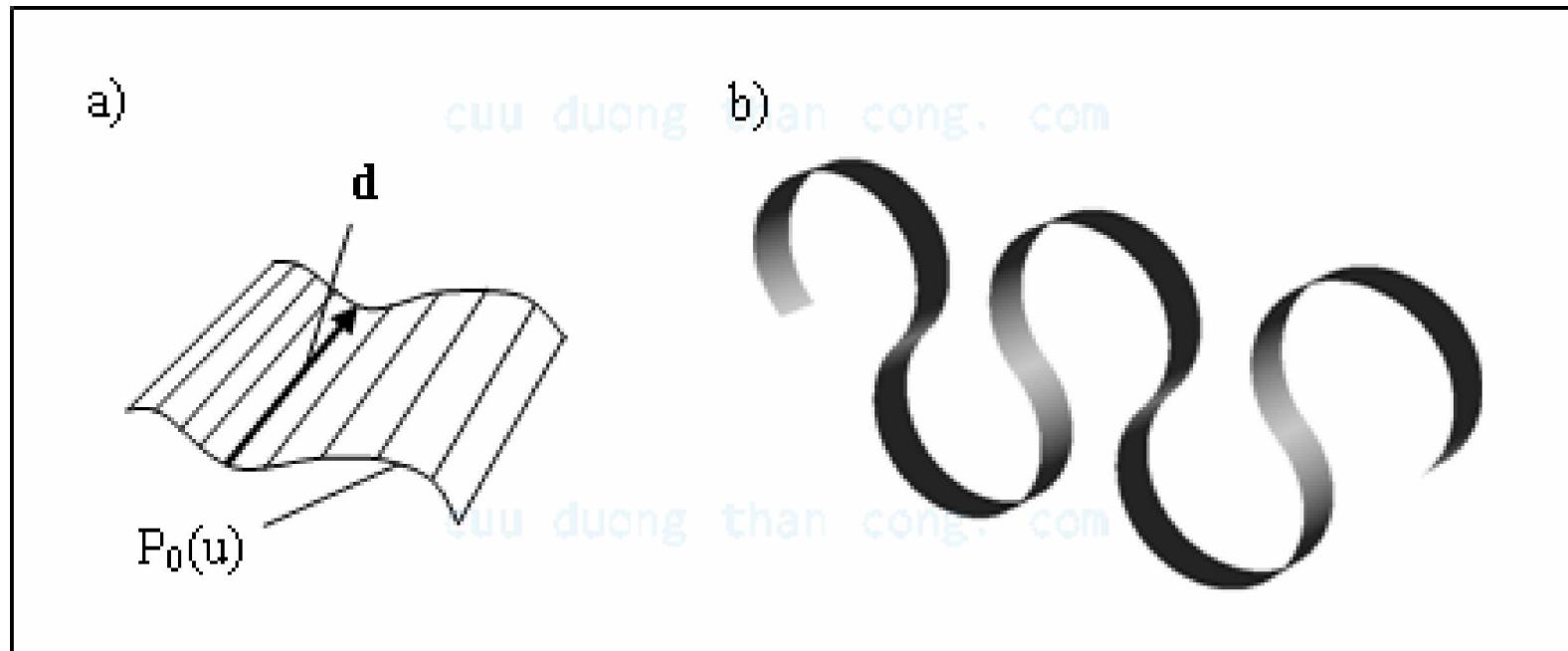
- **Mặt nón:** $P(u, v) = (1-v)P_0 + vP_1(u)$
- P_0 là đỉnh của mặt nón
- Ví dụ: $P_1(u) = (r(u)\cos(u), r(u)\sin(u), 1)$
 $r(u) = 0.5 + 0.2\cos(5u)$



MẶT CHỦA CẠNH THĂNG

□ Mặt trục:

- $P_1(u) = P_0(u) + \mathbf{d}$
- biểu diễn tham số: $P(u, v) = P_0(u) + \mathbf{d}v$

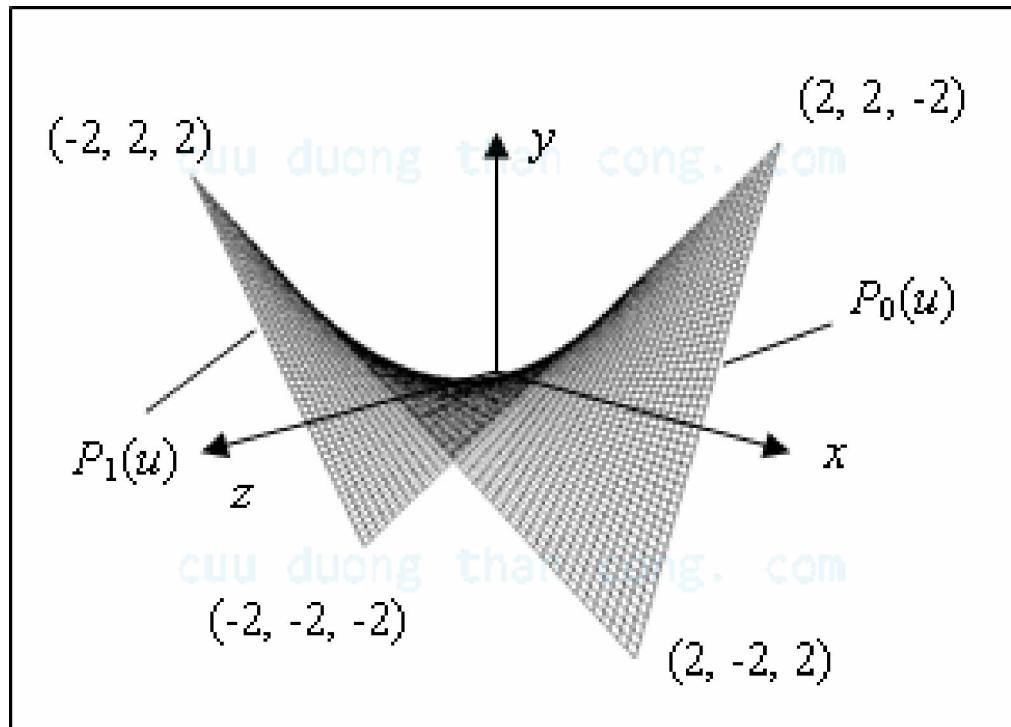


MẶT CHỮA CẠNH THĂNG

□ Mảnh tuyến tính đôi

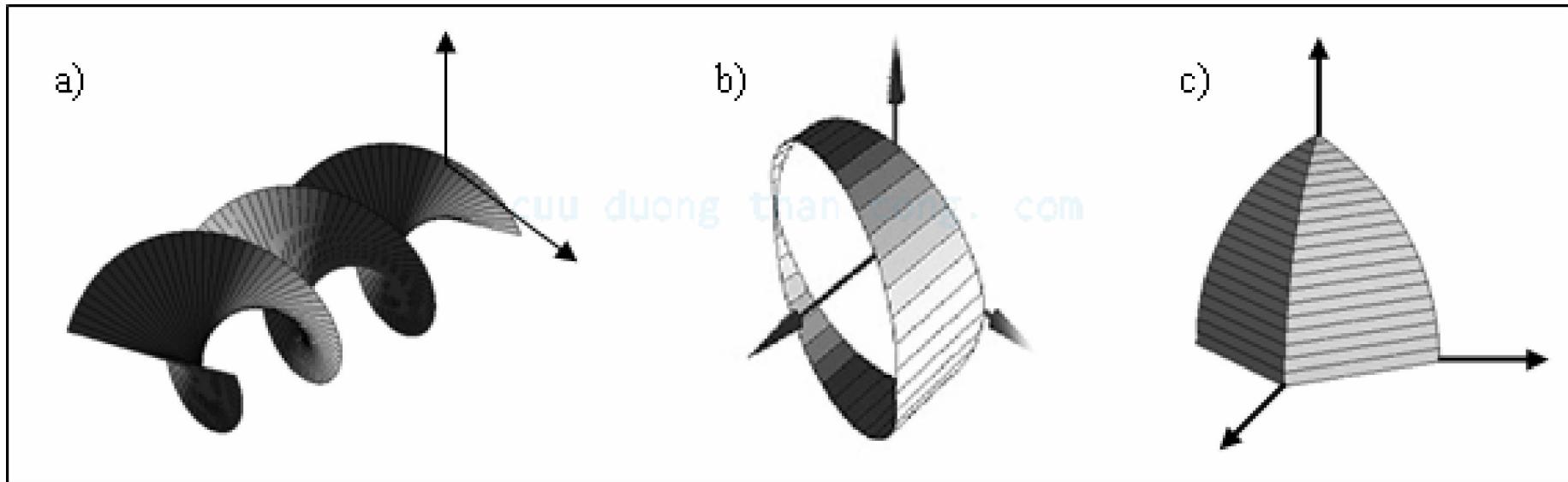
$$- P_0(u) = (1 - u)P_{00} + uP_{01}$$

$$- P(u, v) = (1 - v)(1 - u)P_{00} + (1 - v)uP_{01} + v(1 - u)P_{10} + uvP_{11}$$



MẶT CHÚA CẠNH THẲNG

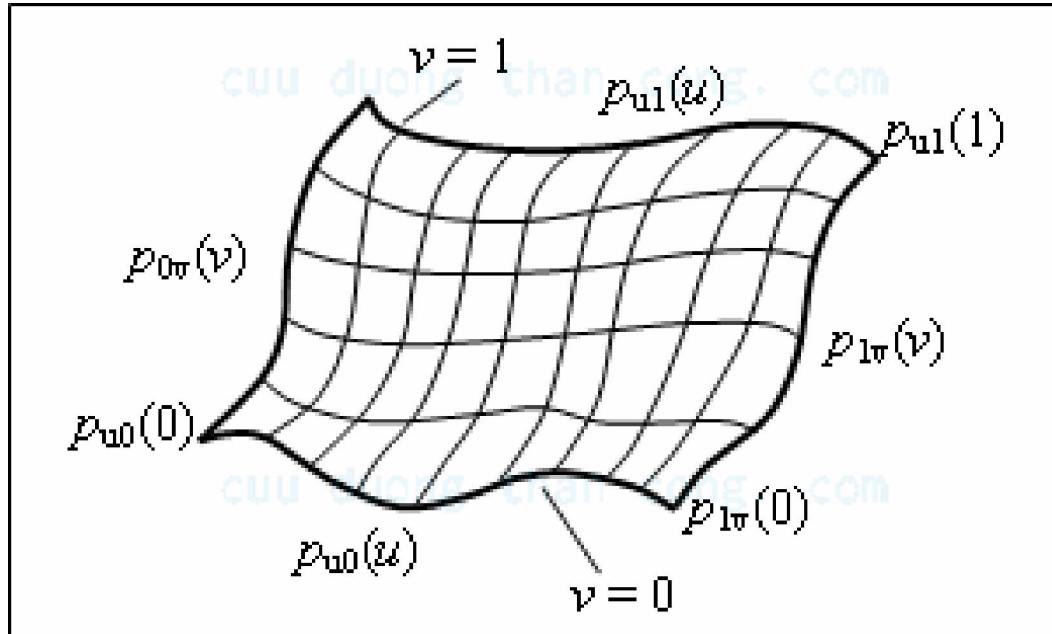
- ☐ Một số mặt chứa cạnh thẳng khác



MẶT CHỨA CẠNH THẲNG

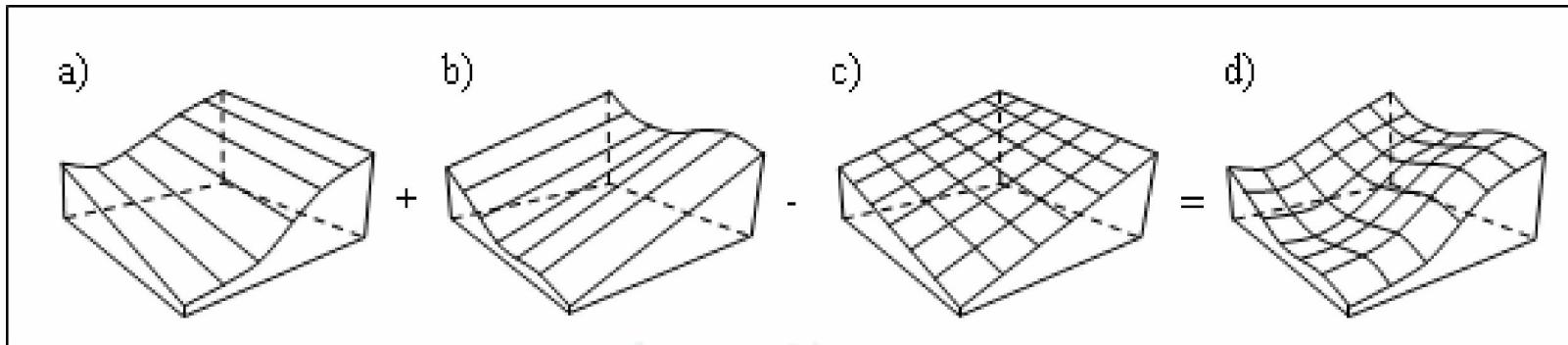
☐ Mảnh trộn tuyến tính đối: mảnh Coons

- tổng quát hóa của mặt chứa cạnh thẳng là mặt được tạo bởi 4 đường cong biên $p_{u0}(u)$, $p_{u1}(u)$, $p_{0v}(v)$ và $p_{1v}(v)$, với u và v nhận giá trị trong đoạn $[0, 1]$



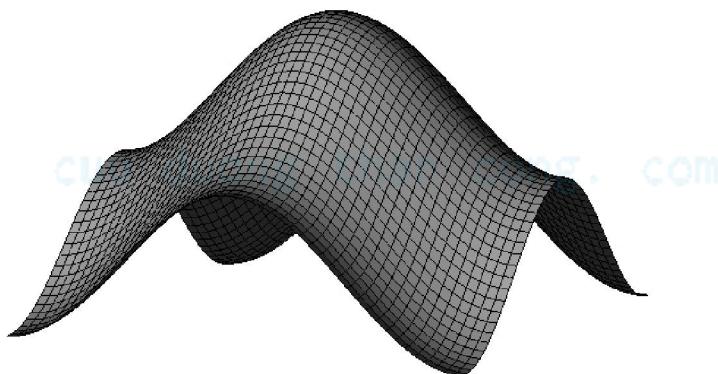
MẶT CHỮA CẠNH THẲNG

☐ Mảnh trộn tuyến tính đôi: cách xây dựng



Cuu_duong_thanCong.com

Công thức: $P(u, v) = [p_{0v}(v)(1 - u) + p_{1v}(v)u] + [p_{u0}(u)(1 - v) + p_{u1}(u)v] - [(1 - u)(1 - v)p_{0v}(0) + u(1 - v)p_{1v}(0) + v(1 - u)p_{0v}(1) + uv p_{1v}(1)]$



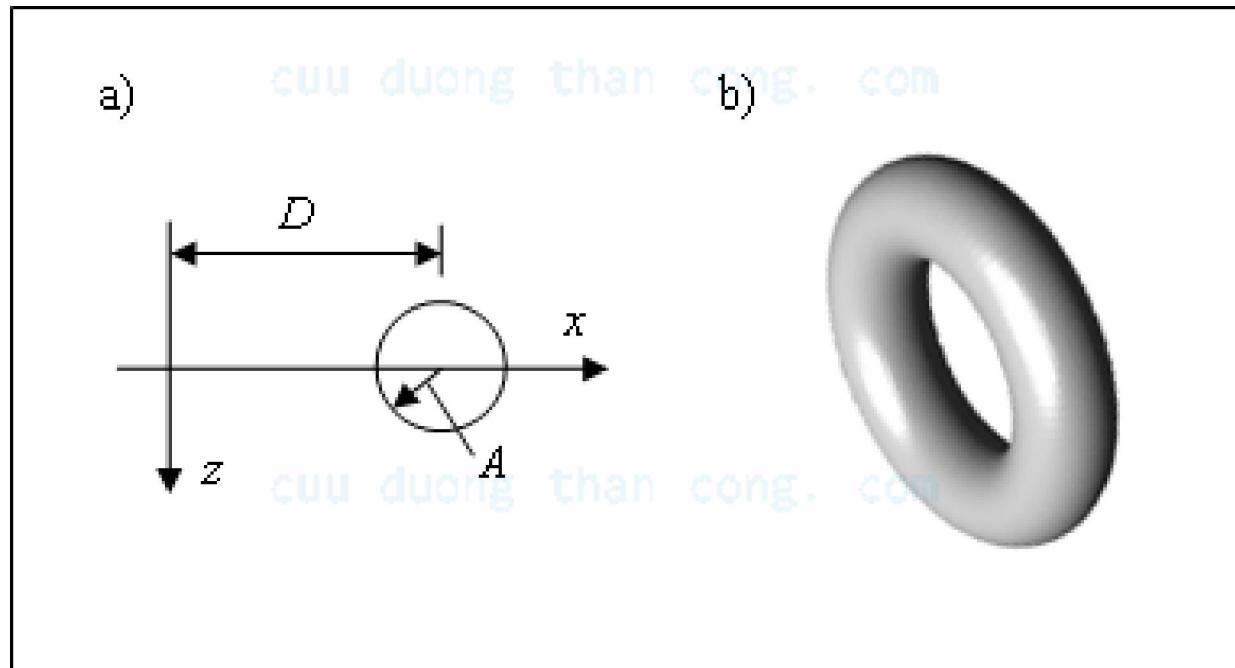
MẶT TRÒN XOAY

- ❑ Mặt tròn xoay được tạo bởi quét đc phẳng C (profile) xung quanh một trục
- ❑ Profile có phương trình $C(v) = (X(v), Z(v))$ khi quay quanh trục z góc u $\rightarrow (X(v)\cos(u), X(v)\sin(u), Z(v))$
- ❑ Kinh tuyến: đc profile ở một thời điểm nào đó. Vĩ tuyến: một điểm bất kỳ thuộc đường cong khi quay một vòng.
- ❑ Phương trình tham số:
$$P(u, v) = (X(v)\cos(u), X(v)\sin(u), Z(v))$$
- ❑ Tiếp tuyến:
$$\mathbf{n}(u, v) = X(v)(Z'(v)\cos(u), Z'(v)\sin(u), -X'(v))$$

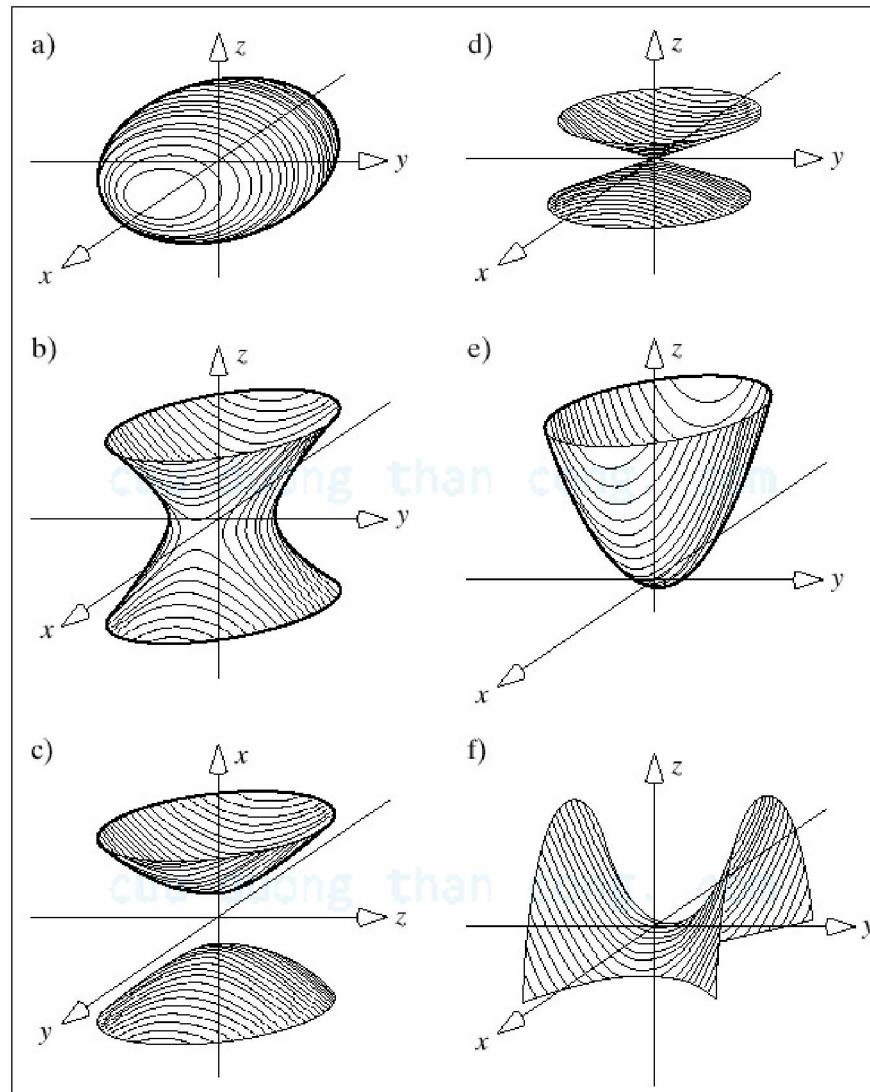
MẶT TRÒN XOAY

❑ Ví dụ:

- đc profile: $C(v) = (D + A\cos(v), A\sin(v))$
- $P(u, v) = ((D + A\cos(v))\cos(u), (D + A\cos(v))\sin(u), A\sin(v))$



MẶT BẬC HAI



MẶT BẬC HAI

Tên mặt cong	Dạng ẩn	Dạng tham số	khoảng biến thiên của v và u
Ellipsoid	$x^2 + y^2 + z^2 - 1$	$(\cos(v)\cos(u), \cos(v)\sin(u), \sin(v))$	$(-\pi/2, \pi/2), (-\pi, \pi)$
Hyperboloid một tầng	$x^2 + y^2 - z^2 - 1$	$(\sec(v)\cos(u), \sec(v)\sin(u), \tan(v))$	$(-\pi/2, \pi/2), (-\pi, \pi)$
Hyperboloid hai tầng	$x^2 - y^2 - z^2 - 1$	$(\sec(v)\cos(u), \sec(v)\tan(u), \tan(v))$	* , $(-\pi/2, \pi/2)$
Elliptic cone	$x^2 + y^2 - z^2$	$(v\cos(u), v\sin(u), v)$	Số thực bất kỳ, $(-\pi, \pi)$
Elliptic paraboloid	$x^2 + y^2 - z$	$(v\cos(u), v\sin(u), v^2)$	$v \geq 0, (-\pi, \pi)$
Hyperbolic paraboloid	$-x^2 + y^2 - z$	$(v\cos(u), v\sec(u), v^2)$	$v \geq 0, (-\pi, \pi)$

* khoảng biến thiên của v cho tầng #1 là $(-\pi/2, \pi/2)$ và tầng #2 là $(\pi/2, 3\pi/2)$

cuuduongthancong.com

MẶT BẬC HAI

- Lưu ý: - vết cắt: là giao của mặt phẳng với mặt cong.
 - vết cắt chính: khi mặt phẳng song song với mặt phẳng chính.
 - vết cắt là thiết diện nón
- Ellipsoid:
 - hai tham số bằng nhau → mặt tròn xoay
 - ba tham số bằng nhau → mặt cầu
 - vết cắt là ellipse
- Hyperboloid một tầng
 - $a = b$ → mặt tròn xoay
 - vết cắt chính $z = k$ là ellipse, hai vết cắt chính kia là hyperbola
 - là mặt chứa cạnh thẳng

MẶT BẬC HAI

❑ Hyperboloid hai tầng :

- $a = b \rightarrow$ mặt tròn xoay
- vết cắt chính $x = k$ là ellipse, hai vết cắt chính kia là hyperbola

❑ Elliptic Cone :

- $a = b \rightarrow$ mặt tròn xoay
- mặt nón có đường sinh tựa trên ellipse
- là mặt chứa cạnh thẳng, vết cắt chính $z = k$ là ellipse

❑ Elliptic Paraboloid :

- vết cắt chính $z = k$ là ellipse
- $a = b \rightarrow$ mặt tròn xoay

MẶT BẬC HAI

❑ Hyperbolic Paraboloid :

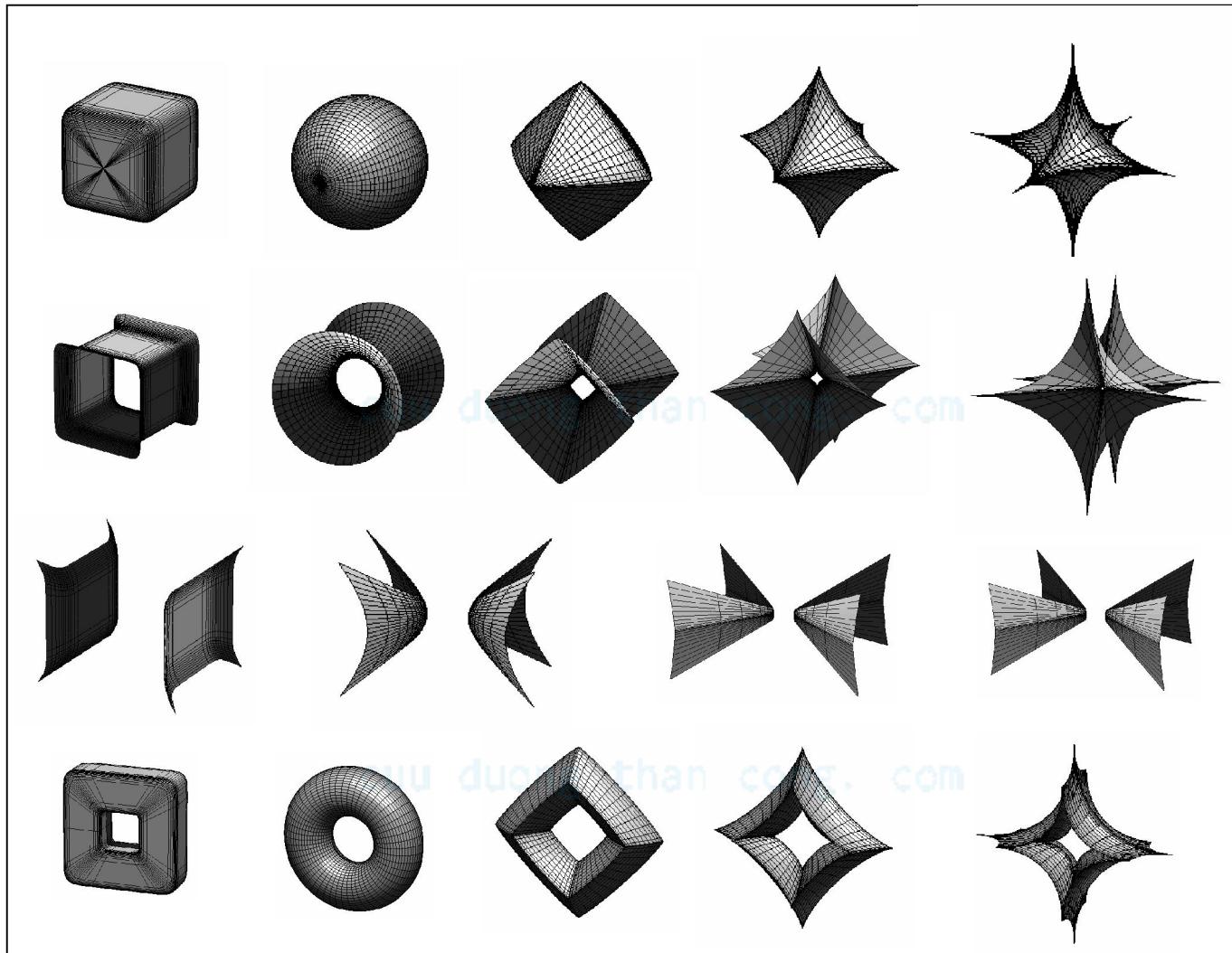
- mặt yên ngựa
- là mặt chứa cạnh thẳng
- vết cắt với $z = k$ là hyperbola, hai vết cắt kia là parabol

❑ Pháp tuyến của mặt bậc hai:

Ví dụ: pháp tuyến của ellipsoid

- tính theo hàm ẩn $F' = (2x, 2y, 2z)$
- thay dạng biểu diễn ẩn của x, y và z
 $n(u, v) = (\cos(v)\cos(u), \cos(v)\sin(u), \sin(v))$

MẶT SIÊU BẬC HAI



MẶT SIÊU BẬC HAI

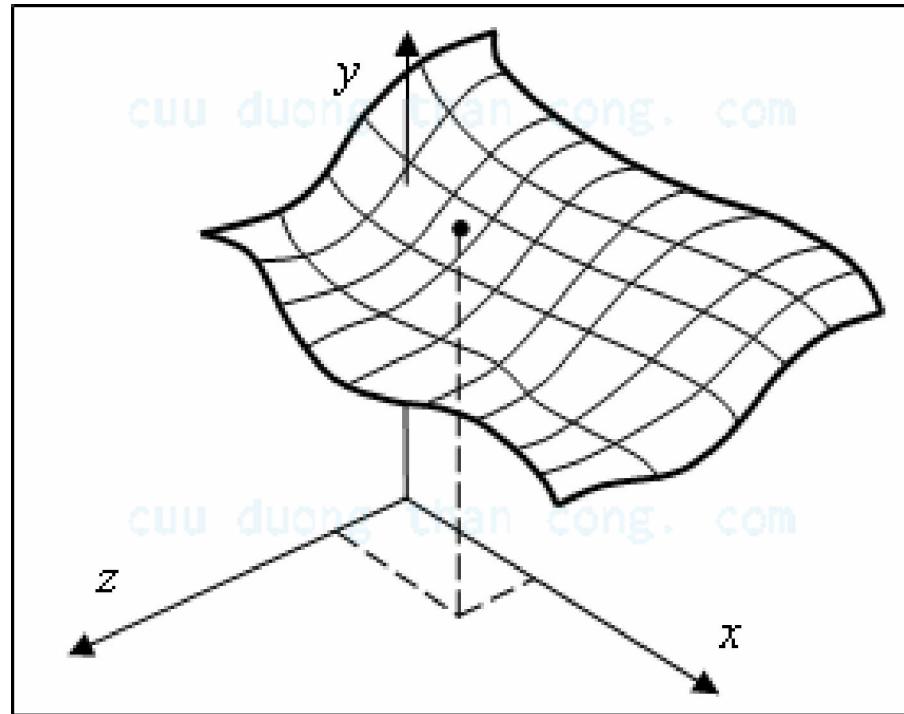
Tên mặt cong	Dạng ẩn	Dạng tham số	Khoảng biến thiên của v và u
Superellipsoid	$(x^n + y^m)^{m/n} + z^m - 1$	$(\cos^{2/m}(v)\cos^{2/n}(u), \cos^{2/m}(v)\sin^{2/n}(u), \sin^{2/m}(v))$	$[-\pi/2, \pi/2], [-\pi, \pi]$
Superhyperboloid một tầng	$(x^n + y^m)^{m/n} - z^m - 1$	$(\sec^{2/m}(v)\cos^{2/n}(u), \sec^{2/m}(v)\sin^{2/n}(u), \tan^{2/m}(v))$	$[-\pi/2, \pi/2], [-\pi, \pi]$
Superhyperboloid hai tầng	$(x^n - y^m)^{m/n} - z^m - 1$	$(\sec^{2/m}(v)\sec^{2/n}(u), \sec^{2/m}(v)\tan^{2/n}(u), \tan^{2/m}(v))$	$(-\pi/2, \pi/2)$
Supertoroid	$((x^n - y^m)^{1/m} - d)^m + z^m - 1$	$((d + \cos^{2/m}(v))\cos^{2/n}(u), (d + \cos^{2/m}(v))\sin^{2/n}(u), \sin^{2/m}(v))$	$(-\pi, \pi), [-\pi, \pi)$

MẶT BIỂU DIỄN BỞI HÀM TƯỜNG MINH

- ☐ Hàm đơn trị theo một trục tọa độ nào đó

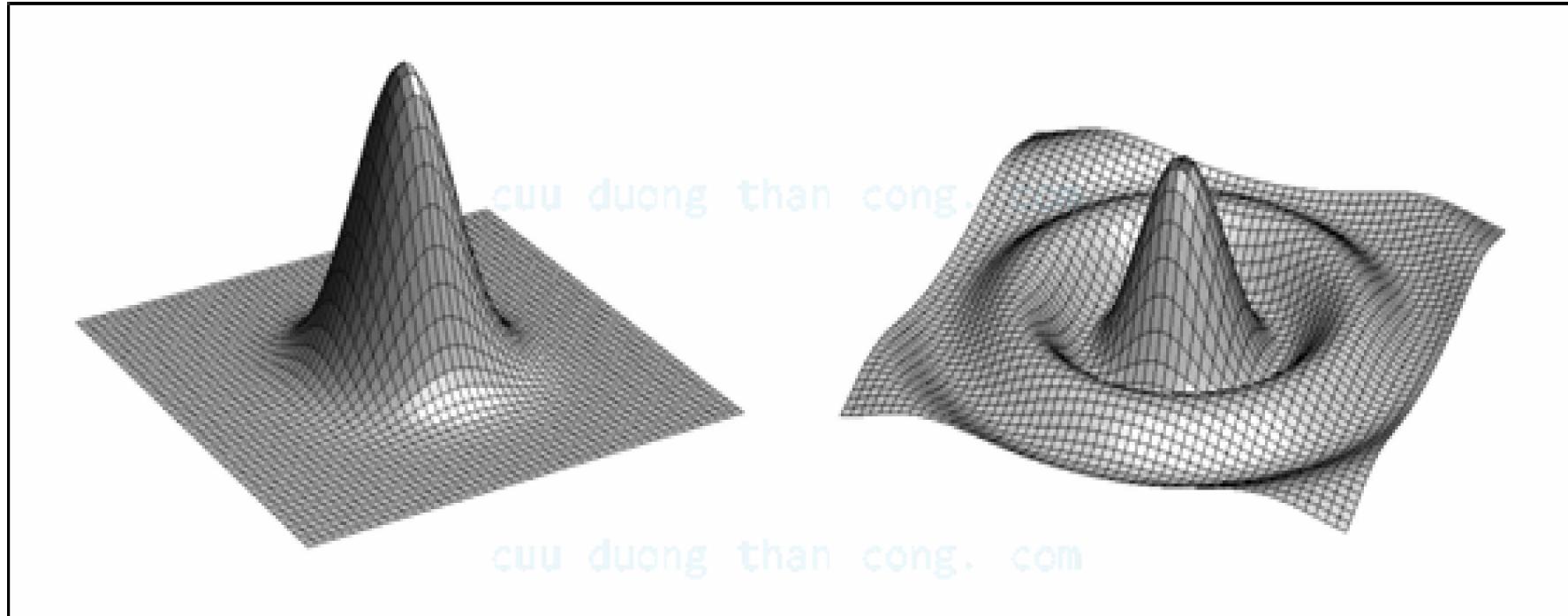
$$f(x, z) = e^{-ax^2 - bz^2}$$

$$f(x, z) = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$$



MẶT BIỂU DIỄN BỞI HÀM TƯỜNG MINH

- Dạng tham số: $P(u, v) = (u, f(u, v), v)$
- Pháp tuyến: $\mathbf{n}(u, v) = (-\partial f / \partial u, 1, -\partial f / \partial v)$



Trường Đại Học Bách Khoa TP Hồ Chí Minh
Khoa Khoa học & Kỹ thuật Máy tính



ĐỒ HỌA MÁY TÍNH

CHƯƠNG 7:

**PHÉP NHÌN TRONG
KHÔNG GIAN 3 CHIỀU**

NỘI DUNG TRÌNH BÀY

- ❑ Ôn tập về camera
- ❑ Xây dựng camera trong chương trình
- ❑ Hình chiếu phối cảnh
- ❑ Hình chiếu phối cảnh của điểm
- ❑ Hình chiếu phối cảnh của đoạn thẳng
- ❑ Độ sâu giả
- ❑ Sử dụng tọa độ đồng nhất
- ❑ Ý nghĩa hình học của phép biến đổi phối cảnh
- ❑ Thể tích nhìn chuẩn
- ❑ Cắt xén với thể tích nhìn

GIỚI THIỆU

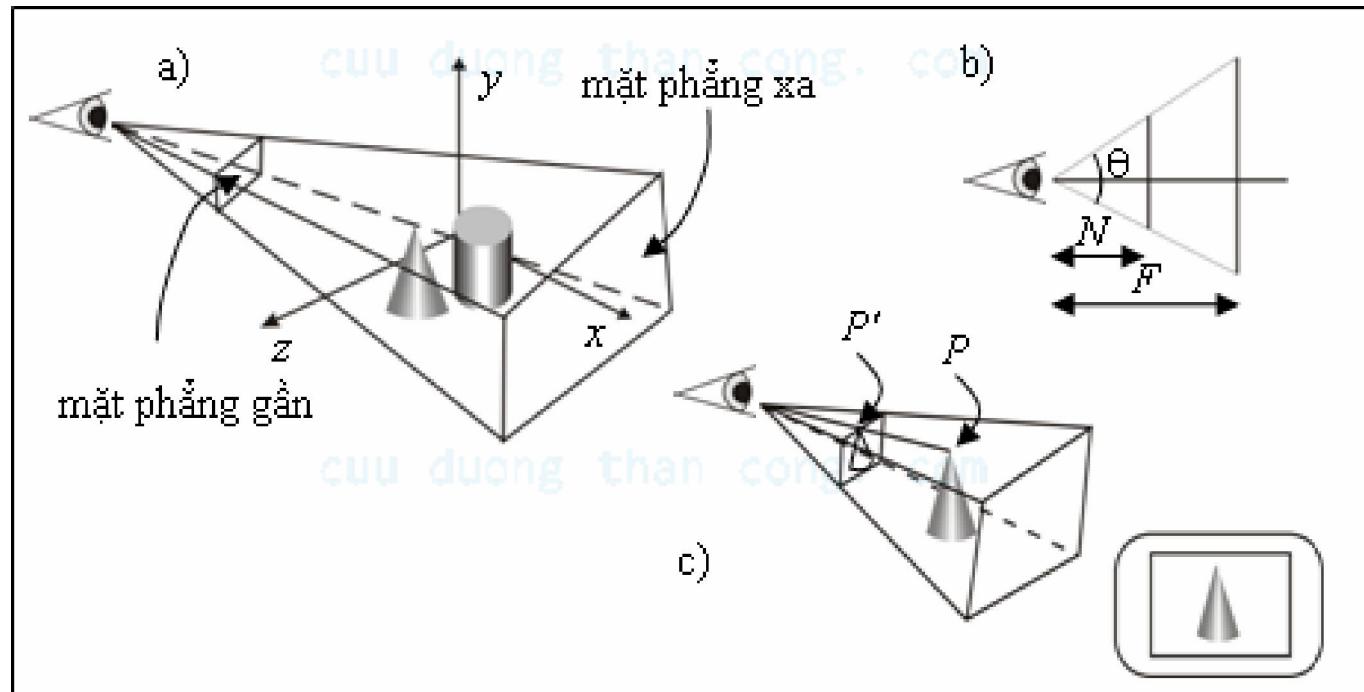
- ❑ Xây dựng và điều khiển camera tạo hình chiếu phổi cảnh
- ❑ Điều khiển vị trí và hướng của camera
- ❑ Điều khiển thể tích nhìn của camera
- ❑ Cắt xén với thể tích nhìn của camera

cuu duong than cong. com

cuu duong than cong. com

ÔN TẬP VỀ CAMERA

- ❑ Thể tích nhìn: một phần của hình chóp có đỉnh là mắt nhìn.
- ❑ Các thành phần: mắt nhìn, góc nhìn (tính bằng độ), mặt phẳng gần, mặt phẳng xa, mặt phẳng nhìn (thường là mặt phẳng gần)



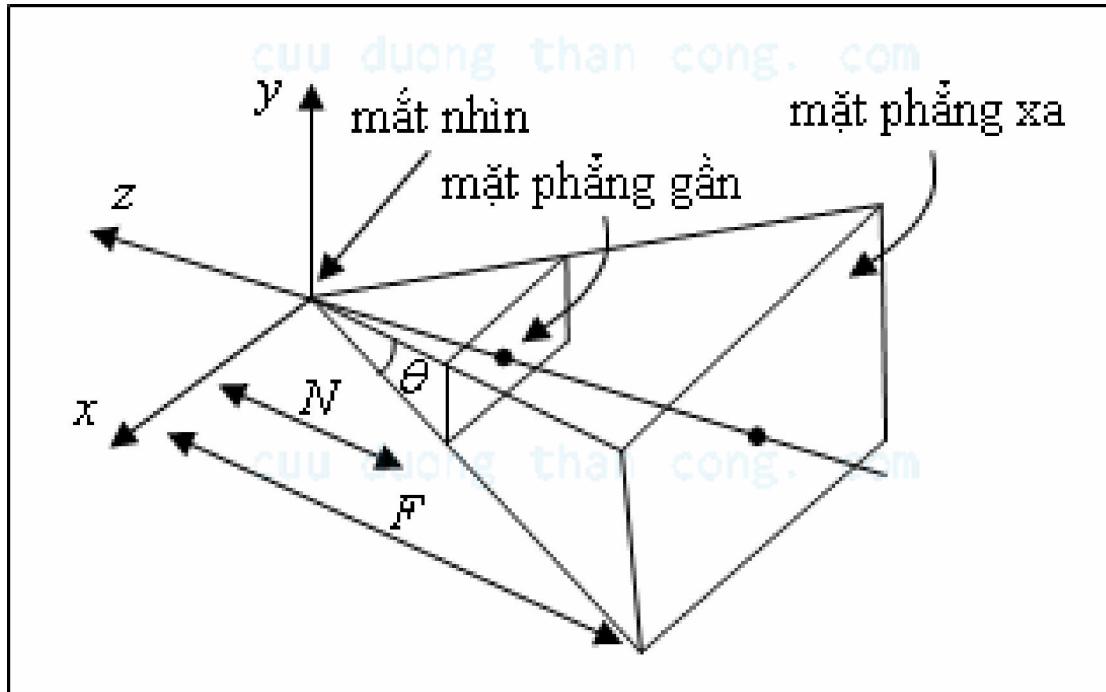
ÔN TẬP VỀ CAMERA

- Thiết lập thể tích nhìn: hình dạng của thể tích nhìn chứa trong ma trận phép chiếu.

```
glMatrixMode(GL_PROJECTION) ;
```

```
glLoadIdentity() ;
```

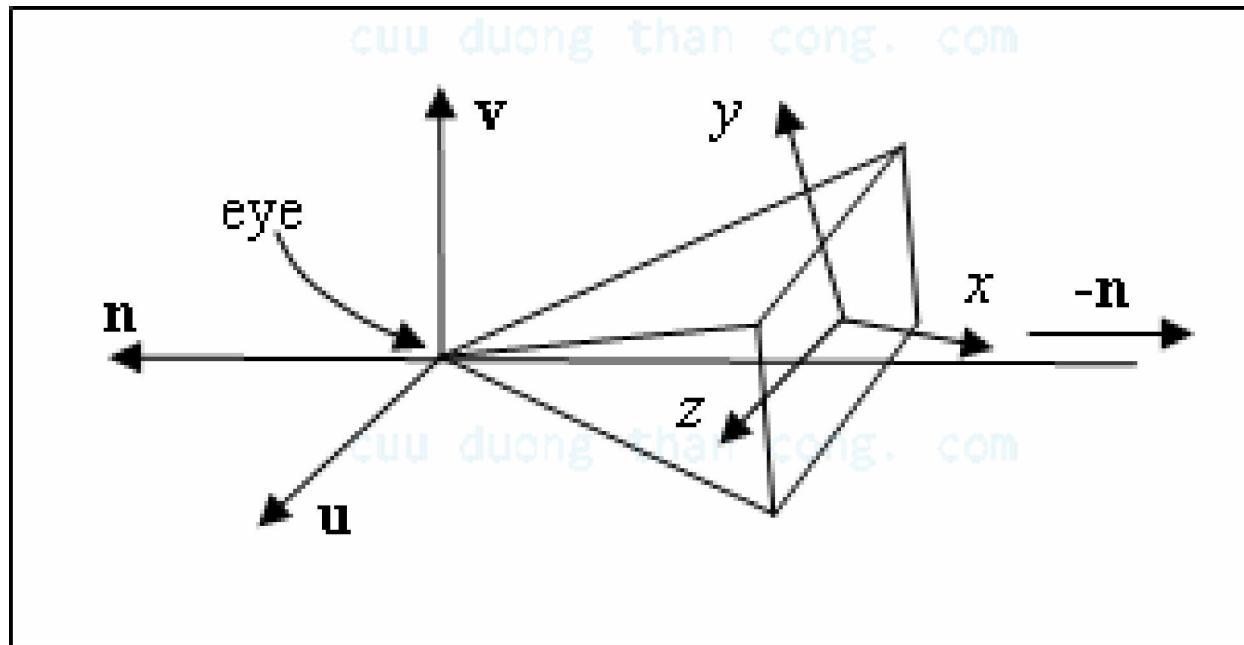
```
gluPerspective(viewAngle, aspectRatio, N, F) ;
```



ÔN TẬP VỀ CAMERA

□ Định vị trí và định hướng cho camera

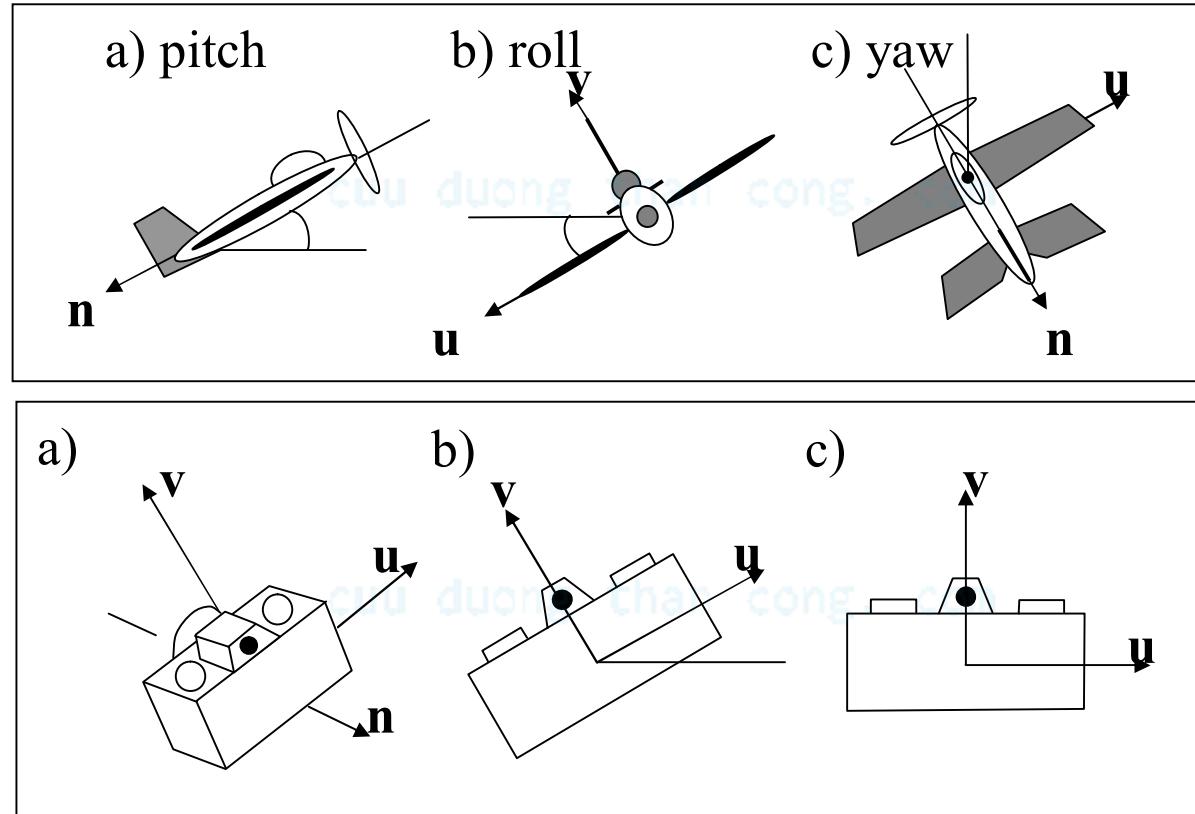
```
glMatrixMode(GL_MODELVIEW) ;  
glLoadIdentity() ;  
gluLookAt(eye.x, eye.y, eye.z, look.x, look.y, look.z,  
          up.x, up.y, up.z) ;
```



ÔN TẬP VỀ CAMERA

□ Định vị trí và định hướng cho camera

Khi điều chỉnh hướng của camera người ta thường dùng các thuật ngữ của ngành hàng không: pitch, roll, yaw

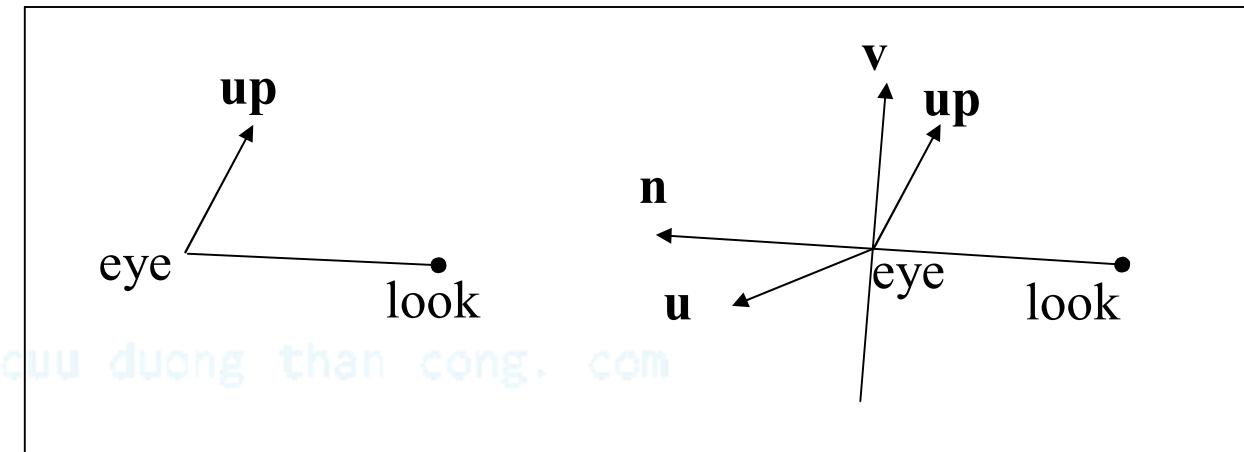


ÔN TẬP VỀ CAMERA

- ❑ Chức năng của gluLookAt(): từ eye, look, up \rightarrow u, v, n
 - n song song với eye – look
 - u, v vuông góc với n
 - up hướng trên camera
 - u chỉ phía bên camera

$n = \text{eye} - \text{look}$.
 $u = \text{up} \times n$,
 $v = n \times u$

cuuduongthancong.com

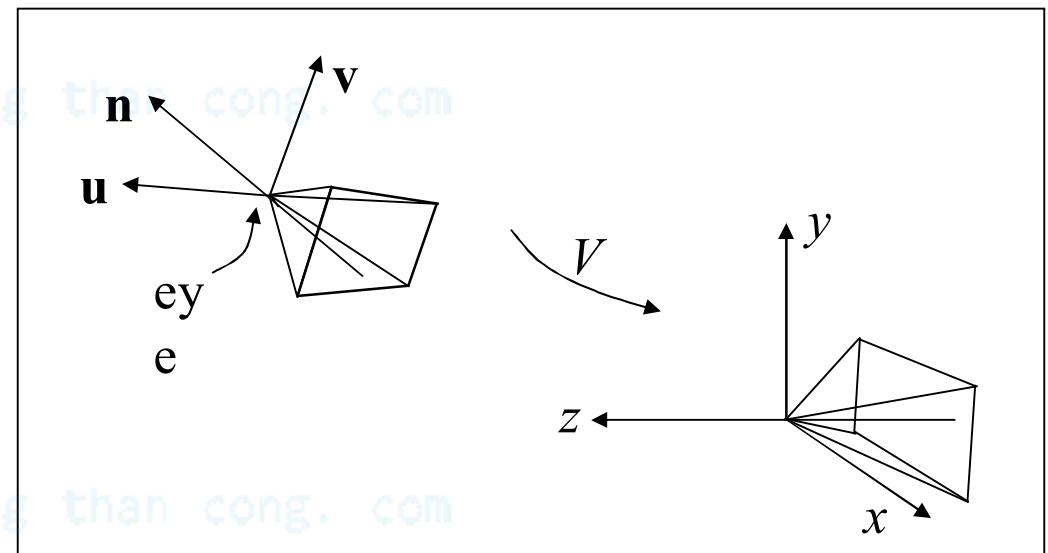


ÔN TẬP VỀ CAMERA

□ Ma trận mô hình-phép nhìn là tích của V và M

$$V = \begin{pmatrix} u_x & u_y & u_z & d_x \\ v_x & v_y & v_z & d_y \\ n_x & n_y & n_z & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (d_x, d_y, d_z) = (-\text{eye} \bullet \mathbf{u}, -\text{eye} \bullet \mathbf{v}, -\text{eye} \bullet \mathbf{n})$$

$$V \begin{pmatrix} \text{eye}_x \\ \text{eye}_y \\ \text{eye}_z \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad V \begin{pmatrix} u_x \\ u_y \\ u_z \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$



XÂY DỰNG CAMERA TRONG CT

```
class Camera
{
    private:
        Point3    eye;
        Vector3   u, v, n;
        double    viewAngle, aspect, nearDist, farDist;
        void      setModelViewMatrix();
    public:
        Camera();
        void set(Point3 Eye, Point3 look, Vector3 up);
        void roll(float angle);
        void pitch(float angle);
        void yaw(float angle);
        void slide(float delU, float delV, float delN);
        void setShape(float vAng, float asp, float nearD, float farD);
};
```

XÂY DỰNG CAMERA TRONG CT

```
void Camera :: setModelViewMatrix(){
    float m[16];
    Vector3 eVec(eye.x, eye.y, eye.z);
    m[0] = u.x ; m[4] = u.y; m[8] = u.z; m[12] = -eVec.dot(u);
    m[1] = v.x ; m[5] = v.y; m[9] = v.z; m[13] = -eVec.dot(v);
    m[2] = n.x ; m[6] = n.y; m[10] = n.z; m[14] = -eVec.dot(n);
    m[3] = 0   ; m[7] = 0  ; m[11] = 0  ; m[15] = 1.0;
    glMatrixMode(GL_MODELVIEW);
    glLoadMatrixf(m);      cuu duong than cong. com
}
void Camera :: set(Point3 Eye, Point3 look, Vector3 up){
    eye.set(Eye);
    n.set(eye.x - look.x, eye.y - look.y, eye.z - look.z);
    u.set(up.cross(n));
    n.normalize();
    u.normalize();      cuu duong than cong. com
    v.set(n.cross(u));
    setModelViewMatrix();
}
```

XÂY DỰNG CAMERA TRONG CT

- ☐ Trượt: di chuyển camera dọc theo các trục u, v và n mà không quay nó.

```
void Camera :: slide(float delU, float delV, float delN){  
    eye.x += delU*u.x + delV*v.x + delN*n.x;  
    eye.y += delU*u.y + delV*v.y + delN*n.y;  
    eye.z += delU*u.z + delV*v.z + delN*n.z;  
    setModelViewMatrix();  
}
```

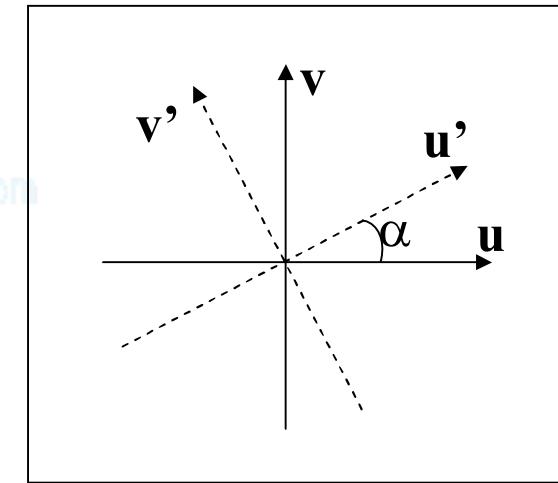
cuu duong than cong. com

XÂY DỰNG CAMERA TRONG CT

- Quay camera: quay xung quanh các trục u , v và n của nó
 - pitch (quay xung quanh u), roll (quay xung quanh n), yaw (quay xung quanh v).

$$u' = \cos(\alpha)u + \sin(\alpha)v,$$

$$v' = -\sin(\alpha)u + \cos(\alpha)v,$$



```
void Camera :: roll (float angle){  
    float cs = cos(3.14159265/180 * angle);  
    float sn = sin(3.14159265/180 * angle);  
    Vector3 t = u;  
    u.set(cs*t.x - sn*v.x, cs*t.y - sn*v.y, cs*t.z - sn*v.z);  
    v.set(sn*t.x + cs*v.x, sn*t.y + cs*v.y, sn*t.z + cs*v.z);  
    setModelViewMatrix();}
```

VÍ DỤ

```
#include <windows.h>
#include <gl/gl.h>
#include <gl/glut.h>
#include "camera.h"
Camera cam;
void myKeyboard(unsigned char key, int x, int y){
    switch(key){
        case 'F':    cam.slide(0, 0, 0.2); break;
        case 'F' - 64: cam.slide(0, 0, -0.2); break;
        case 'P':    cam.pitch(-1.0); break;
        case 'P' - 64: cam.pitch(1.0); break;
    }
    glutPostRedisplay();
}
void myInit(){
    glClearColor(1.0,1.0,1.0,0.0);
    glColor3f(0.0f, 0.0f, 0.0f);
    glPointSize(2.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 400.0, 0.0, 400.0);
}
```

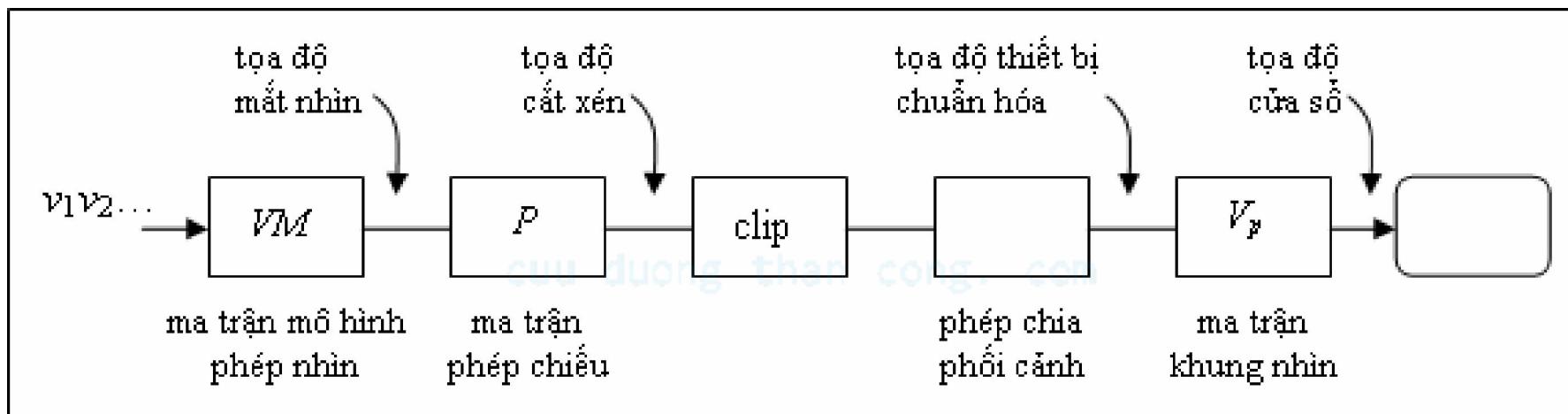
VÍ DỤ

```
void myDisplay(){  
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  
    glutWireTeapot(1.0);  
    glFlush();  
    glutSwapBuffers();}  
int main(int argc, char* argv[]){  
    ... khởi tạo cửa sổ  
    glutKeyboardFunc(myKeyboard);  
    glutDisplayFunc(myDisplay);  
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);  
    glColor3f(0.0f, 0.0f, 0.0f);  
    glViewport(0, 0, 640, 480);  
    cam.set(4, 4, 4, 0, 0, 0, 0, 1, 0);  
    cam.setShape(30.0f, 64.0f/48.0f, 0.5f, 50.0f);  
    glutMainLoop();  
    return 0;  
}
```

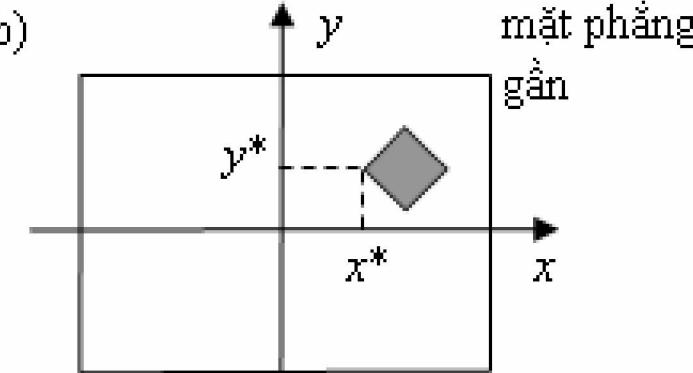
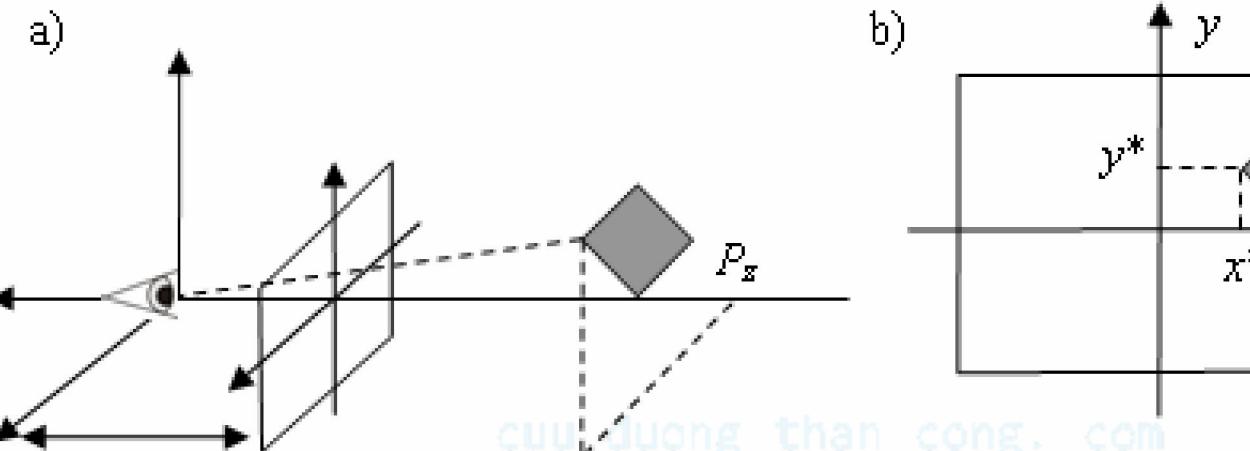
HÌNH CHIẾU PHỐI CẢNH

Đường ống đồ họa:

- điểm sau khi thực hiện phép biến đổi mô hình – phép nhìn sẽ có tọa độ nằm trong hệ tọa độ camera.
- **thực hiện phép chiếu phối cảnh**
- thực hiện việc cắt xén với cửa sổ trên mặt phẳng nhìn



HÌNH CHIẾU PHỐI CẢNH CỦA ĐIỂM



$$(P_x, P_y, P_z) \rightarrow (x^*, y^*)$$

$$\frac{x^*}{P_x} = \frac{N}{-P_z}$$

$$(x^*, y^*) = \left(N \frac{P_x}{-P_z}, N \frac{P_y}{-P_z} \right)$$

HÌNH CHIẾU PHỐI CẢNH CỦA ĐT

□ Phép chiếu những đường thẳng song song

- đt đi qua $A = (A_x, A_y, A_z)$ có hướng $c = (c_x, c_y, c_z) \rightarrow$ pt tham số $P(t) = A + ct$.

- hình chiếu đt là:

$$p(t) = \left(N \frac{A_x + c_x t}{-A_z - c_z t}, N \frac{A_y + c_y t}{-A_z - c_z t} \right)$$

- khi đt song song với mặt phẳng nhìn ($c_z = 0$), thì hình chiếu của đt là:

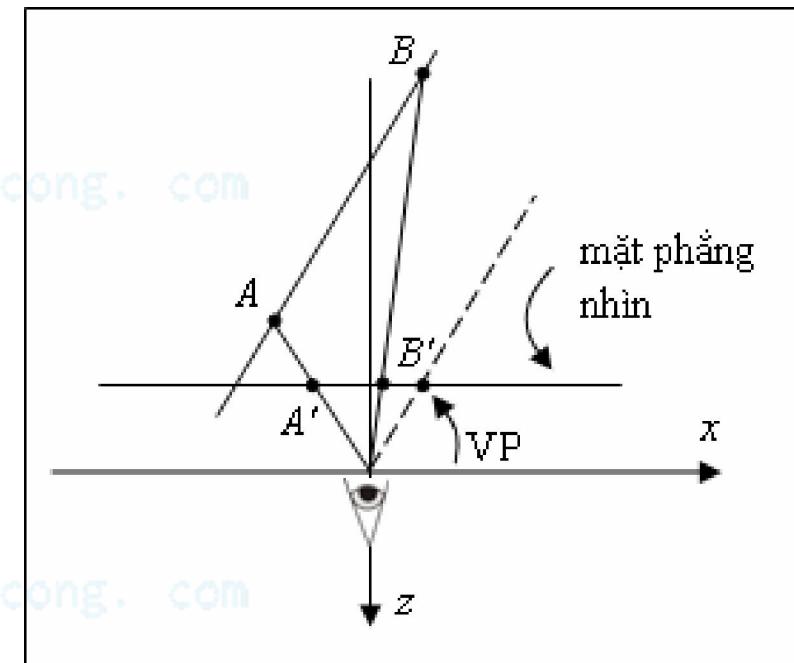
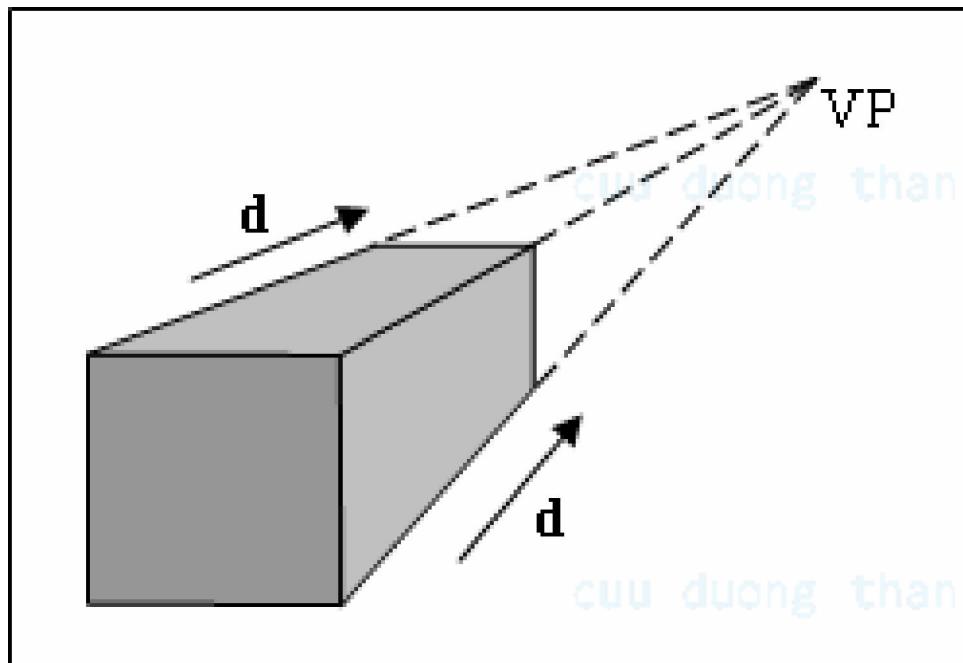
$$p(t) = \frac{N}{-A_z} (A_x + c_x t, A_y + c_y t)$$

- khi đt không song song với mặt phẳng nhìn, thì hình chiếu của một điểm nằm ở vô cùng là (điểm biến mất)

$$p(\infty) = \left(N \frac{c_x}{-c_z}, N \frac{c_x}{-c_z} \right)$$

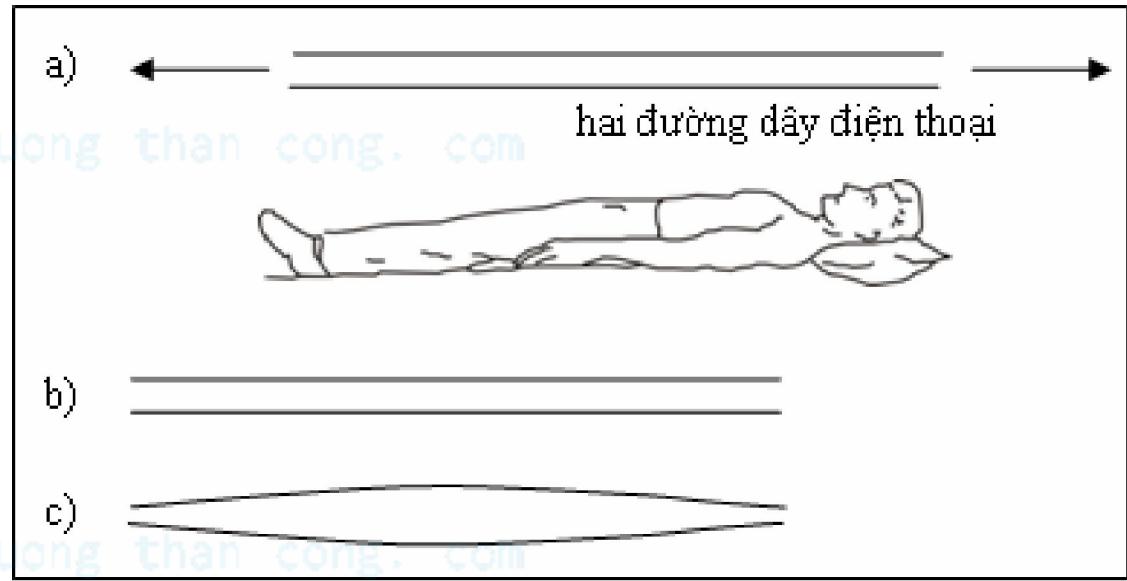
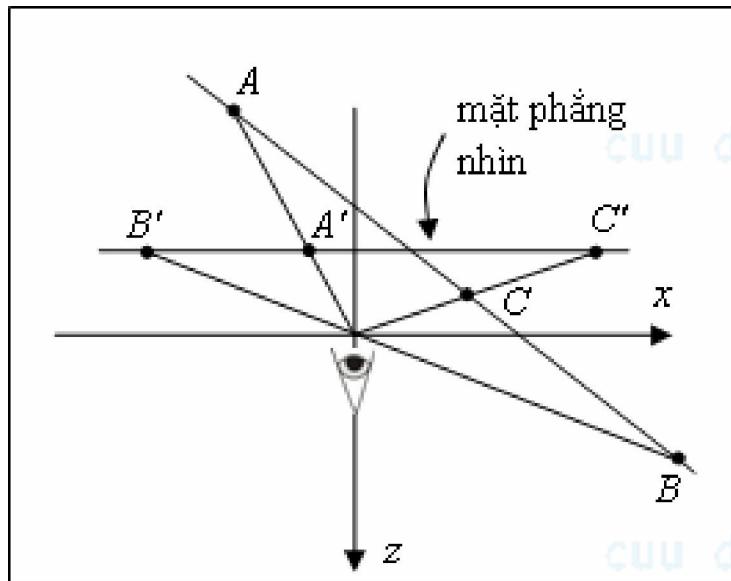
HÌNH CHIẾU PHỐI CẢNH CỦA ĐT

- Phép chiếu những đường thẳng song song



HÌNH CHIẾU PHỐI CẢNH CỦA ĐT

- Đường thẳng chạy ra phía sau mắt nhìn
- Bất hợp lý khi quan sát các đt dài song song

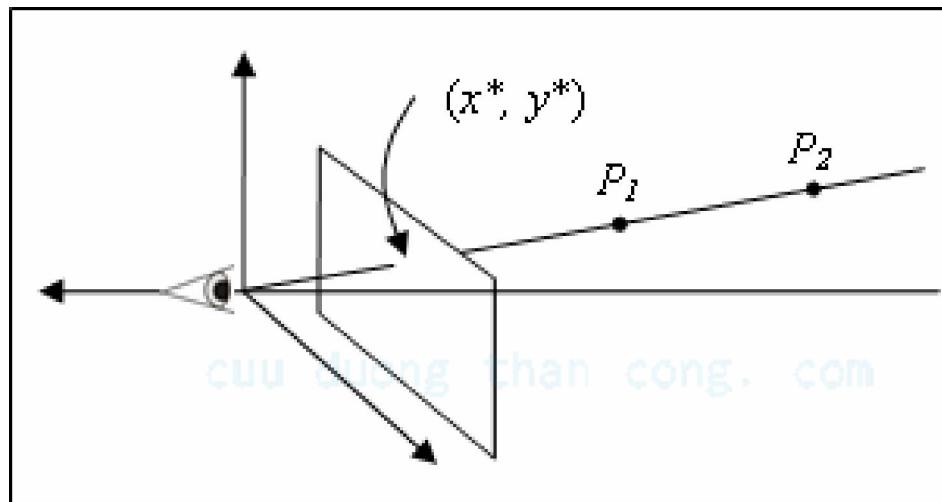


ĐỘ SÂU GIẢ

- ❑ Phép chiếu phối cảnh làm mất thông tin độ sâu
- ❑ Tốn tg khi tính độ sâu theo công thức $\sqrt{P_x^2 + P_y^2 + P_z^2}$
- ❑ Vì P_z càng âm thì điểm càng ở xa \rightarrow dùng P_z tính độ sâu

$$(x^*, y^*, z^*) = \left(N \frac{P_x}{-P_z}, N \frac{P_y}{-P_z}, \frac{aP_z + b}{-P_z} \right)$$

cuu duong than cong. com

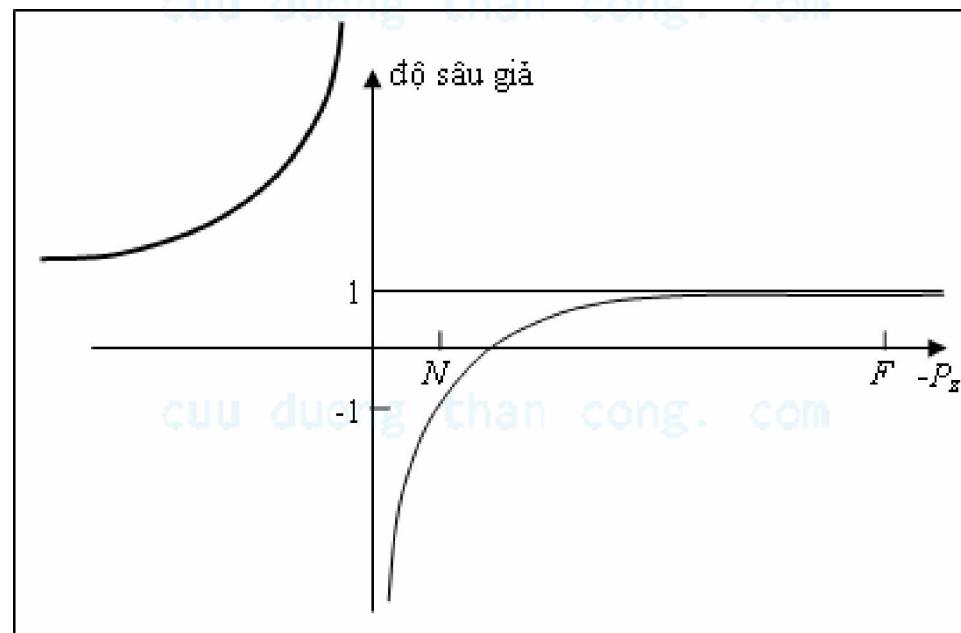


ĐỘ SÂU GIẢ

☐ a và b được chọn sao cho độ sâu giả nằm trong [-1, 1]

- chọn $z^* = -1$ khi $P_z = -N$
- chọn $z^* = 1$ khi $P_z = -F$

$$a = -\frac{F + N}{F - N}, b = \frac{-2FN}{F - N}$$



SỬ DỤNG TỌA ĐỘ ĐỒNG NHẤT

- Biểu diễn tọa độ đồng nhất của $P = (P_x, P_y, P_z)$ là $P = (P_x, P_y, P_z, 1)$ của vector $v = (v_x, v_y, v_z)$ là $v = (v_x, v_y, v_z, 0)$
- Mở rộng cách biểu diễn đồng nhất $P = (wP_x, wP_y, wP_z, w)$, cách điểm nằm trên cùng tia, điểm ở vô cùng có $w=0$.
- $(3, 6, 2, 3) \rightarrow$ tọa độ thông thường là $(1, 2, 2/3)$
- Cách chuyển đổi:
 - thông thường \rightarrow đồng nhất (thêm 1)
 - đồng nhất \rightarrow thông thường (chia cho thành phần tọa độ thứ 4 và bỏ đi thành phần tọa độ thứ 4)

cuuduongthancong.com

SỬ DỤNG TỌA ĐỘ ĐỒNG NHẤT

☐ Nhân với phép biến đổi affine

$$\begin{pmatrix} 2 & -1 & 3 & 1 \\ 6 & .5 & 1 & 4 \\ 0 & 4 & 2 & -3 \\ \hline 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} wP_x \\ wP_y \\ wP_z \\ w \end{pmatrix} = \begin{pmatrix} wQ_x \\ wQ_y \\ wQ_z \\ w \end{pmatrix}$$

☐ Nhân với ma trận chiếu (phép biến đổi phối cảnh)

$$\begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & a & b \\ \hline 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} wP_x \\ wP_y \\ wP_z \\ w \end{pmatrix} = \begin{pmatrix} wNP_x \\ wNP_y \\ w(aP_z + b) \\ -wP_z \end{pmatrix}$$

SỬ DỤNG TỌA ĐỘ ĐỒNG NHẤT

□ Phép chia phối cảnh

$$\left(N \frac{P_x}{-P_z}, N \frac{P_y}{-P_z}, \frac{aP_z + b}{-P_z} \right)$$

□ Phép biến đổi phối cảnh

$$(P_x, P_y, P_z) \rightarrow \left(N \frac{P_x}{-P_z}, N \frac{P_y}{-P_z}, \frac{aP_z + b}{-P_z} \right)$$

□ Phép chiếu trực giao

$$\left(N \frac{P_x}{-P_z}, N \frac{P_y}{-P_z}, \frac{aP_z + b}{-P_z} \right) \rightarrow \left(N \frac{P_x}{-P_z}, N \frac{P_y}{-P_z}, 0 \right)$$

(phép chiếu phối cảnh) = (phép biến đổi phối cảnh) +
(phép chiếu trực giao)

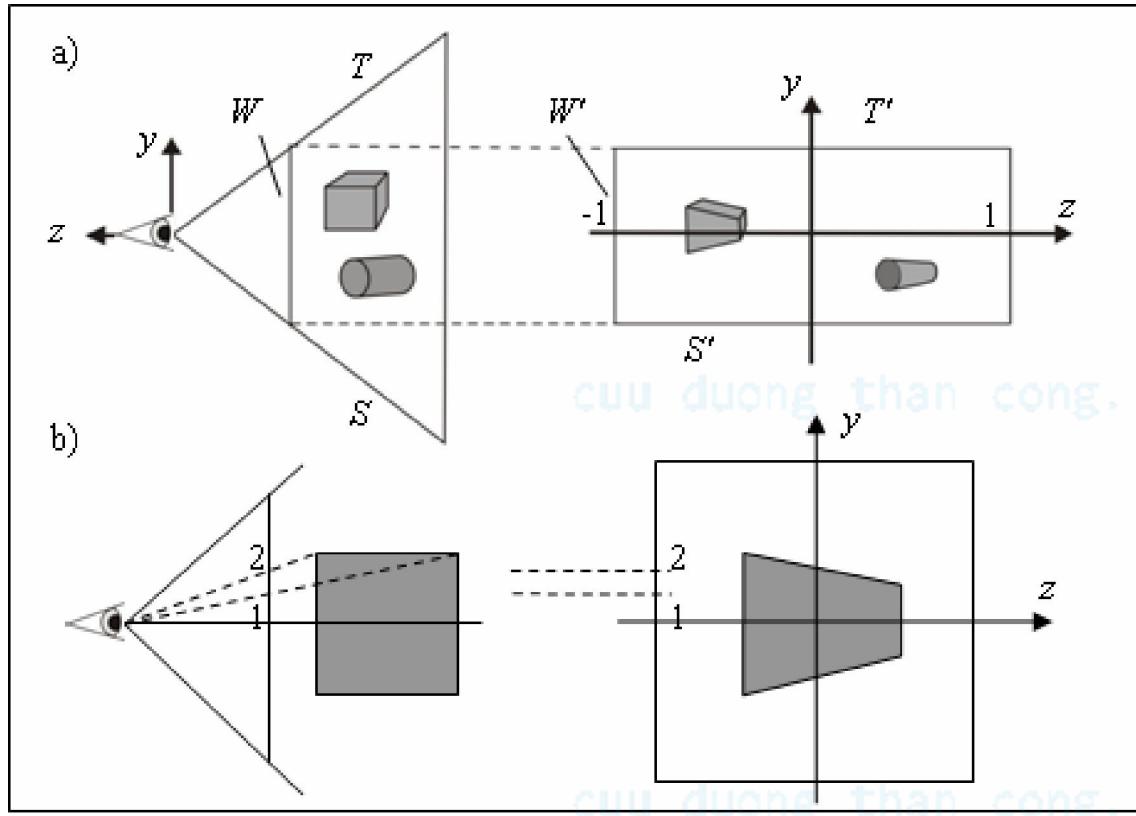
Ý NGHĨA HÌNH HỌC CỦA PHÉP BĐPC

- ❑ Biến đổi $P \rightarrow P'$ (cùng trong 3D)
- ❑ Bảo toàn tính thẳng, tính phẳng và tính nằm trong
- ❑ Làm cong không gian 3D
- ❑ Biến đổi thể tích nhìn thành một hình hộp

cuu duong than cong. com

cuu duong than cong. com

Ý NGHĨA HÌNH HỌC CỦA PHÉP BDPC



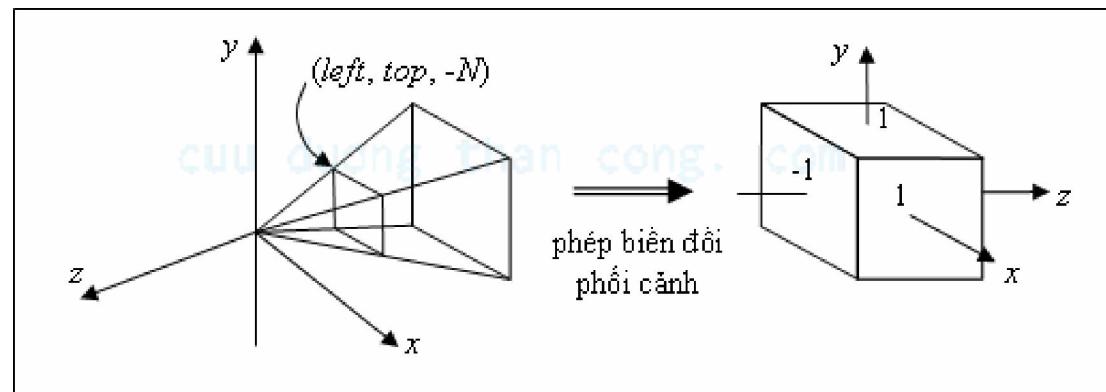
- $Z = -1$
- $Z = 1$
- $y = \text{top}$
- $y = \text{bottom}$
- $x = \text{left}$
- $x = \text{right}$

THỂ TÍCH NHÌN CHUẨN

- Sau khi thực hiện PBĐPC, chúng ta tịnh tiến, co dãn thể tích nhìn để được thể tích nhìn chuẩn. Ma trận là

$$R = \begin{pmatrix} \frac{2N}{right - left} & 0 & \frac{right + left}{right - left} & 0 \\ 0 & \frac{2N}{top - bott} & \frac{top + bott}{top - bott} & 0 \\ 0 & 0 & \frac{-(F + N)}{F - N} & \frac{-2FN}{F - N} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

- Thể tích nhìn chuẩn không phụ thuộc vào camera, dễ dàng cho cắt xén.



THỂ TÍCH NHÌN CHUẨN

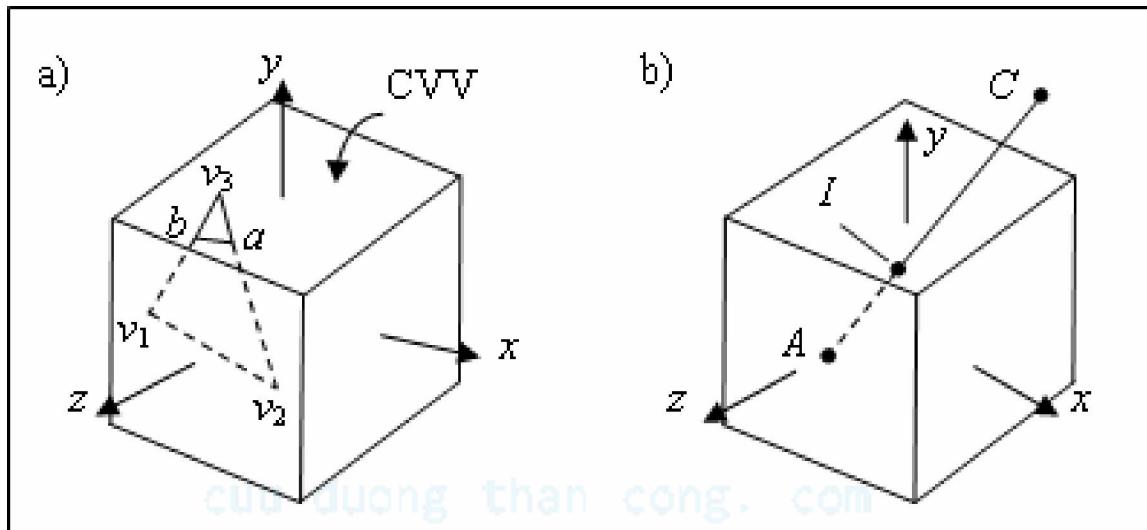
- ❑ Hàm `glFrustum(left, right, bott, top, N, F)` tạo ra ma trận R
- ❑ Hàm `gluPerspective(viewAngle, aspect, N, F)` cũng tạo ra ma trận R bằng cách tính các giá trị:

$$top = N \tan\left(\frac{\pi}{180} viewAngle / 2\right)$$

$$bott = -top, right = top \times aspect \text{ và } left = -right$$

cuu duong than cong. com

CẮT XÉN VỚI THỂ TÍCH NHÌN



- ❑ Điểm được biểu diễn dưới dạng tọa độ đồng nhất
- ❑ Sau khi thực hiện quá trình cắt xén xong, một số đỉnh có thể bị mất, một số đỉnh mới có thể được thêm vào

cuu duong than cong. con

CẮT XÉN VỚI THỂ TÍCH NHÌN

- ❑ Xác định tính trong ngoài của điểm so với mặt phẳng

$$- BC0 \quad w + x - x = -1 \quad BC1 \quad w - x - x = 1$$

$$- BC2 \quad w + y - y = -1 \quad BC3 \quad w - y - y = 1$$

$$- BC4 \quad w + z - z = -1 \quad BC5 \quad w - z - z = 1$$

- ❑ Chấp nhận đơn giản: 12 giá trị BC đều dương

- ❑ Loại bỏ đơn giản: cả hai đầu mút đều nằm ngoài một mp

- ❑ Tìm giao điểm

$$\text{edge}(t) = (a_x + (c_x - a_x)t, a_y + (c_y - a_y)t, a_z + (c_z - a_z)t, a_w + (c_w - a_w)t) \text{ với } x = 1$$

$$\frac{a_x + (c_x - a_x)t}{a_w + (c_w - a_w)t} = 1 \quad t = \frac{a_w - a_x}{(a_w - a_x) - (c_w - c_x)}$$

CẮT XÉN VỚI THẺ TÍCH NHÌN

```
int clipEdge(Point4& A, Point4& C)
```

```
{
```

```
    double tIn = 0.0, tOut = 1.0, tHit;
```

```
    double aBC[6], cBC[6];
```

```
    int aOutcode = 0, cOutcode = 0;
```

cuu duong than cong. com

... tìm các giá trị BC cho A và C...

... tạo mã cho A và C...

```
if ((aOutcode & cOutcode) != 0) //loại bỏ đơn giản
```

```
    return 0;
```

cuu duong than cong. com

```
if ((aOutcode | cOutcode) = 0) //chấp nhận đơn giản
```

```
    return 1;
```

CẮT XÉN VỚI THẺ TÍCH NHÌN

```
for(int i = 0; i < 6; i++)  
{  
    if(cBC[i] < 0) // đi ra: C nằm ngoài  
    {  
        tHit = aBC[i]/(aBC[i] – cBC[i]);  
        tOut = MIN(tOut, tHit);  
    }  
    else if (aBC[i] < 0) // đi vào: A nằm ngoài  
    {  
        tHit = aBC[i]/(aBC[i] – cBC[i]);  
        tIn = MAX(tIn, tHit);  
    }  
    if(tIn > tOut) return 0; // CI rỗng; kết thúc sớm  
}
```

CẮT XÉN VỚI THẺ TÍCH NHÌN

```
// Cập nhật các điểm đầu mút nếu cần thiết
Point4 tmp;
if(aOutcode != 0) { // A nằm ngoài: tIn thay đổi
    tmp.x = A.x + tIn * (C.x - A.x);
    tmp.y = A.y + tIn * (C.y - A.y);
    tmp.z = A.z + tIn * (C.z - A.z);
    tmp.w = A.w + tIn * (C.w - A.w);
}
if(cOutcode != 0) { // C nằm ngoài: tOut thay đổi
    C.x = A.x + tOut * (C.x - A.x);
    C.y = A.y + tOut * (C.y - A.y);
    C.z = A.z + tOut * (C.z - A.z);
    C.w = A.w + tOut * (C.w - A.w);
}
A = tmp; // cập nhật A
return 1;}
```

Trường Đại Học Bách Khoa TP Hồ Chí Minh
Khoa Khoa học & Kỹ thuật Máy tính



ĐỒ HỌA MÁY TÍNH

CHƯƠNG 8:

TÔ MÀU

VẬT THỂ 3 CHIỀU

NỘI DUNG TRÌNH BÀY

- ❑ Giới thiệu
- ❑ Mô hình tô màu
- ❑ Sử dụng nguồn sáng
- ❑ Tô màu phẳng, tô màu trơn
- ❑ Dán texture lên mặt đa giác

cuu duong than cong. com

cuu duong than cong. com

GIỚI THIỆU

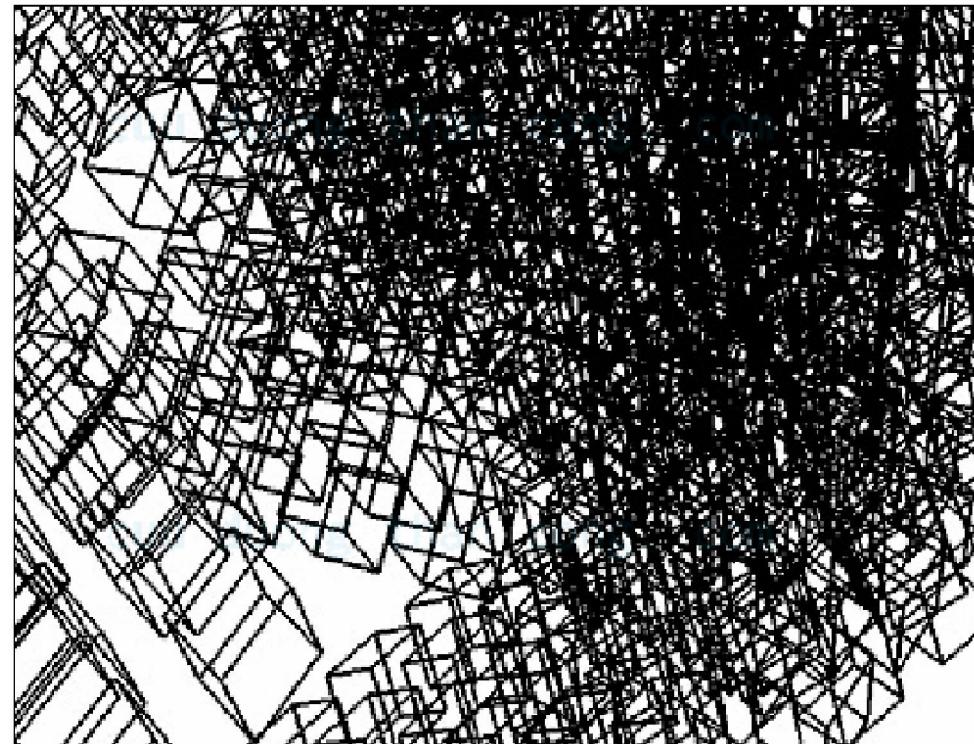
- ❑ Tập trung vào tính toán màu sắc cho pixel
- ❑ Dựa vào mô hình màu (mô hình hóa việc ánh sáng tương tác với đối tượng)
- ❑ Không mô phỏng hết các nguyên lý vật lý của sự phát tán và phản xạ ánh sáng
- ❑ Đưa ra mô hình mang tính xấp xỉ và tạo nên nhiều mức độ chân thực khác nhau

cuu duong than cong. com

GIỚI THIỆU

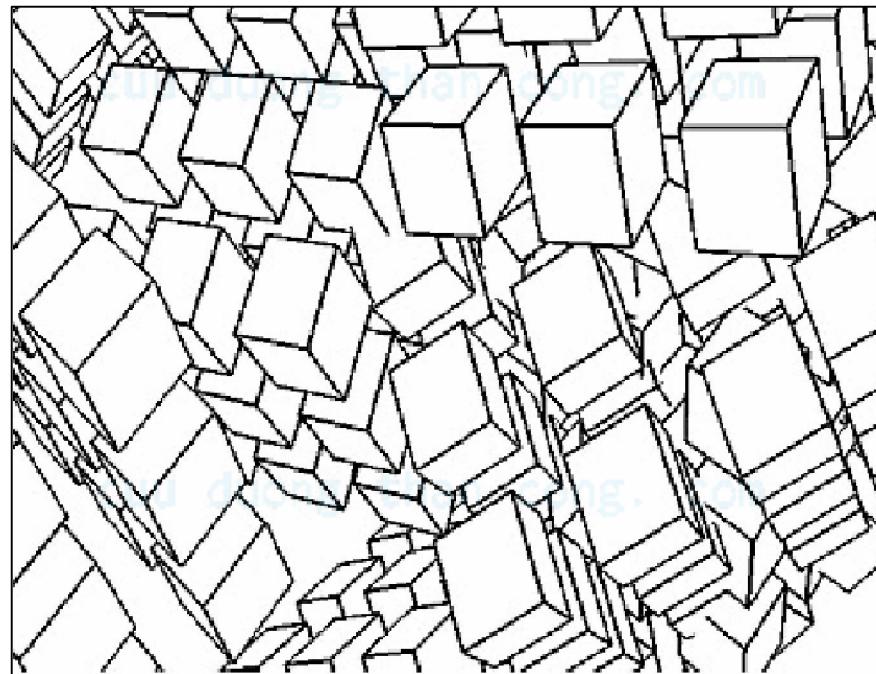
□ Mô hình khung dây (wireframe)

- đơn giản, chỉ có các cạnh được vẽ
- nhìn xuyên qua vật thể, khó phân biệt được vật thể



GIỚI THIỆU

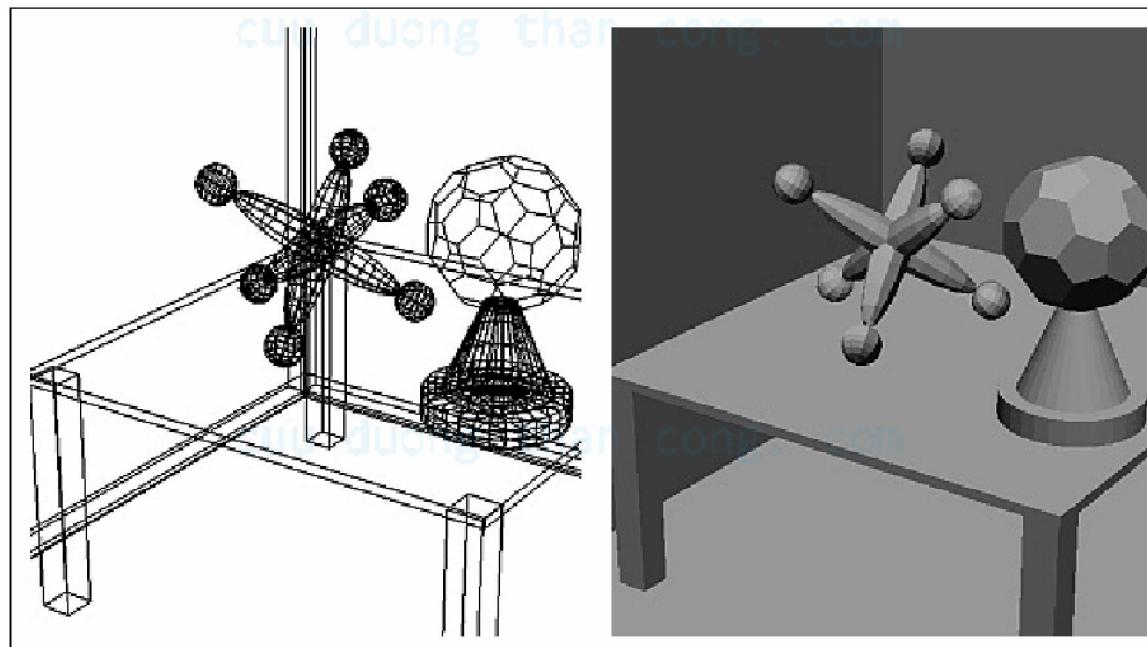
- Line Drawing: mô hình khung dây với các mặt khuất được loại bỏ.
 - chỉ có các cạnh được vẽ
 - nhìn giống khối rắn, phân biệt được vật thể



GIỚI THIỆU

❑ Tô màu phẳng (flat shading):

- ánh sáng phản xạ được tính tại một điểm bất kỳ
- các điểm thuộc mặt được tô bởi một màu
- thấy rõ ranh giới giữa các mặt

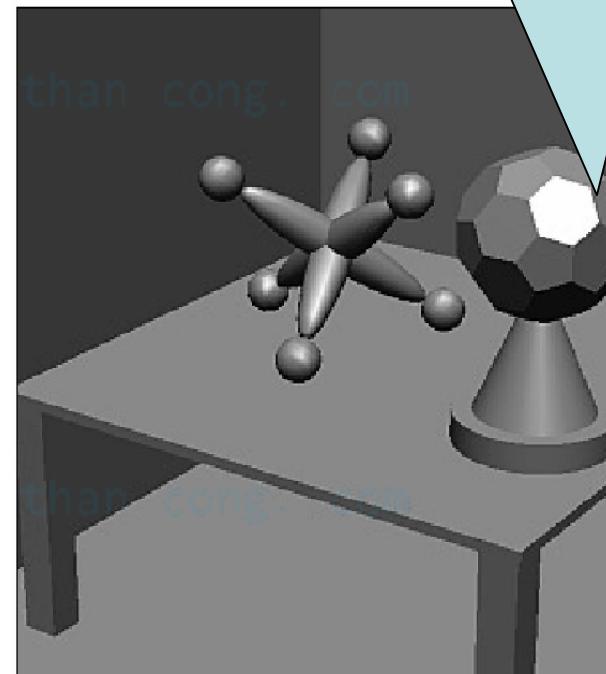
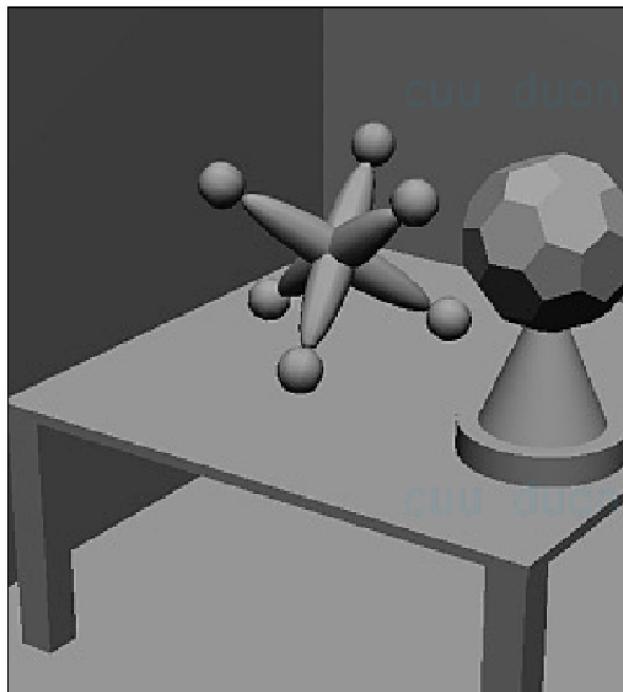


GIỚI THIỆU

☐ Tô màu mượt (smooth shading):

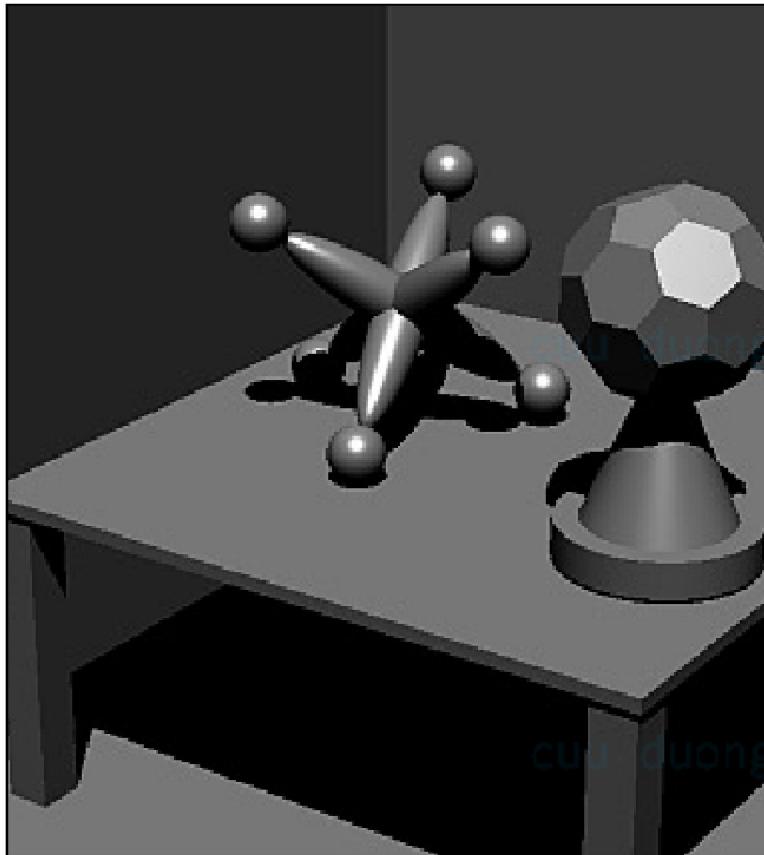
- màu được tính ở một số điểm, sau đó nội suy cho các điểm còn lại.
- ranh giới giữa các mặt biến mất

a/s phản chiếu



GIỚI THIỆU

- ☐ Tạo bóng đổ, dán texture lên bề mặt đối tượng

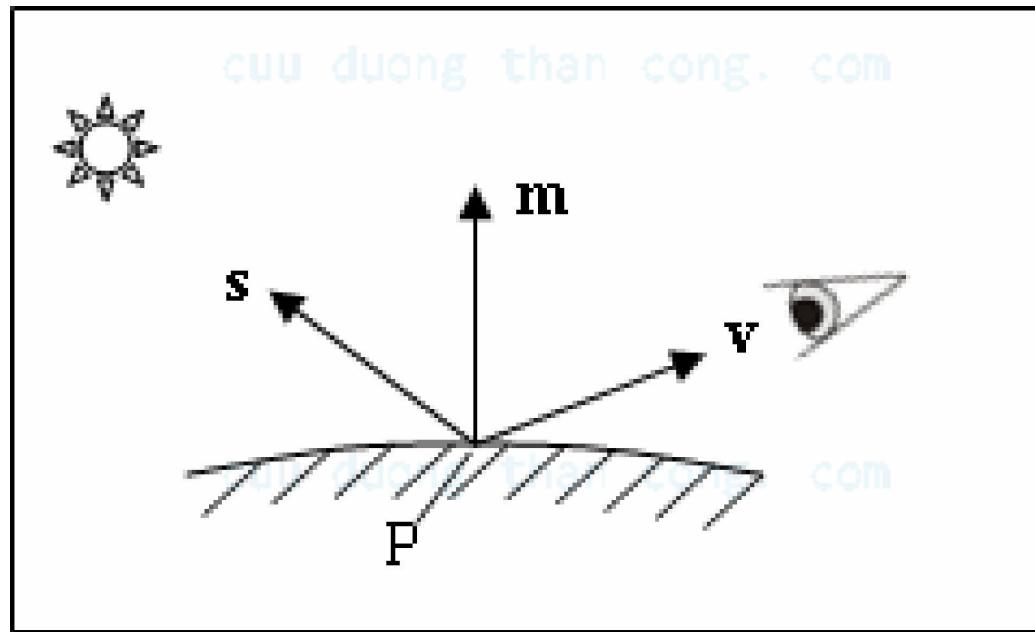


MÔ HÌNH TÔ MÀU

- ❑ Nguồn sáng không màu có nhiều mức xám
- ❑ Nguồn sáng điểm và nguồn sáng nền
- ❑ Ánh sáng tới tương tác với bề mặt theo 3 cách: (a) hấp thụ chuyển thành nhiệt, (b) **phản xạ**, (c) truyền vào trong
- ❑ Thành phần ánh sáng phản xạ
 - A/s khuyếch tán: phát tán theo mọi hướng, tương tác mạnh với bề mặt, màu sắc phụ thuộc vào màu sắc vật thể
 - A/s phản chiếu: tính định hướng cao, không đi vào vật thể, phản xạ ở bề mặt vật thể, tăng độ sáng bề mặt đối tượng, màu giống màu a/s tới

MÔ HÌNH TÔ MÀU

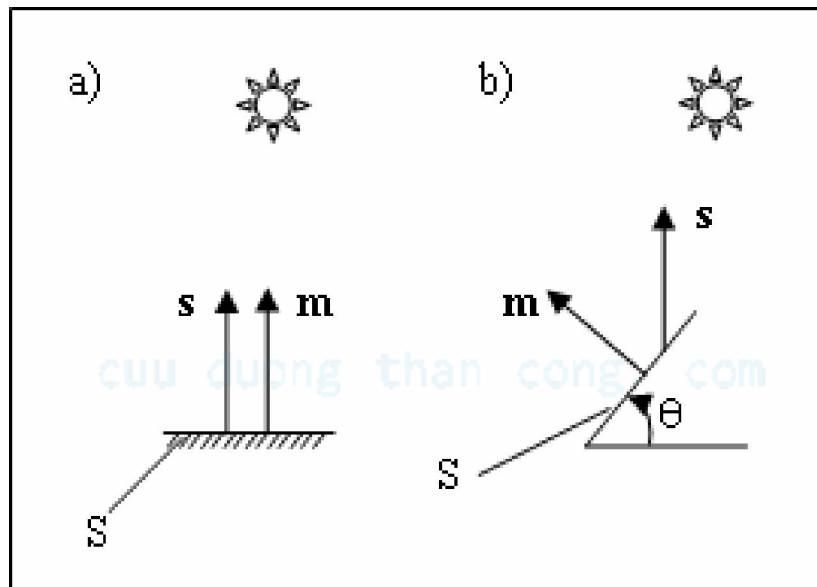
- Yếu tố hình học để xác định ánh sáng phản xạ
 - vector pháp tuyến m của bề mặt
 - vector v từ P đến mắt nhìn
 - vector s từ P đến nguồn sáng



MÔ HÌNH TÔ MÀU

□ Thành phần ánh sáng khuyếch tán

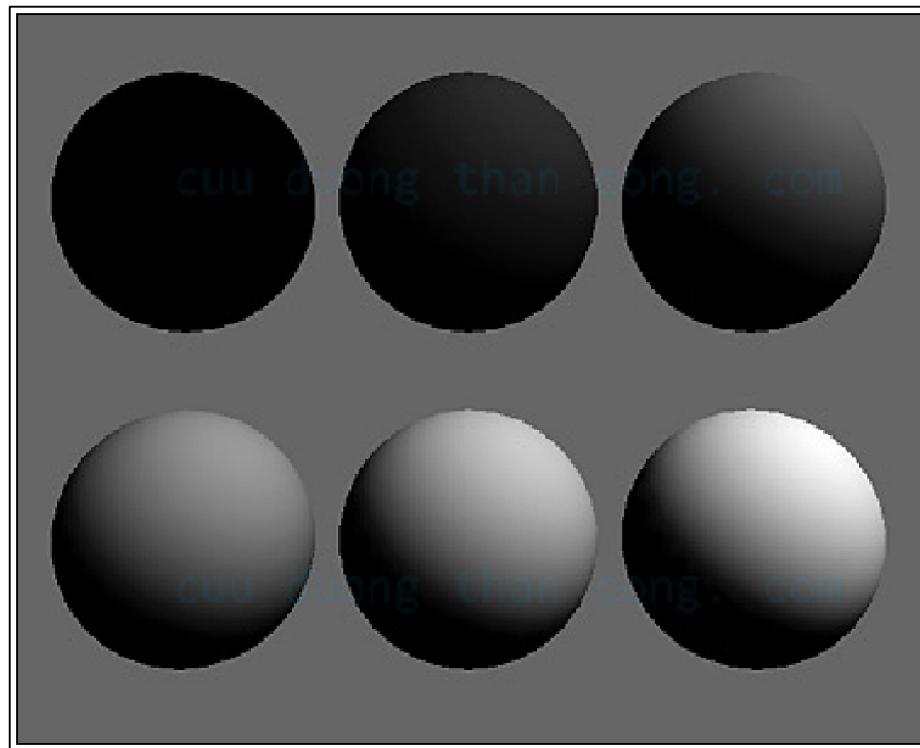
- phần ánh xạ phản xạ đến được mắt nhìn, ký hiệu I_d
- khuyếch tán đồng nhất theo mọi hướng \rightarrow chỉ phụ thuộc m, s .
- định luật Lambert: $I_d = I_s \rho_d \frac{s \bullet m}{|s|m|}$ $I_d = I_s \rho_d \max\left(\frac{s \bullet m}{|s|m|}, 0\right)$
- I_s : cường độ n/s, ρ_d : h/s p/x khuyếch tán



MÔ HÌNH TÔ MÀU

❑ Thành phần ánh sáng khuyếch tán

- Ví dụ: hệ số p/x khuyếch tán 0, 0.2, 0.4, 0.6, 0.8, 1.0.
Cường độ nguồn sáng là 1, cường độ a/s nền 0.4

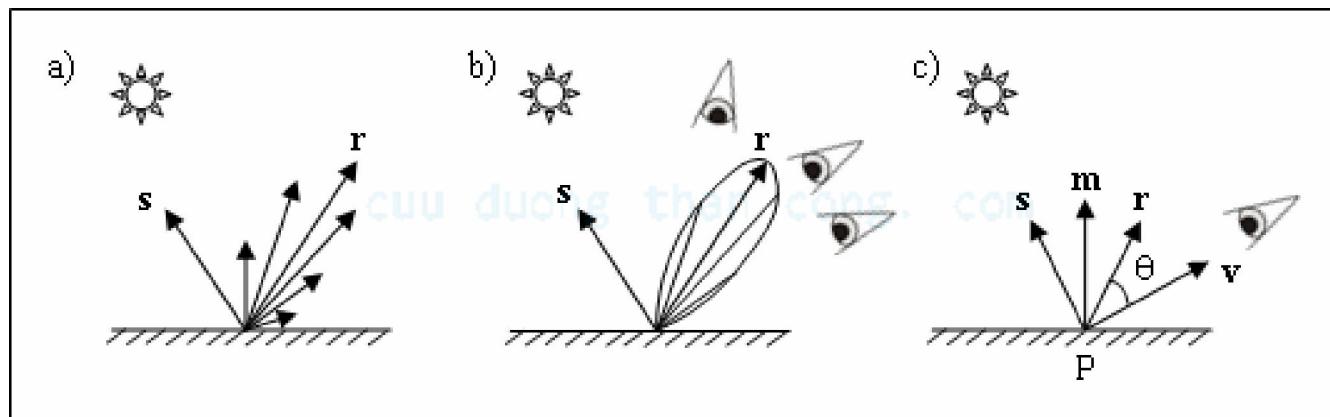


MÔ HÌNH TÔ MÀU

□ Thành phần ánh sáng phản chiếu

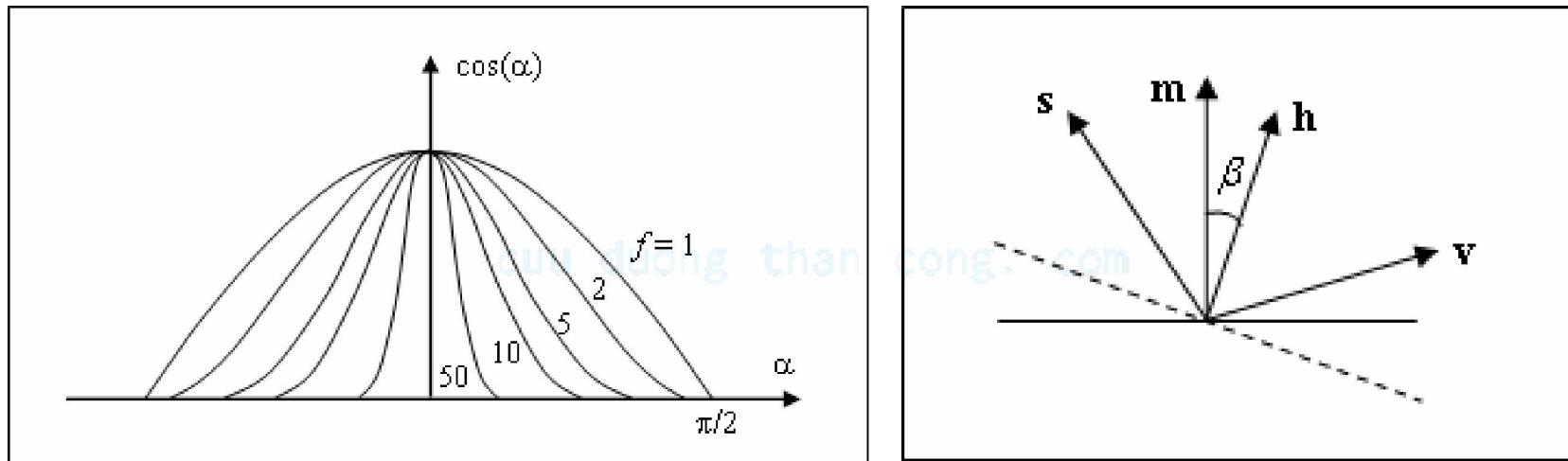
- tăng độ sáng, tăng mức độ chân thực cho đối tượng có đặc tính sáng bóng.
- lượng ánh sáng phản xạ lớn ở hướng p/x r.

$$r = -s + 2 \frac{s \bullet m}{|m|^2} m \quad I_{sp} = I_s \rho_s \left(\frac{r}{|r|} \bullet \frac{v}{|v|} \right)^f \quad f : [1, 200]$$



MÔ HÌNH TÔ MÀU

- Thành phần ánh sáng phản chiếu
 - $f \rightarrow$ vô cùng: tập trung chủ yếu ở hướng p/x



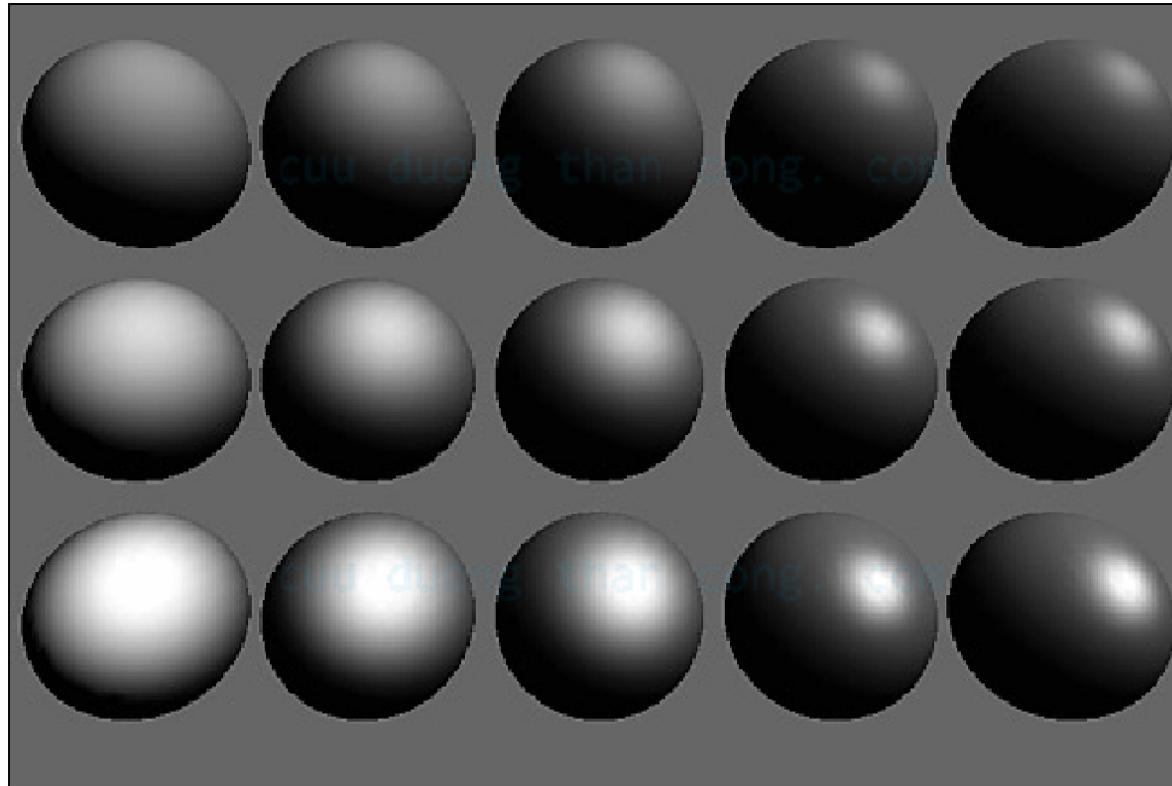
Giảm tg tính toán, dùng vector nửa đường $h = s + v$. $\beta = 0$, nhìn thấy lượng a/s phản chiếu nhiều nhất

$$I_{sp} = I_s \rho_s \max\left(0, \left(\frac{h}{|h|} \bullet \frac{m}{|m|}\right)^f\right)$$

MÔ HÌNH TÔ MÀU

□ Thành phần ánh sáng phản chiếu

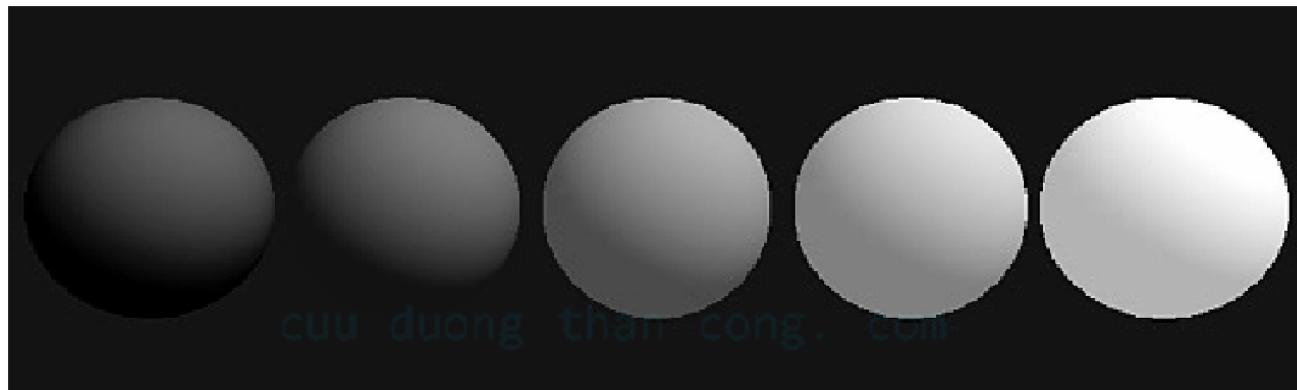
- hệ số ρ_s từ trên xuống là 0.25, 0.5, 0.75. Hệ số f từ trái sang phải là 3, 6, 9, 25, 200.



MÔ HÌNH TÔ MÀU

☐ Ánh sáng nền:

- khắc phục hiện tượng phần không nhận được a/s có màu đen hoàn toàn, thêm a/s nền.
- không có vị trí cố định, chiếu sáng đồng nhất theo mọi hướng
- Thêm I_a : cường độ, ρ_a : hệ số phản chiếu nền



$\rho_a : 0.0, 0.1, 0.3, 0.5, 0.7$: càng lớn khung cảnh càng sáng

MÔ HÌNH TÔ MÀU

- ☐ Tổng hợp các thành phần ánh sáng

$$I = I_a \rho_a + I_d \rho_d \times lambert + I_{sp} \rho_s \times phong^f$$

$$lambert = \max(0, \frac{s \cdot m}{\|s\| \|m\|}) \quad \text{and} \quad phong = \max(0, \frac{h \cdot m}{\|h\| \|m\|})$$

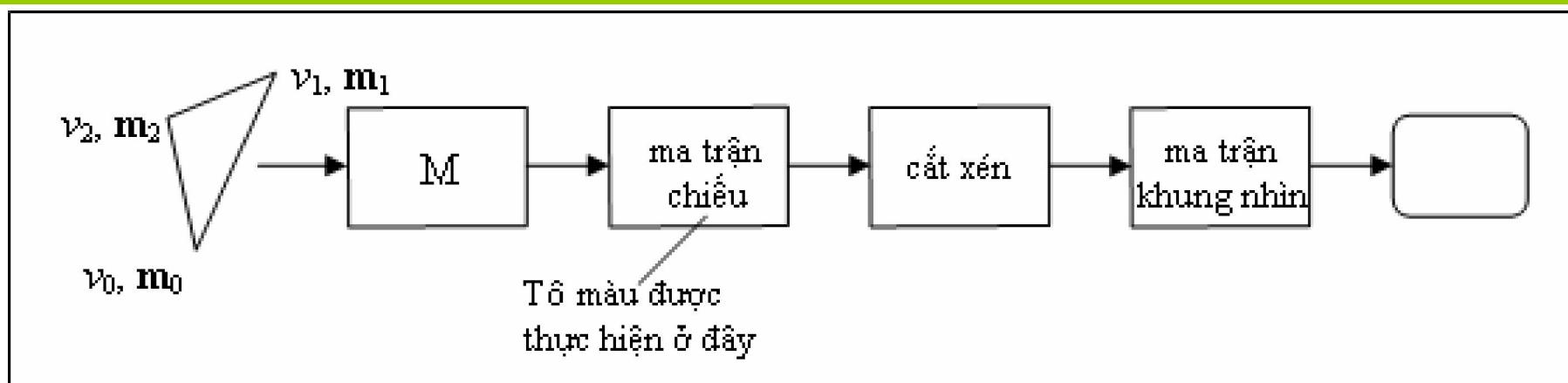
- ☐ Thêm màu sắc

$$I_r = I_{ar} \rho_{ar} + I_{dr} \rho_{dr} \times lambert + I_{spr} \rho_{sr} \times phong^f$$

$$I_g = I_{ag} \rho_{ag} + I_{dg} \rho_{dg} \times lambert + I_{spg} \rho_{sg} \times phong^f$$

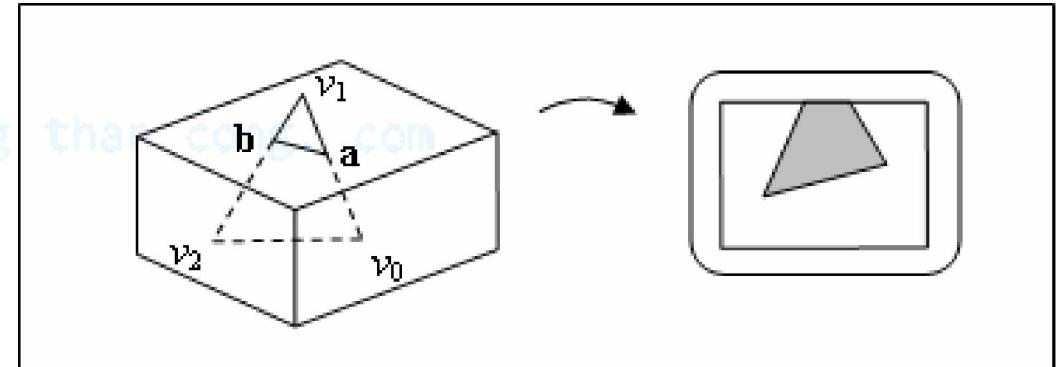
$$I_b = I_{ab} \rho_{ab} + I_{db} \rho_{db} \times lambert + I_{spb} \rho_{sb} \times phong^f$$

MÔ HÌNH TÔ MÀU



- Chỉ định camera, nguồn sáng, vector pháp tuyến đỉnh
- Sau khi thực hiện phép biến đổi mô hình-phép nhìn, tất cả được biểu diễn trong hệ tọa độ camera

```
cuu duong than cong.com
glBegin(GL_POLYGON);
for(int i = 0; i< 3; i++){
    glNormal3f(norm[i].x, norm[i].y, norm[i].z);
    glVertex3f(pt[i].x, pt[i].y, pt[i].z);}
glEnd();
```



SỬ DỤNG NGUỒN SÁNG

❑ Tạo nguồn sáng

- vị trí

```
GLfloat myLightPosition[] = {3.0, 6.0, 5.0, 1.0};  
glLightfv(GL_LIGHT0, GL_POSITION, myLightPosition);  
 glEnable(GL_LIGHTING); //enable  
 glEnable(GL_LIGHT0); //enable this particular source  
(x, y, z, 1) → nguồn sáng điểm, (x, y, z, 0) → n/s định hướng
```

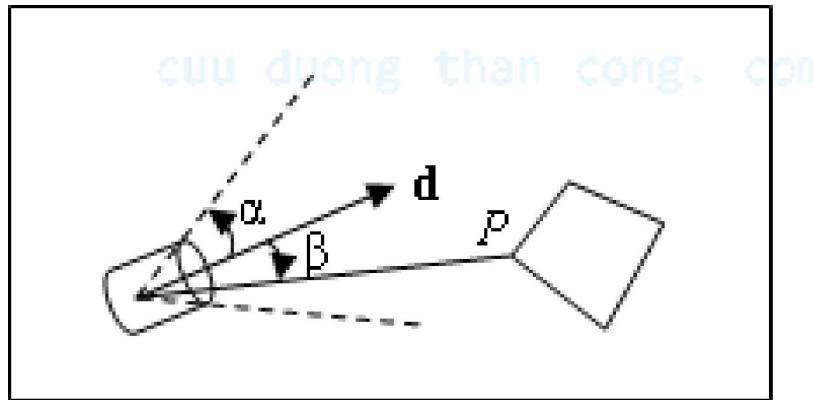
- màu sắc

```
GLfloat amb0[] = {0.2, 0.4, 0.6, 1.0};  
GLfloat diff0[] = {0.8, 0.9, 0.5, 1.0};  
GLfloat spec0[] = {1.0, 0.8, 1.0, 1.0};  
glLightfv(GL_LIGHT0, GL_AMBIENT, amb0);  
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff0);  
glLightfv(GL_LIGHT0, GL_SPECULAR, spec0);
```

SỬ DỤNG NGUỒN SÁNG

❑ Nguồn sáng dạng đèn pha

```
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0); // góc cắt bằng 45°  
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 4.0); // ε = 4.0  
GLfloat dir[] = {2.0, 1.0, -4.0}; // hướng của nguồn sáng  
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, dir);
```



❑ Suy giảm ánh sáng theo khoảng cách

$$atten = \frac{1}{k_c + k_l D + k_q D^2}$$

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 2.0);  
GL_LINEAR_ATTENUATION và GL_QUADRATIC_ATTENUATION
```

SỬ DỤNG NGUỒN SÁNG

☐ Chỉ định thuộc tính vật liệu

```
GLfloat myDiffuse[] = {0.8, 0.2, 0.0, 1.0};
```

```
glMaterialfv(GL_FRONT, GL_DIFFUSE, myDiffuse);
```

GL_BACK,
GL_FRONT_AND_BACK

GL_AMBIENT,
GL_SPECULAR, GL_EMISSION

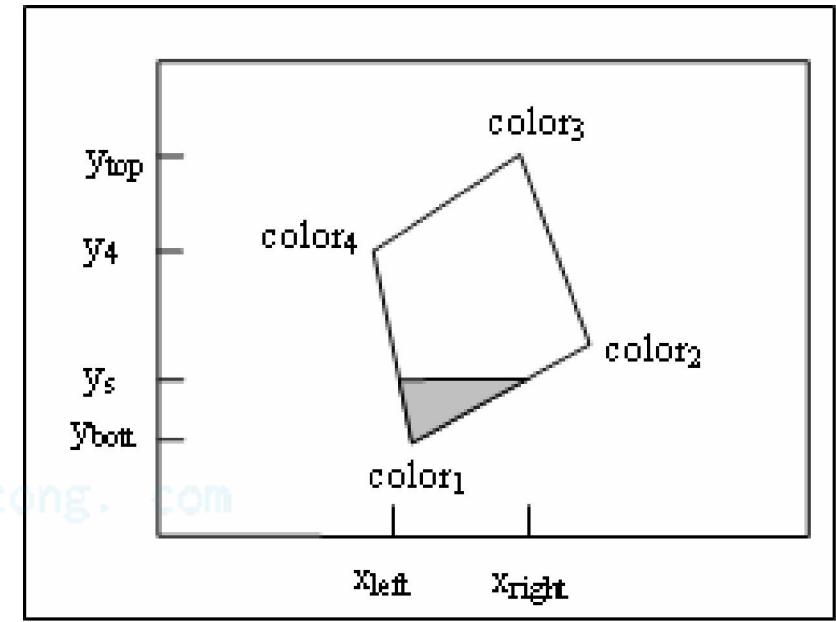
$$I_r = e_r + I_{mr} \rho_{ar} + \sum_i atten_i \times spot_i \times (I_{ar}^i \rho_{ar} + I_{dr}^i \rho_{dr} \times lambert_i + I_{spr}^i \rho_{sr} \times phong_i^f)$$

cuu duong than cong. com

TÔ MÀU PHẲNG, TÔ MÀU TRƠN

☐ Tô màu mặt đa giác

- di chuyển theo các đường quét, xác định màu cho mỗi pixel.
- đa giác lồi hiệu quả vì mỗi đường quét chỉ cắt đa giác tối đa ở 2 điểm

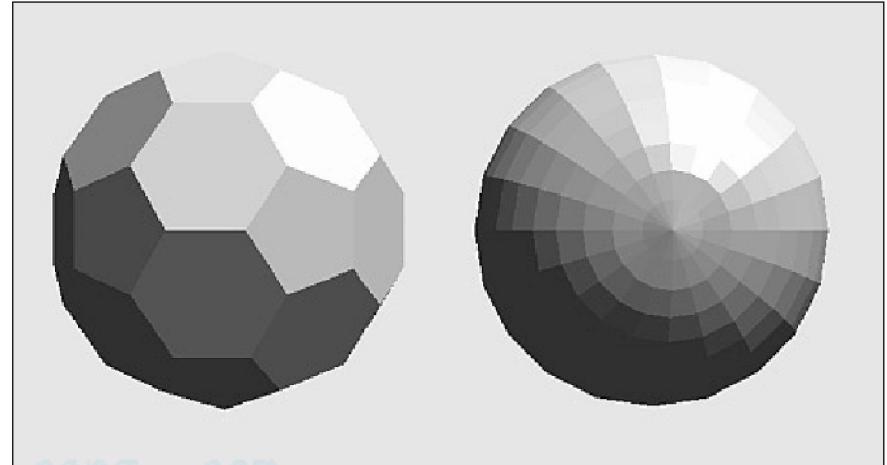


```
for (int y = ybott ; y <= ytop ; y++) {  
    find xleft and xright;  
    for (int x = xleft; x <= xright; x++) {  
        find the color c for this pixel;  
        put c into the pixel at (x, y); } }
```

TÔ MÀU PHẲNG, TÔ MÀU TRƠN

❑ Tô màu phẳng:

- nguồn sáng ở xa, mặt là đa giác phẳng → các điểm thuộc mặt có cùng màu sắc
- glShadeModel(GL_FLAT);



```
for (int y = ybott ; y <= ytop ; y++) {  
    find xleft and xright;  
    find the color c for this scan line;  
    for (int x = xleft; x <= xright; x++) {  
        put c into the pixel at (x, y); } }
```

TÔ MÀU PHẲNG, TÔ MÀU TRƠN

□ Tô màu Gouraud:

$$\text{color}_{\text{left}} = \text{lerp}(\text{color}_1, \text{color}_4, f), \quad f = \frac{y_s - y_{\text{bott}}}{y_4 - y_{\text{bott}}}$$

$$c(x) = \text{lerp}\left(\text{color}_{\text{left}}, \text{color}_{\text{right}}, \frac{x - x_{\text{left}}}{x_{\text{right}} - x_{\text{left}}}\right)$$

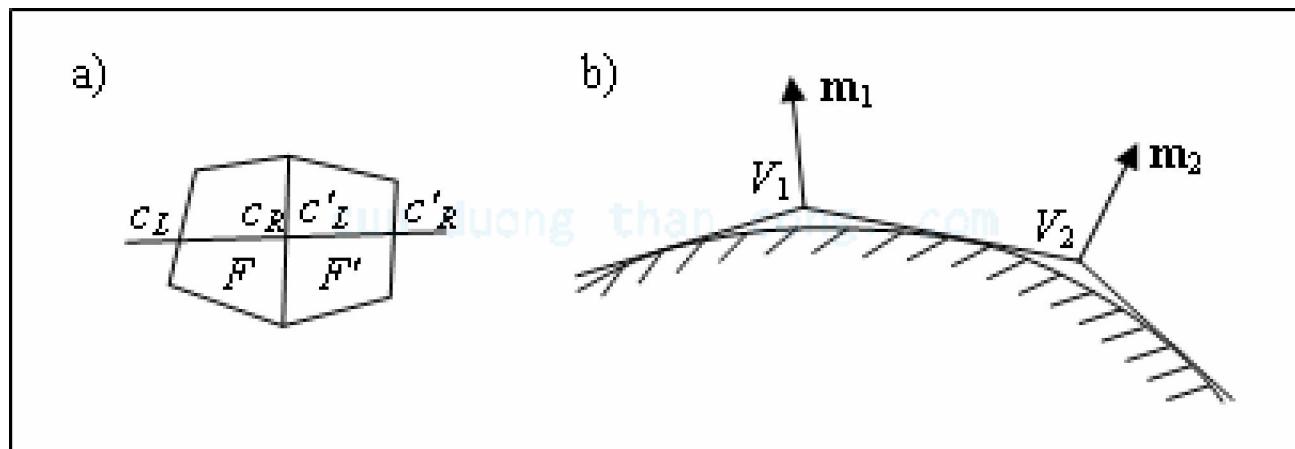
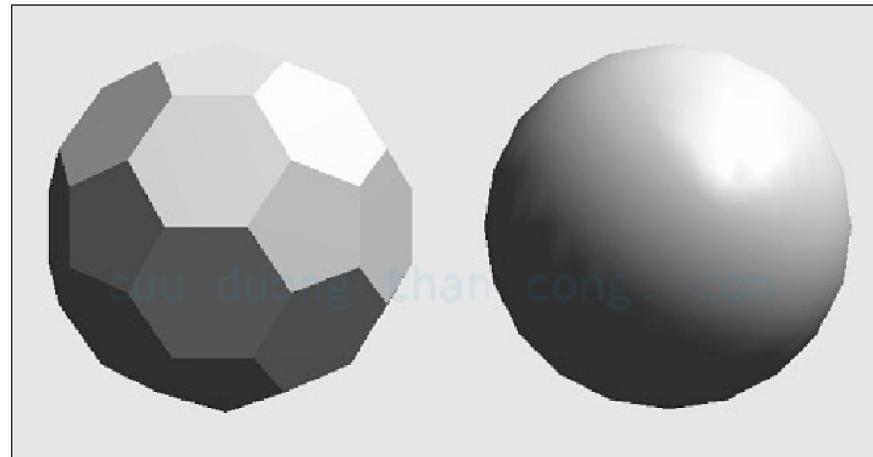
$$c(x+1) = c(x) + \frac{\text{color}_{\text{right}} - \text{color}_{\text{left}}}{x_{\text{right}} - x_{\text{left}}}$$

```
for (int y = ybott ; y <= ytop ; y++){
    find xleft and xright;
    find colorleft and colorright;
    colorinc = (colorright - colorleft)/ (xright - xleft)
    for (int x = xleft, c = colorleft; x <= xright; x++, c += colorinc ) {
        put c into the pixel at (x, y);}}
```

TÔ MÀU PHẲNG, TÔ MÀU TRƠN

Tô màu Gouraud:

- glShadeModel(GL_SMOOTH);

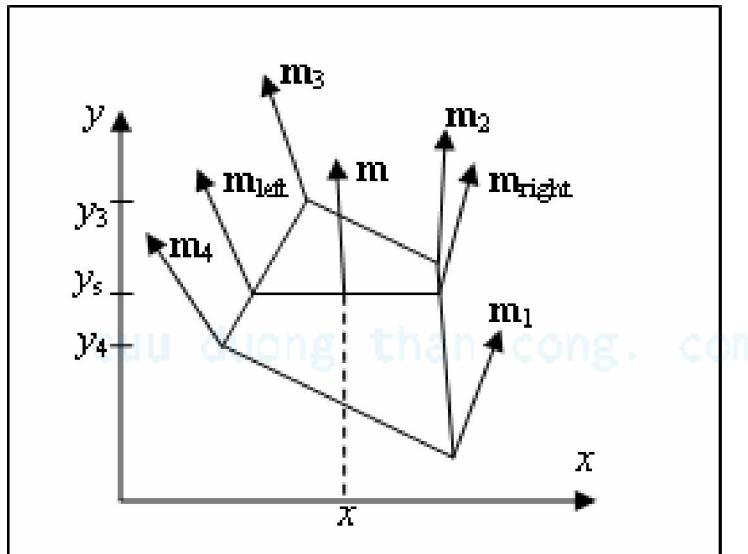


TÔ MÀU PHẲNG, TÔ MÀU TRƠN

☐ Tô màu Phong:

- chậm hơn nhiều so với phương pháp tô màu Gouraud
- OpenGL không hỗ trợ phương pháp tô màu này

$$m_{left} = lerp\left(m_4, m_3, \frac{y_s - y_4}{y_3 - y_4}\right)$$



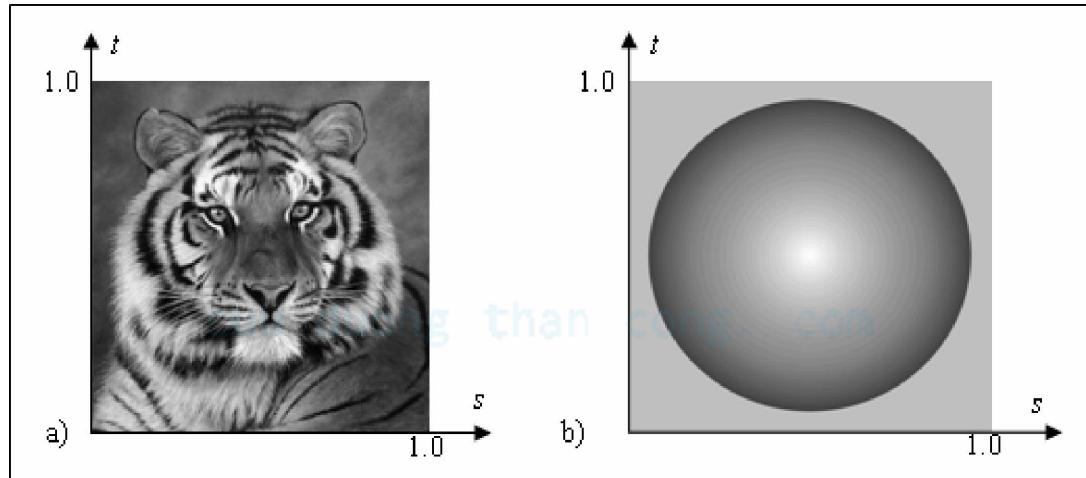
DÁN TEXTURE LÊN MẶT ĐA GIÁC



cuu duong than cong. com

DÁN TEXTURE LÊN MẶT ĐA GIÁC

- ❑ Có hai loại texture: texture bitmap và texture được tạo bằng giải thuật



```
Color3 texture(float s, float t){  
    return txtr[(int)(s*C)][(int)(t*R)];}  
  
float fakeSphere(float s, float t){  
    float r = sqrt((s - 0.5)*(s - 0.5) + (t - 0.5)*(t - 0.5));  
    if (r < 0.3) return 1 - r/0.3;  
    else return 0.2;}
```

DÁN TEXTURE LÊN MẶT ĐA GIÁC

- ❑ Các bước để dán texture lên bề mặt
 - đọc hoặc tạo dữ liệu texture vào mảng **textureData**
 - tạo đối tượng texture:
 - glGenTextures(1, &textureObject);
 - gán kiểu texture
 - glBindTexture(GL_TEXTURE_2D, textureObject);
 - gán số liệu cho đối tượng texture
 - glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, wid, height, 0, GL_RGB, GL_UNSIGNED_BYTE, textureData);

DÁN TEXTURE LÊN MẶT ĐA GIÁC

❑ Các bước để dán texture lên bề mặt

– lựa chọn texture:

- glBindTexture(GL_TEXTURE_2D, textureObject);

– gán tọa độ texture cho các đỉnh của đa giác

```
glBegin(GL_QUADS);
```

```
    glTexCoord2f(0.0, 0.0); glVertex3f(1.0, 2.5, 1.5);
```

```
    glTexCoord2f(0.0, 0.6); glVertex3f(1.0, 3.7, 1.5);
```

```
    glTexCoord2f(0.8, 0.6); glVertex3f(2.0, 3.7, 1.5);
```

```
    glTexCoord2f(0.8, 0.0); glVertex3f(2.0, 2.5, 1.5);
```

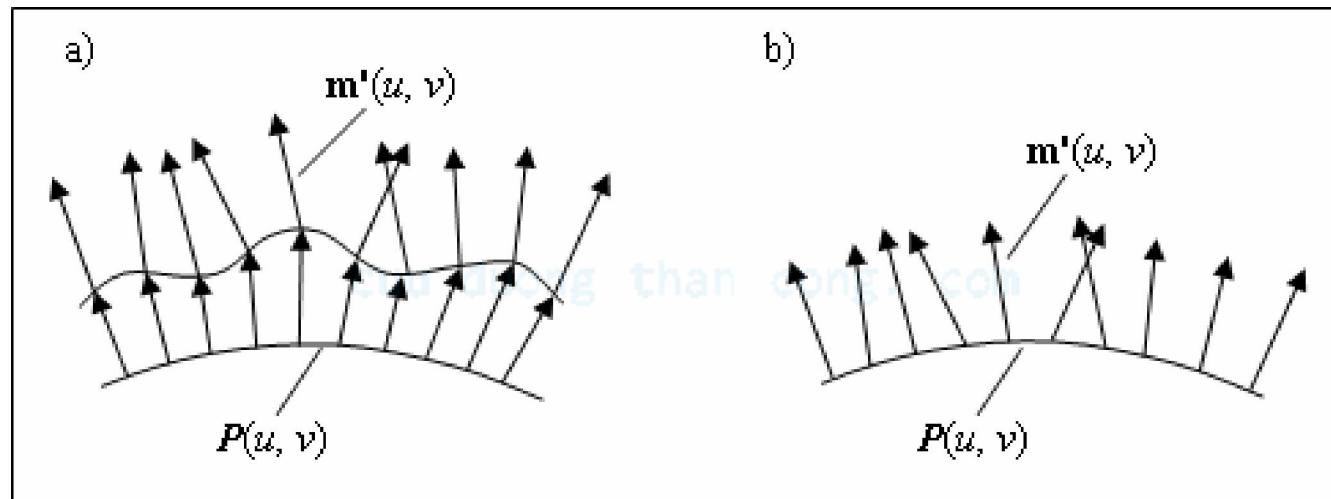
```
glEnd();
```

DÁN TEXTURE LÊN MẶT ĐA GIÁC

□ Ứng dụng của texture

- Tạo đối tượng phát sáng
- Điều chỉnh hệ số phản xạ
- Mô phỏng sự gồ ghề bằng phép ánh xạ Bump

cuu duong than cong. com



Trường Đại Học Bách Khoa TP Hồ Chí Minh
Khoa Khoa học & Kỹ thuật Máy tính



ĐỒ HỌA MÁY TÍNH

CHƯƠNG 9:

**KỸ THUẬT LẬP, ĐỀ
QUY ỨNG DỤNG TẠO
HOA VĂN**

NỘI DUNG TRÌNH BÀY

□ Giới thiệu

cuu duong than cong. com

cuu duong than cong. com

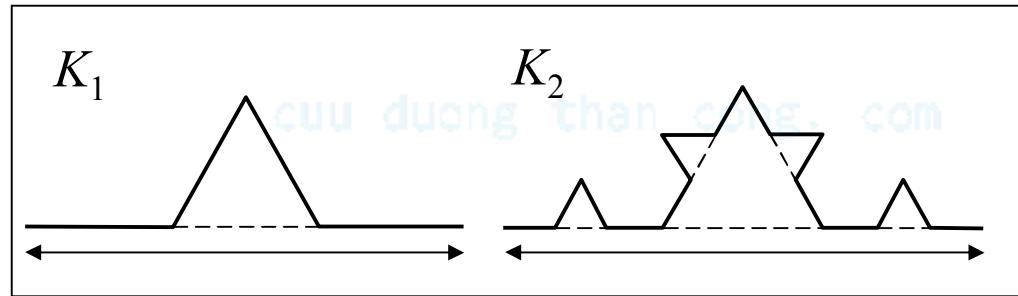
FRACTAL VÀ TÍNH TỰ TƯƠNG TỰ

- ❑ Tính tự tương tự: giống nhau ở mọi tỷ lệ
- ❑ Tính tự tương tự chính xác: hình ảnh sau khi phóng to giống hệt hình ảnh ban đầu
- ❑ Tính tự tương tự thống kê: mức độ bất quy tắc và lượn sóng chỉ giống nhau ở mức độ trung bình
- ❑ Ví dụ: đường bờ biển khi nhìn từ trên cao, sau đó tiến lại gần
- ❑ Fractal: các hình thức khác nhau của sự tự tương tự
- ❑ Các đường cong có chiều dài vô tận có chiều nằm giữa 1 và 2

FRACTAL VÀ TÍNH TỰ TƯƠNG TỰ

☐ Tinh chỉnh đường cong: đường cong Koch

- Tạo K_{n+1} từ K_n bằng cách chia mỗi đoạn của K_n thành 3 phần bằng nhau và thay đoạn giữa bằng một tam giác đều
- Chiều dài bằng $(4/3)^i$

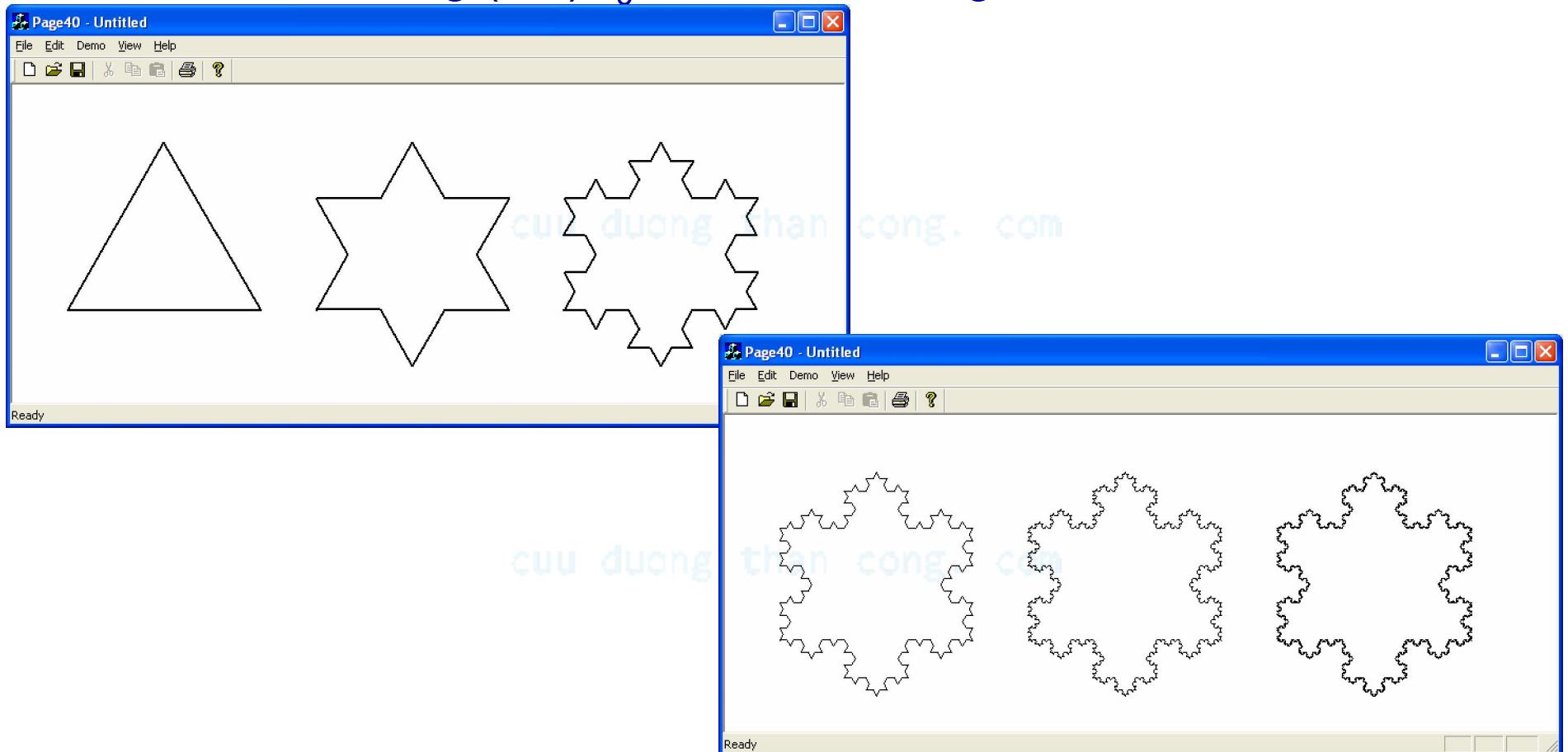


cuu duong than cong. com

FRACTAL VÀ TÍNH TỰ TƯƠNG TỰ

□ Bông tuyết Koch

- Chu vi bằng $3(4/3)^n \rightarrow$ vô cùng
- Diện tích bằng $(8/5)S_0$ khi $n \rightarrow$ vô cùng



FRACTAL VÀ TÍNH TỰ TƯƠNG TỰ

❑ Vẽ đường cong Koch

Vẽ Kn:

```
if (n bằng 0)
    Vẽ một đoạn thẳng;
else
{
    Vẽ Kn-1
    Quay trái 600
    Vẽ Kn-1
    Quay phải 1200
    Vẽ Kn-1
    Quay trái 600
}
```

SINH CHUỖI VÀ TẠO ĐƯỜNG CONG PEANO

- Dùng chuỗi để điều khiển con rùa
 - 'F' có nghĩa là forward(L, l) (đi theo hướng hiện hành một khoảng L, có vẽ đoạn thẳng)
 - '+' có nghĩa là turn(A) (quay phải một góc A độ)
 - '-' có nghĩa là turn(-A) (quay trái một góc A độ)

'F-F++F-F', với góc A bằng 60^0 \rightarrow vẽ Koch bậc 1

- Từ chuỗi 'F-F++F-F' chuyển thành chuỗi phức tạp?
 - dựa vào tập luật sinh
 - 'F' \rightarrow "F-F++F-F" (luật của đường cong Koch)

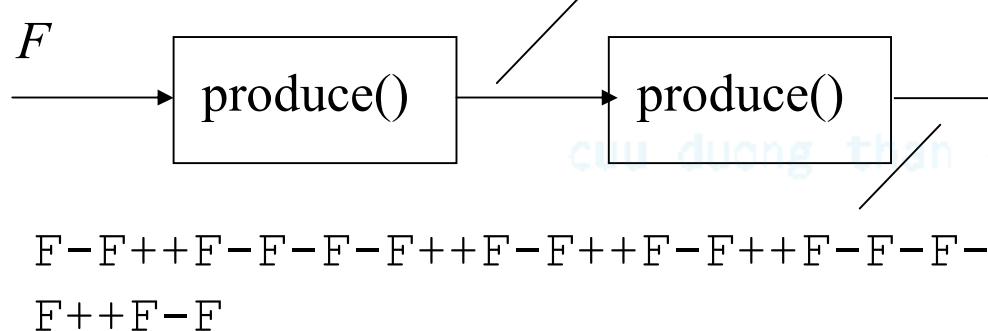
SINH CHUỖI VÀ TẠO ĐƯỜNG CONG PEANO

- ❑ Chuỗi ban đầu: F
- ❑ Thế hệ thứ nhất: $S_1 = "F-F++F-F"$
- ❑ Thế hệ thứ 2: $S_2 = "F-F++F-F-F-F-F++F-F++F-F-F-F++F-F"$
- ❑ for (each character ch in the input file)
 if (ch == '+' || ch == '-') write it to the output file;
 else if (ch == 'F') write "F-F++F-F" to the output file;
- ❑ for (each character ch in the input file)
 if (ch == '+') turn(A);
 else if (ch == '-') turn(-A);
 else if (ch == 'F') forward(1, 1);

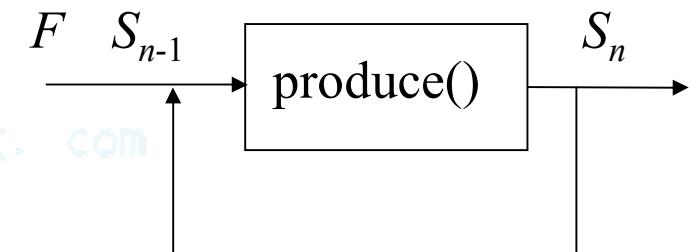
SINH CHUỖI VÀ TẠO ĐƯỜNG CONG PEANO

a) Hai giai đoạn của quá trình sinh chuỗi

$F - F + + F - F$



b) Hệ thống hàm lặp



SINH CHUỖI VÀ TẠO ĐƯỜNG CONG PEANO

□ Mở rộng ngôn ngữ

- 'F' → 'F'
- 'X' → "X+YF+" ← tập luật sinh đường cong con rồng
- 'Y' → "-FX-Y"
- atom = "FX"

□ Bắt đầu với nguyên tử FX, chúng ta có

- $S_1 = "FX+YF+"$
- $S_2 = "FX+YF++-FX-YF+"$
- X, Y bỏ qua khi vẽ
- F, +, - vẫn được thông dịch như trước đây.

SINH CHUỖI VÀ TẠO ĐƯỜNG CONG PEANO

- Sinh chuỗi đệ quy và thao tác vẽ

```
void produceString(char *st, int order){  
    for(;st ; st++)  
        switch(*st) {  
            case '+': CD -= angle; break;  
            case '-': CD +=angle; break;  
            case 'F': if (order > 0) produceString(Fstr, order - 1);  
                        else forward(length, 1);  
                        break;  
            case 'X': if (order > 0) produceString(Xstr, order - 1);  
                        break;  
            case 'Y': if (order > 0) produceString(Ystr, order - 1);  
                        break;}}  
    
```

SINH CHUỖI VÀ TẠO ĐƯỜNG CONG PEANO

❑ Cho phép rẽ nhánh:

'[': saveTurtle() → lưu trạng thái hiện tại của con rùa

❑ Trạng thái của con rùa = {CP, CD}

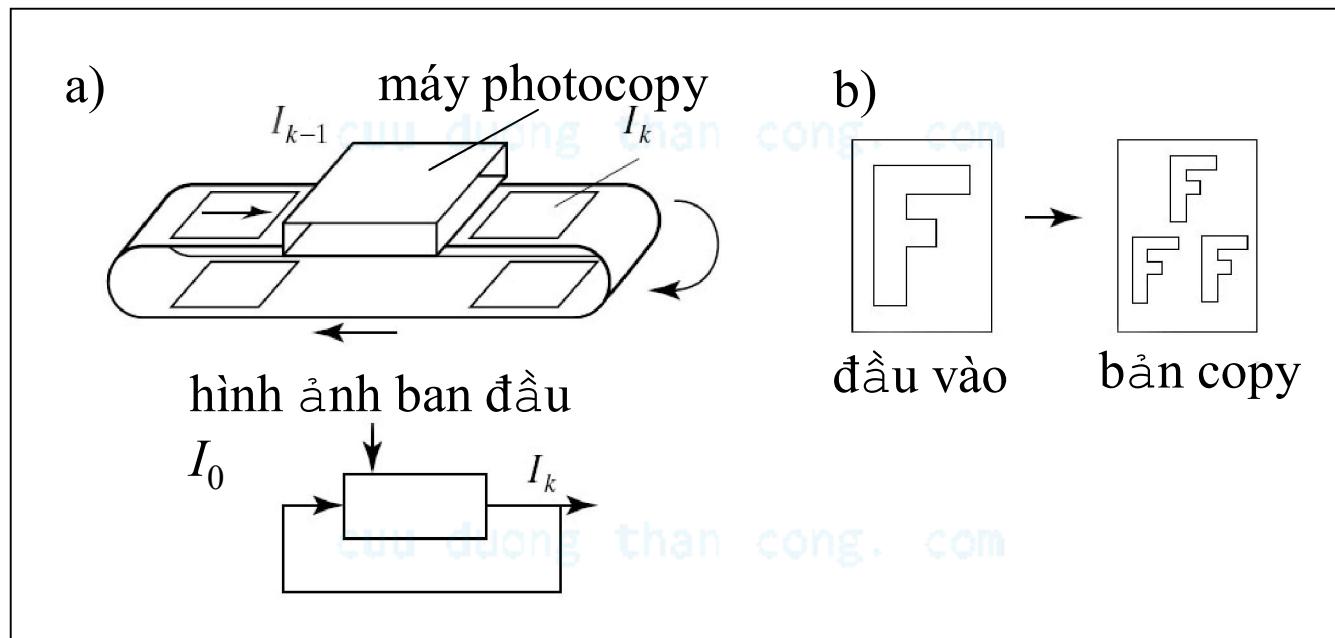
❑ thêm các lệnh sau vào produceString():

'[': saveTurtle();break; ← đẩy trạng thái hiện hành của con rùa vào ngăn xếp.

SỬ DỤNG HỆ HÀM LẶP TẠO HÌNH ẢNH

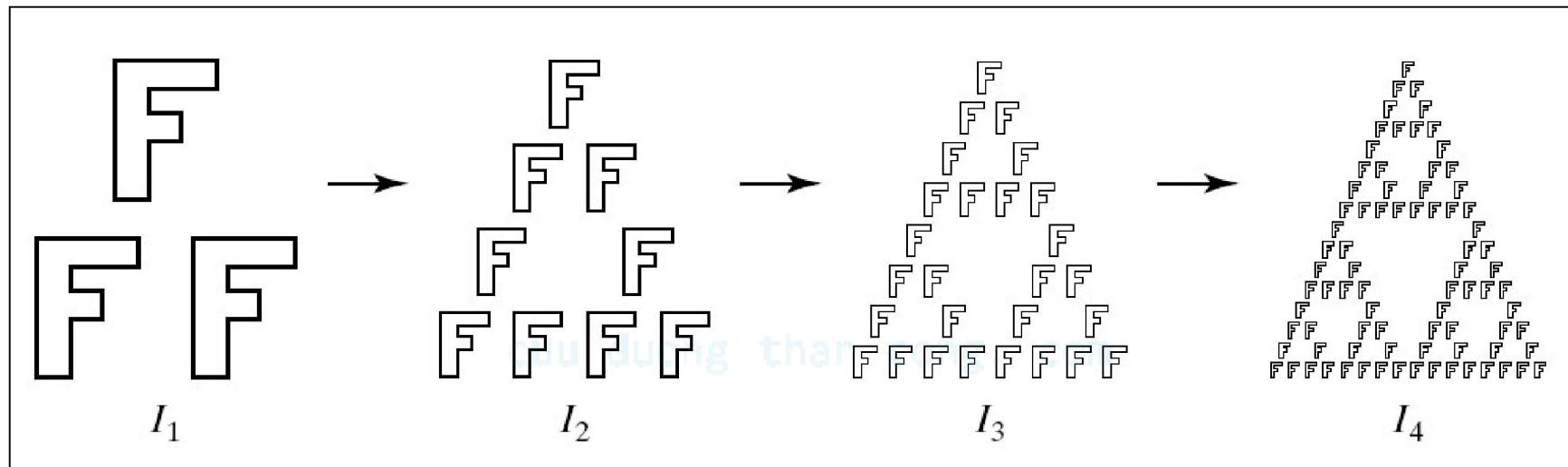
☐ Máy photocopy thực nghiệm

- Các hình ảnh ở đầu ra có hội tụ về hình ảnh nào không?



SỬ DỤNG HỆ HÀM LẬP TẠO HÌNH ẢNH

☐ Máy photocopy Sierpinski



-Hình ảnh hội tụ về tam giác Sierpinski và không phụ thuộc vào hình ảnh ban đầu

-Gồm 3 thấu kính mỗi thấu kính thực hiện ba phép biến đổi cho chữ F để nhận được 3 chữ F mới

SỬ DỤNG HỆ HÀM LẬP TẠO HÌNH ẢNH

❑ Máy photocopy Sierpinski

- Các phép biến đổi là

$$M_1 = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad M_2 = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} \quad M_3 = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

cuuduongthancong.com

Để tiện lợi, chúng ta liệt kê các phần tử của ma trận dưới dạng danh sách

$$T = \{m_{11}, m_{12}, m_{21}, m_{22}, m_{13}, m_{23}\}$$

SỬ DỤNG HỆ HÀM LẶP TẠO HÌNH ẢNH

□ Lý thuyết của quá trình copy

- hình ảnh nhập $I =$ tập các điểm màu đen $= \{(x, y) \text{ sao cho } (x, y) \text{ được tô màu đen}\}$
- hình ảnh xuất $= T_1(I) \cup T_2(I) \cup T_3(I)$
- tổng quát hóa $W(\cdot) = T_1(\cdot) \cup T_2(\cdot) \cup T_3(\cdot)$
- khi lặp đi lặp lại thao tác đưa hình ảnh xuất vào ngõ nhập thì quỹ đạo của $I_0, I_1, I_2 \dots$ sẽ hội tụ về một hình ảnh gọi là nhân tố hấp dẫn A
 - $W(A) = A$
 - A không phụ thuộc vào hình ảnh ban đầu

SỬ DỤNG HỆ HÀM LẮP TẠO HÌNH ẢNH

❑ Thực hiện thao tác vẽ ở lần lặp thứ k

- I_0 là một đường gấp khúc
- I_0 là một điểm

❑ Ví dụ: vẽ lá cây dương xỉ

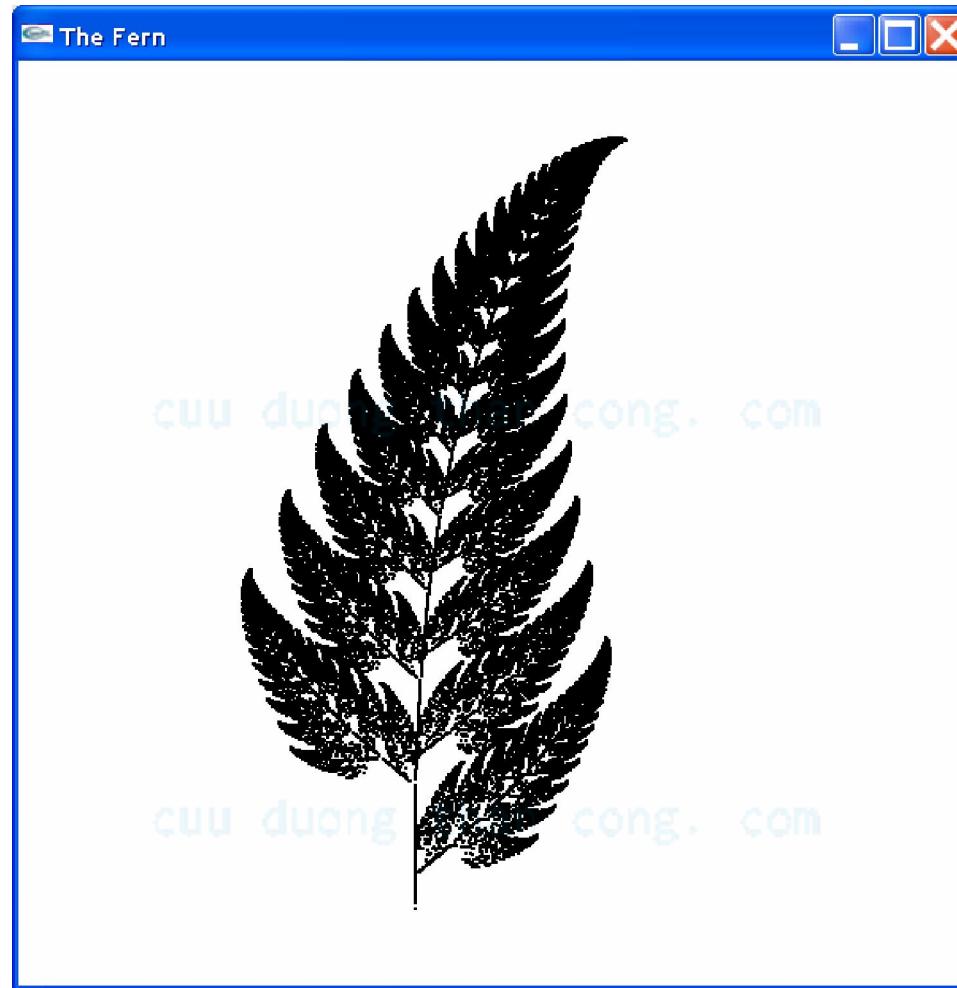
- $T1 = \{0, 0, 0, .16, 0, 0\};$
- $T2 = \{.2, .23, -.26, .22, 0, 1.6\};$
- $T3 = \{- .15, .26, .28, .24, 0, .44\};$
- $T4 = \{.85, -.04, .04, .85, 0, 1.6\};$

cuu duong than cong. com

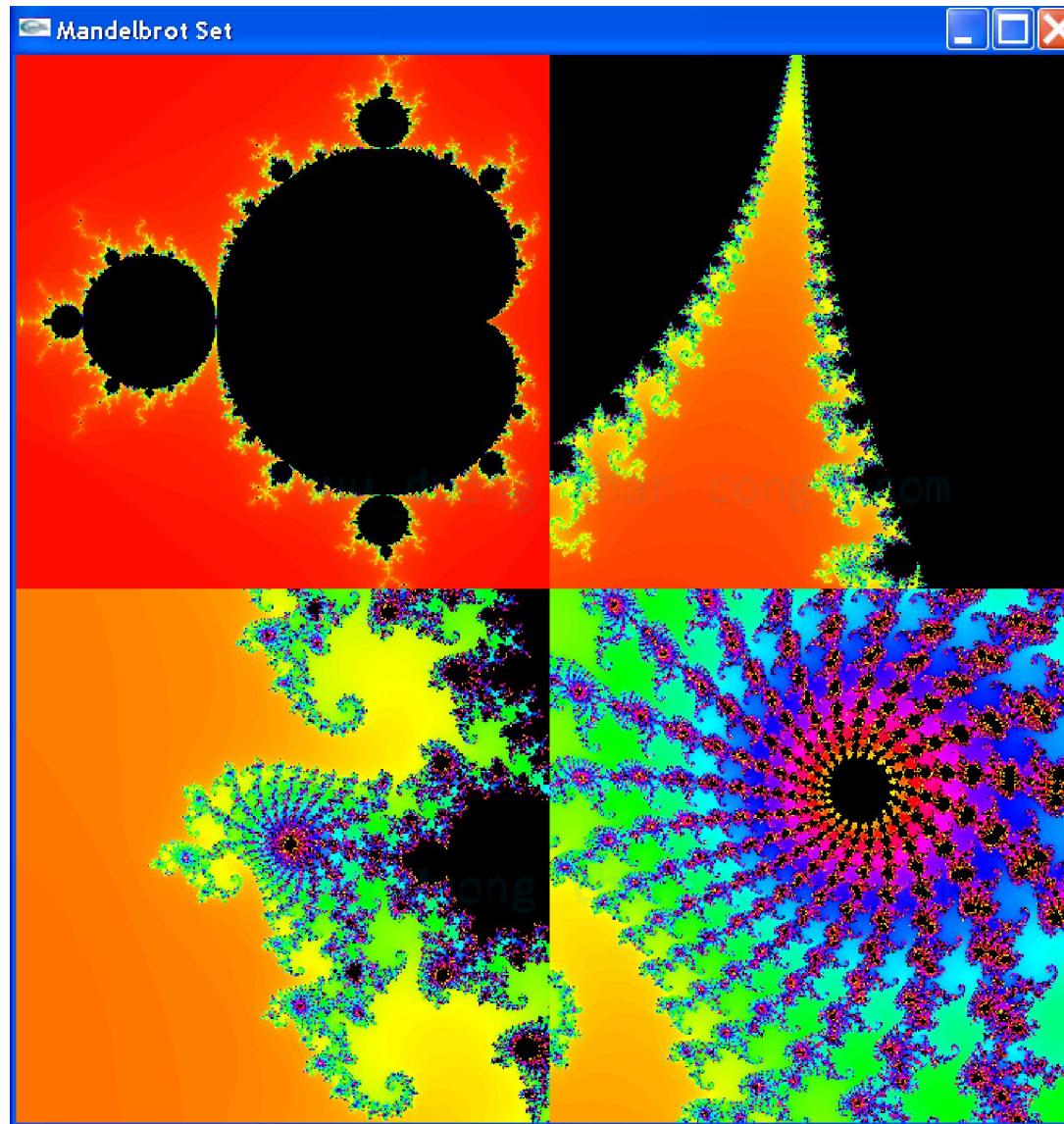
SỬ DỤNG HỆ HÀM LẬP TẠO HÌNH ẢNH

```
void superCopier(RealPolyArray pts, int k)
{
    int i, j;
    RealPolyArray newpts;
    if(k == 0) drawPoints(pts);
    else for(i = 1; i <= N; i++)
    {
        newpts.num = N*pts.num;
        for(j = 0; j < newpts.num; j++)
            transform(affines[i], pts.pt[j], newpts.pt[j]);
        superCopier(newpts, k-1);
    }
}
```

SỬ DỤNG HỆ HÀM LẬP TẠO HÌNH ẢNH

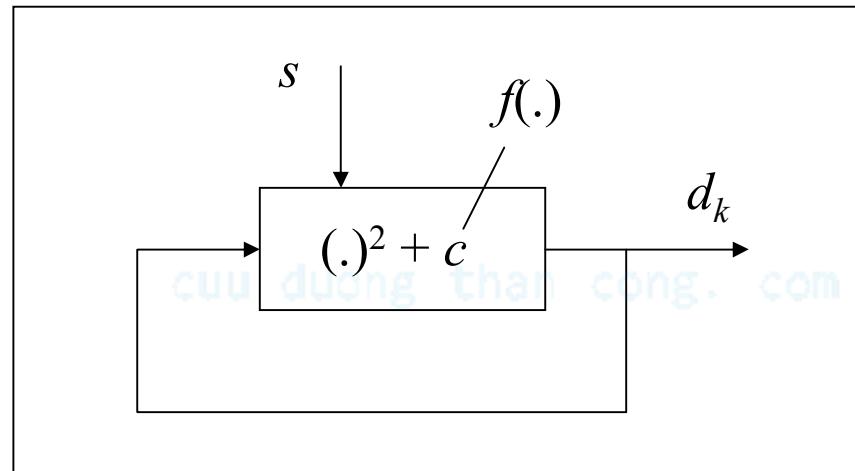


TẬP MANDELBROT



TẬP MANDELBROT

- $f(z) = z^2 + c$
- với giá trị ban đầu s , ta có quy đạo
 - $d_1 = (s)^2 + c$
 - $d_2 = ((s)^2 + c)^2 + c$
 - $d_3 = (((s)^2 + c)^2 + c)^2 + c$
 - $d_4 = ((((s)^2 + c)^2 + c)^2 + c)^2 + c$



TẬP MANDELBROT

- ❑ s và c có thể là số phức
- ❑ quỹ đạo hội tụ và quỹ đạo phân kỳ
- ❑ điểm cố định của hệ thống thỏa mãn

– $f(z) = z \rightarrow z^2 + c = z$

$$p_+, p_- = \frac{1}{2} \pm \sqrt{\frac{1}{4} - c}$$

cuu duong than cong. com

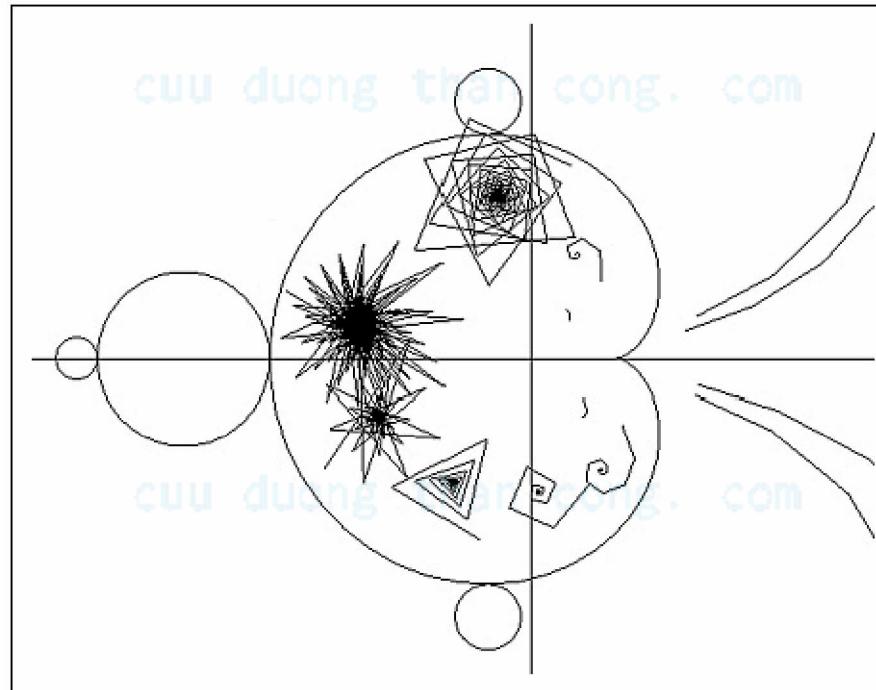
- ❑ điểm cố định hấp dẫn, điểm cố định đầy

cuu duong than cong. com

TẬP MANDELBROT

□ Định nghĩa:

- với s luôn bằng 0
- điểm c nằm trong M nếu quỹ đạo của 0 hội tụ
- điểm c nằm ngoài M nếu quỹ đạo của 0 phân kỳ



TẬP MANDELBROT

☐ c có nằm trong tập M hay không?

d_k vượt qua giá trị 2 thì quỹ đạo phân kỳ

thời gian để d_k vượt qua giá trị 2 được gọi là thời gian cư trú

c nằm trong M có thời gian cư trú rất lớn (Cận trên là Num)

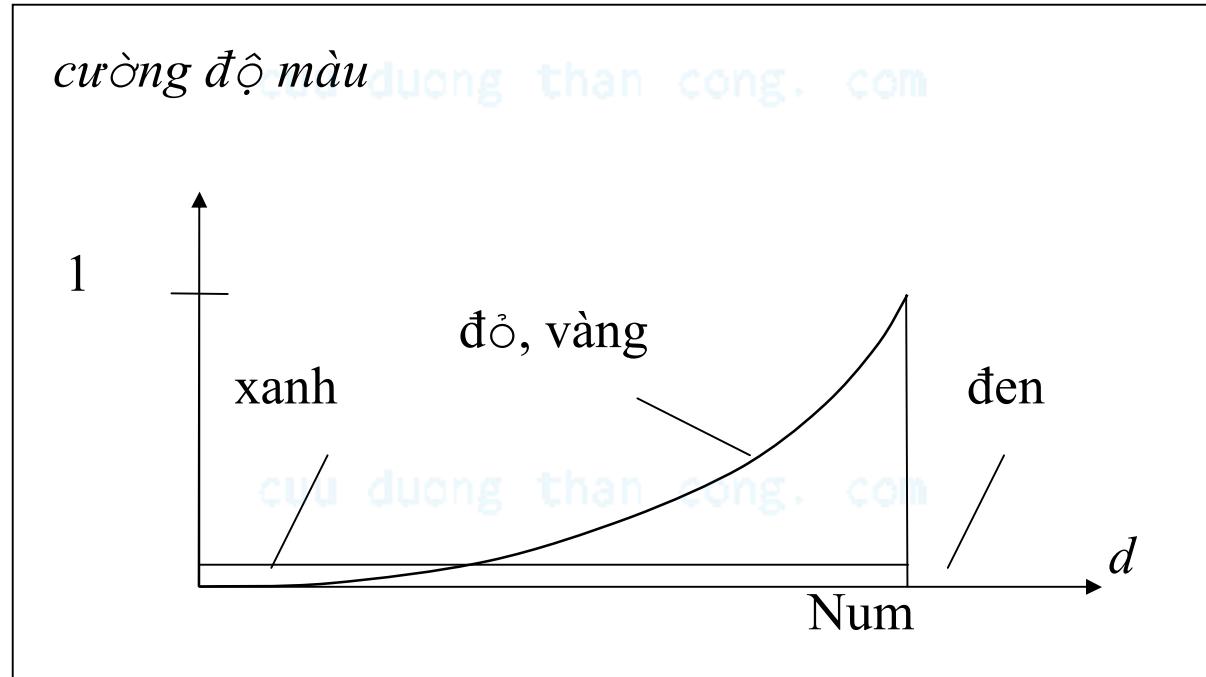
```
int dwell(double cx, double cy){  
    #define Num 100  
    double tmp, dx = cx, dy = cy, fsq = cx * cx + cy * cy;  
    for(int count = 0; count <= Num && fsq <= 4; count++)  
    {  
        tmp = dx;  
        dx = dx * dx - dy * dy + cx;  
        dy = 2.0 * tmp * dy + cy;  
        fsq = dx * dx + dy * dy;  
    }  
    return count;  
}
```

$$(x + yi)^2 = (x^2 - y^2) + (2xy)i,$$

TẬP MANDELBROT

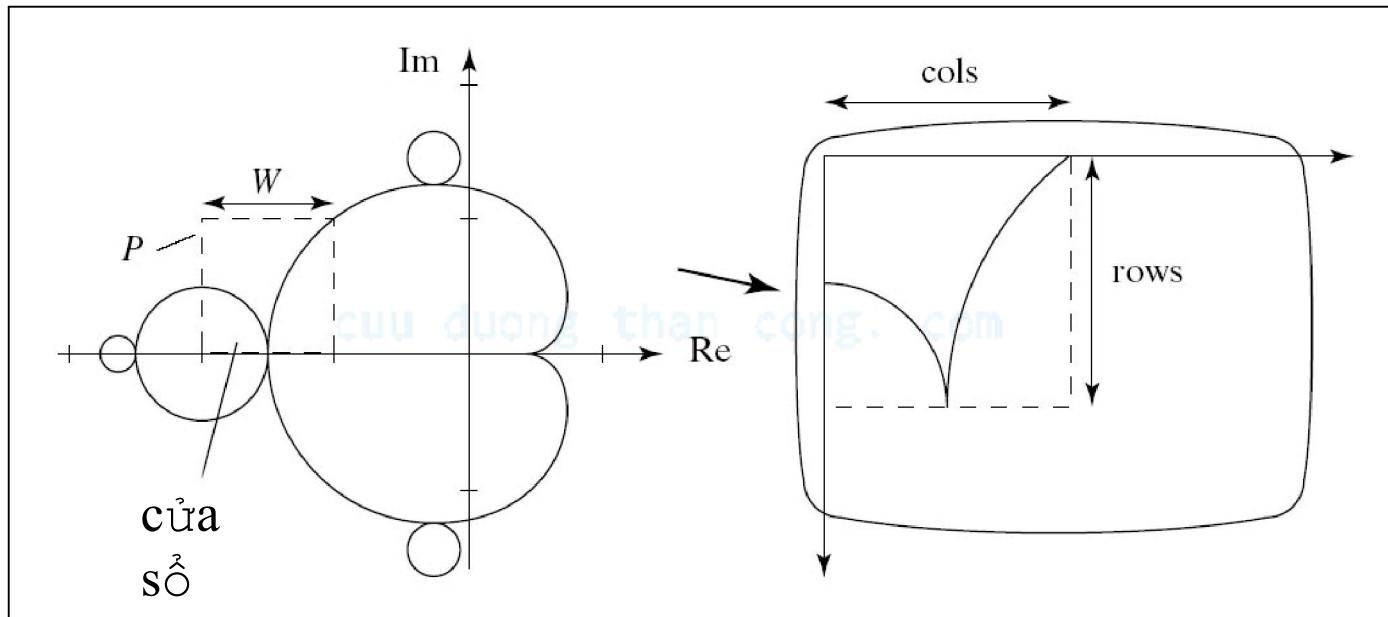
□ Xác định màu cho điểm c

- c nằm trong M: tô bằng màu đen
- c nằm ngoài M: float $v = d/(\text{float}) \text{ Num};$
 $\text{glColor3f}(v*v, v*v, 0.2);$



TẬP MANDELBROT

- ❑ Xác định sự tương ứng giữa pixel trên màn hình với số phức c



$$c_{ij} = \left(P_x + \frac{i + \frac{1}{2}}{cols} W, P_y - \frac{j + \frac{1}{2}}{rows} W \right)$$

TẬP MANDELBROT

```
for( j = 0; j < rows; j++)  
    for(i = 0; i < cols; i++)  
    {
```

*tìm số phức c tương ứng với pixel ở vị trí i, j
xác định thời gian cư trú của quỹ đạo
dựa trên thời gian cư trú xác định màu sắc tương ứng*

```
        setPixel(i, j, color);  
    }
```

cuu duong than cong. com

TẬP JULIA

- ❑ cho c cố định, khảo sát các giá trị khác nhau của s
- ❑ Tập Julia đầy đủ: là tập hợp tất cả các điểm khởi đầu s có quỹ đạo hội tụ.
- ❑ vẽ tập Julia

```
for( j = 0; j < rows; j++)  
for(i = 0; i < cols; i++){
```

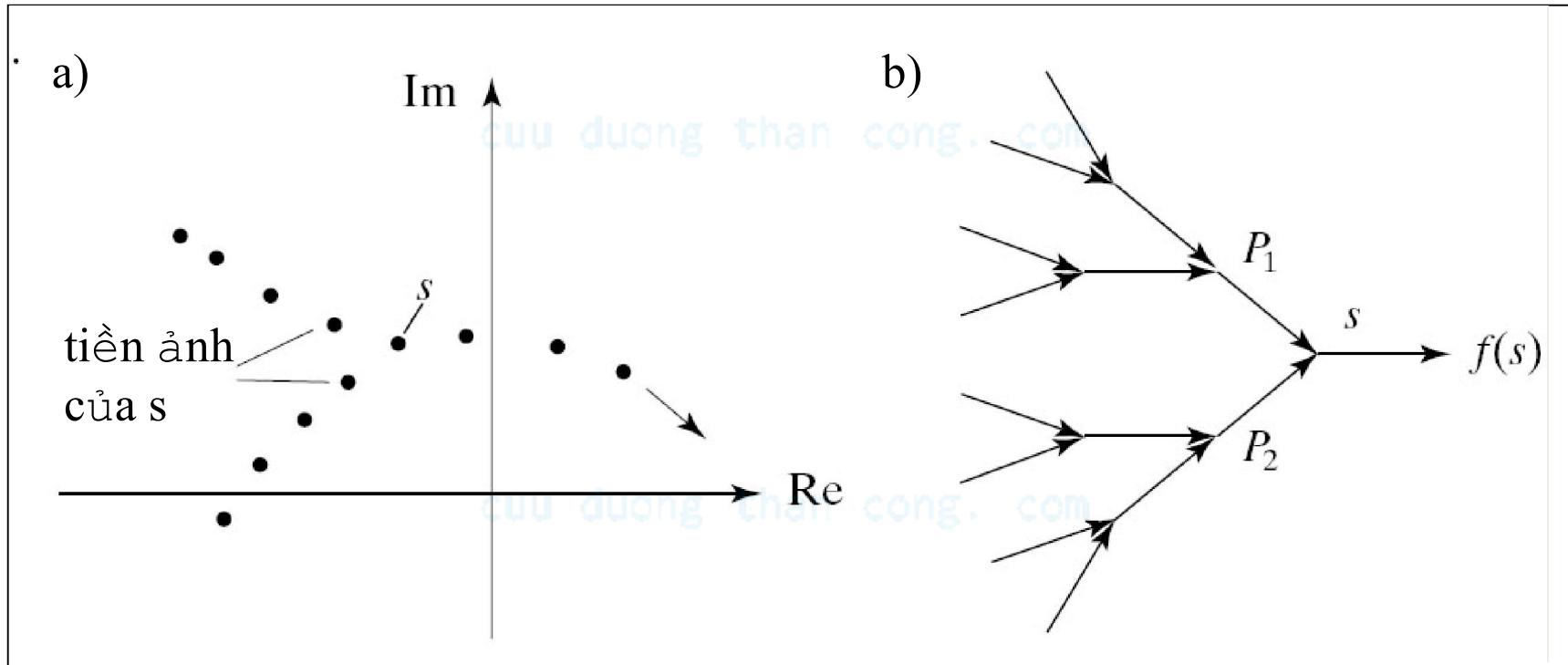
*tìm số phức s tương ứng với pixel ở vị trí i, j
xác định thời gian cư trú của quỹ đạo
dựa trên thời gian cư trú xác định màu sắc tương ứng*

```
setPixel(i, j, color);}
```

- ❑ phải truyền cả s và c cho hàm dwell()

TẬP JULIA

- Tập Julia J_c : là đường biên của tập Julia đầy đủ K_c
- Tiền ảnh của z : $\pm \sqrt{z - c}$
- Các tiền ảnh của z có quỹ đạo giống như quỹ đạo của z



TẬP JULIA

❑ Vẽ tập Julia

do{

 if(coin flip is head) $z = +\sqrt{z - c}$

 else $z = -\sqrt{z - c}$

 draw dot at z;

}while (not bored);

❑ Tính căn bậc hai của $z = x + iy$

$$\sqrt{z} = a + ib \quad , y \geq 0 \quad a = \sqrt{\frac{|z| + x}{2}} \quad b = \sqrt{\frac{|z| - x}{2}}$$
$$\sqrt{z} = -a + ib \quad , y < 0 \quad a = \sqrt{\frac{|z| + x}{2}} \quad b = \sqrt{\frac{|z| - x}{2}}$$