Đồ họa

Tuần 5

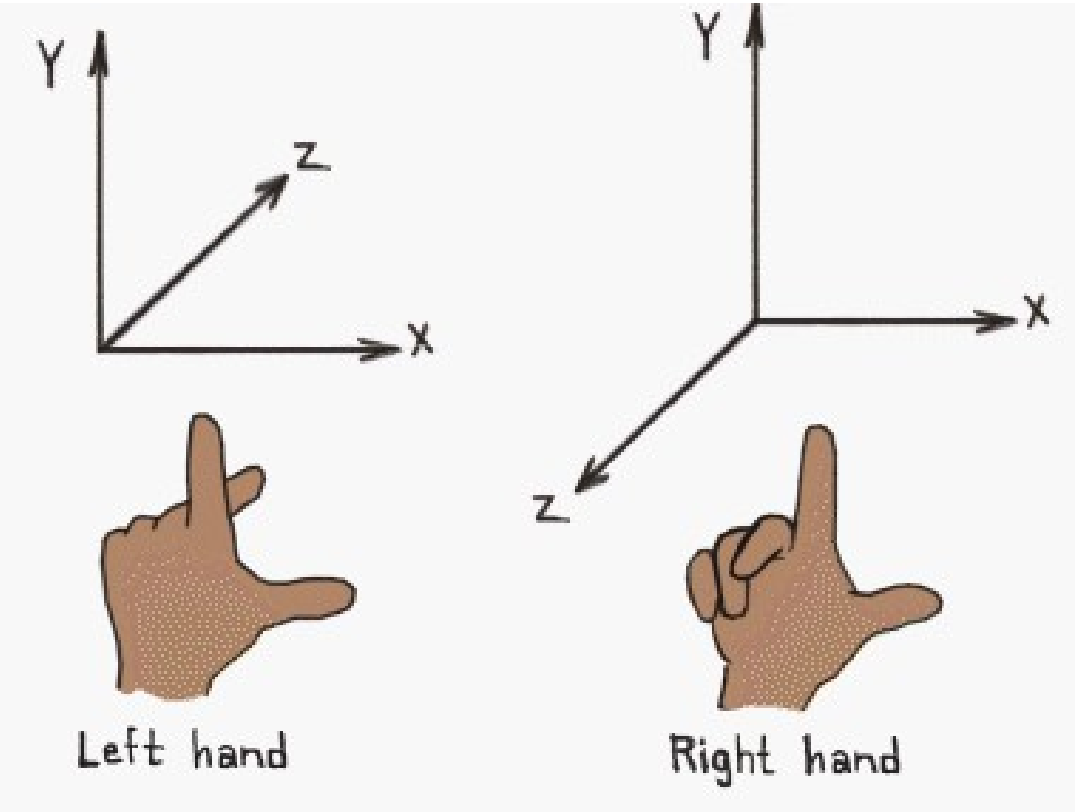
Giảng viên: Trần Đức Minh

Nội dung bài giảng

- Trục tọa độ 3D trên WebGL
- Biến đổi tọa độ 3D
- Xác định mặt trong và mặt ngoài của đa giác
- Cấu trúc dữ liệu xây dựng mô hình 3D
- Dựng hình 3D
- Các phép biến đổi hình 3D
- Các ví dụ

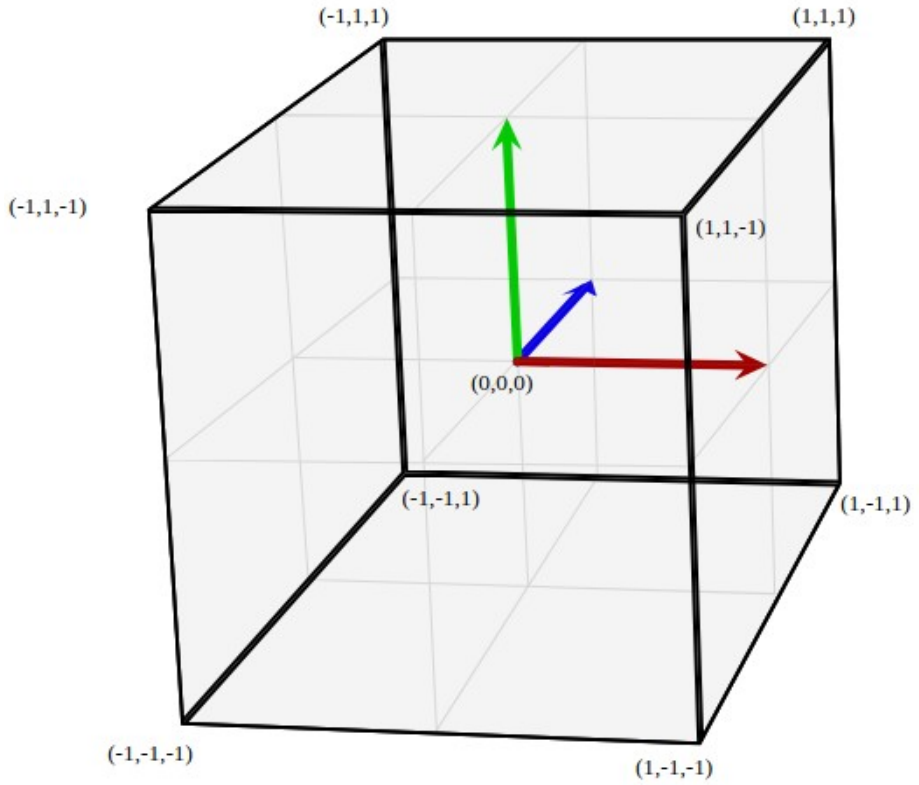
Trục tọa độ 3D trên WebGL

- Trục tọa độ của WebGL
  - Theo quy tắc bàn tay trái



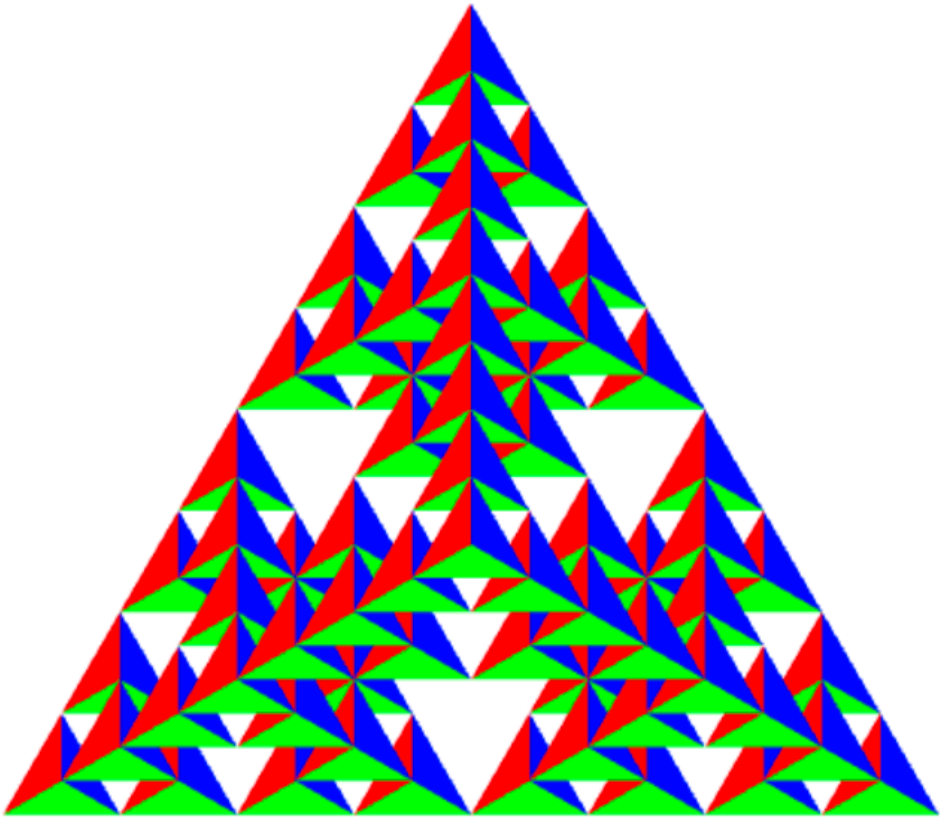
Trục tọa độ 3D trên WebGL

- Miền giá trị của các trục tọa độ 3D



Ví dụ 1

- Vẽ hình chóp Sierpinski

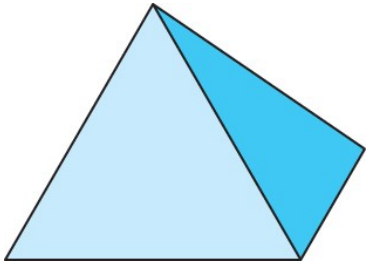


Ví dụ 1

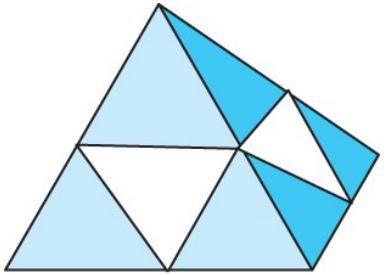
- Vẽ hình chóp Sierpinski
  - Tính toán tọa độ đỉnh của 4 tam giác nhỏ tạo nên hình chóp, sau đó nạp dần dần cứ 3 đỉnh một lần tương ứng với 3 đỉnh của tam giác nhỏ vào mảng dữ liệu đỉnh.
  - Tất cả các đỉnh của tam giác đều được xác định bởi tọa độ trên không gian 3 chiều

Ví dụ 1

- Ý tưởng thuật toán
  - Bắt đầu với một hình chóp



  - Tính toán trung điểm các cạnh của hình chóp, sau đó vẽ các hình chóp mới bằng cách kết hợp các đỉnh và các trung điểm vừa tính toán.



  - Cứ lặp lại như vậy đối với các hình chóp ở bên trong cho đến n lần thì dừng.

Ví dụ 1

- Vẽ hình chóp Sierpinski
  - Thuật toán (xây dựng hàm đệ quy)

chiaHinhChop(đỉnh A, đỉnh B, đỉnh C, đỉnh D, số lần lặp) {

Nếu (số lần lặp = 0) **Thì**

veHinhChop(đỉnh A, đỉnh B, đỉnh C, đỉnh D);

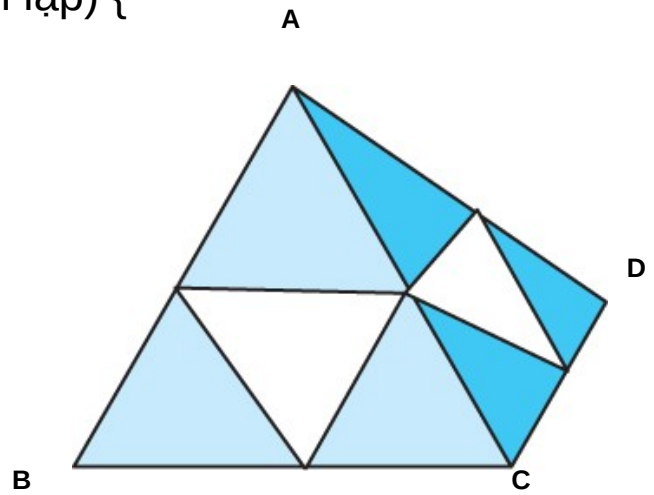
Ngược lại

trung điểm AB = tinhTrungDiem(đỉnh A, đỉnh B);
trung điểm AC = tinhTrungDiem(đỉnh A, đỉnh C);
trung điểm AD = tinhTrungDiem(đỉnh A, đỉnh D);
trung điểm BC = tinhTrungDiem(đỉnh B, đỉnh C);
trung điểm BD = tinhTrungDiem(đỉnh B, đỉnh D);
trung điểm CD = tinhTrungDiem(đỉnh C, đỉnh D);

số lần lặp = số lần lặp – 1;

chiaHinhChop(đỉnh A, trung điểm AB, trung điểm AC, trung điểm AD, số lần lặp);
chiaHinhChop(đỉnh B, trung điểm AB, trung điểm BC, trung điểm BD, số lần lặp);
chiaHinhChop(đỉnh C, trung điểm AC, trung điểm BC, trung điểm CD, số lần lặp);
chiaHinhChop(đỉnh D, trung điểm AD, trung điểm BD, trung điểm CD, số lần lặp);

}



Ví dụ 1

- Vẽ hình chóp Sierpinski
  - Thuật toán

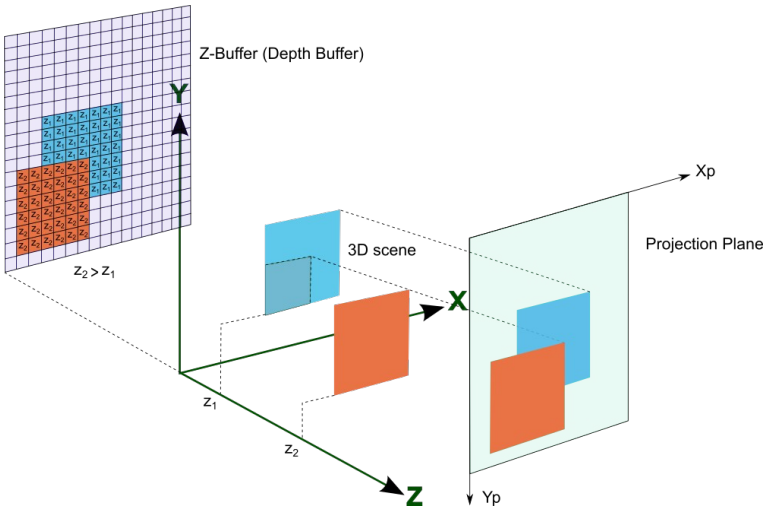
```
veHinhChop(đỉnh A, đỉnh B, đỉnh C, đỉnh D) {
    veTamGiac(đỉnh A, đỉnh B, đỉnh C);
    veTamGiac(đỉnh A, đỉnh B, đỉnh D);
    veTamGiac(đỉnh A, đỉnh C, đỉnh D);
    veTamGiac(đỉnh B, đỉnh C, đỉnh D);
}
```

Ví dụ 1

- Vẽ hình chóp Sierpinski
  - Khởi tạo một **mảng V** được dùng để chứa đỉnh của tất cả các hình chóp.
    - Cứ mỗi khi gặp số lần lặp = 0 thì hàm **veHinhChop** sẽ gọi đến các hàm **veTamGiac** để đưa toàn bộ các đỉnh vẽ tam giác ở bên trong hình chóp được tạo lập cuối cùng vào mảng V này.
  - Sau khi hàm đệ quy **chiaHinhChop** thực hiện xong
    - Thực hiện đưa toàn bộ các đỉnh cần vẽ vào Vertex Shader để bắt đầu thực hiện vẽ các tam giác lên màn hình.

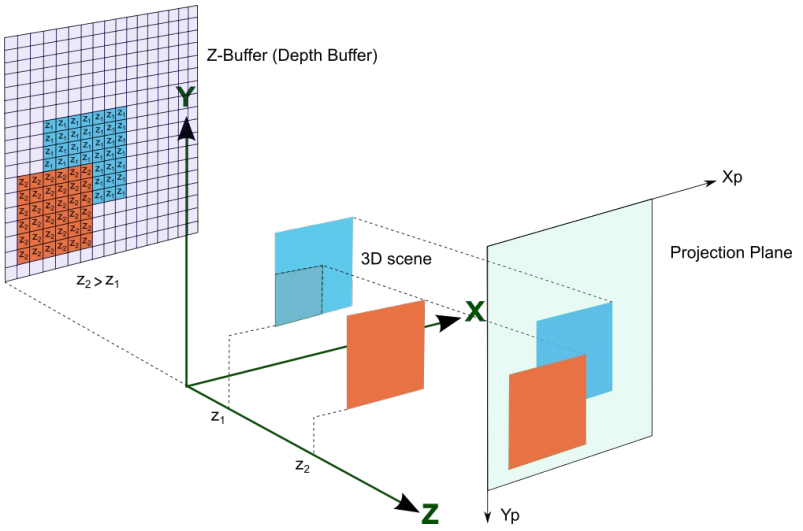
Ví dụ 1

- Hàm bổ trợ
  - **gl.enable(gl.DEPTH_TEST)**: Cho phép kích hoạt kiểm tra bộ đệm chiều sâu, mục đích là để có thể thấy được tạo hình 3D.
    - Khi kiểm tra bộ đệm chiều sâu được kích hoạt, WebGL thực hiện kiểm tra giá trị độ sâu (trục z) của fragment so với nội dung của bộ đệm chiều sâu
      - Nếu vượt qua, fragment sẽ được hiển thị và bộ đệm chiều sâu được cập nhật với giá trị mới.
      - Nếu không vượt qua, fragment sẽ bị loại bỏ và không hiển thị lên màn hình.
      - Ngoài ra, đều một đối tượng đứng sau một đối tượng khác thì đối tượng đứng sau có thể bị loại bỏ toàn bộ hoặc chỉ hiển thị một phần.



Ví dụ 1

- Hàm bổ trợ

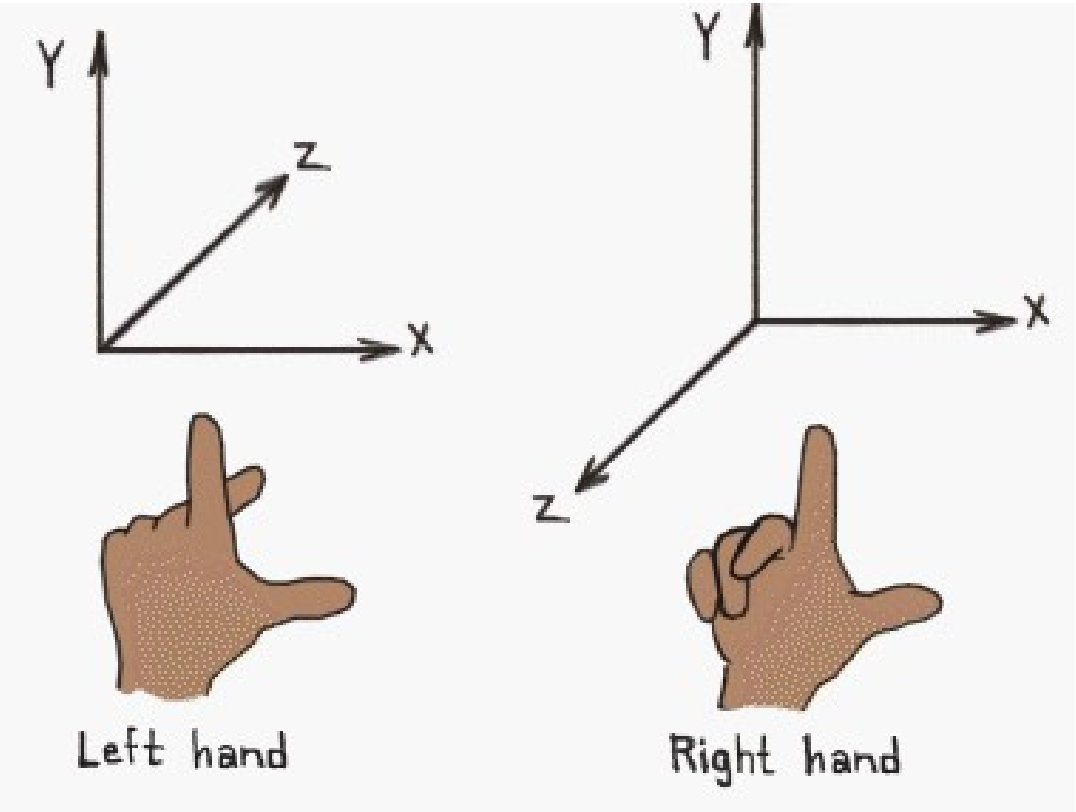


  - **gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT):**
    - Được sử dụng để xóa màn hình với màu xóa cho trước (xác định bởi hàm gl.clearColor(); đồng thời xóa toàn bộ bộ đệm chiều sâu trước mỗi khung hình nhằm tránh sử dụng bộ đệm chiều sâu cũ._

- ***Chú ý***:
  - Thực hiện Đổi màu cho các mặt của hình chóp.

Trục tọa độ 3D của người sử dụng

- Trục tọa độ của người sử dụng
  - Theo quy tắc bàn tay phải



Biến đổi tọa độ 3D

- Cần **biến đổi** từ trục tọa độ người sử dụng sang trục toạ độ WebGL **để hiển thị thông tin** lên màn hình.
- Công thức:

$x_{webgl} = x_{user} / width;$

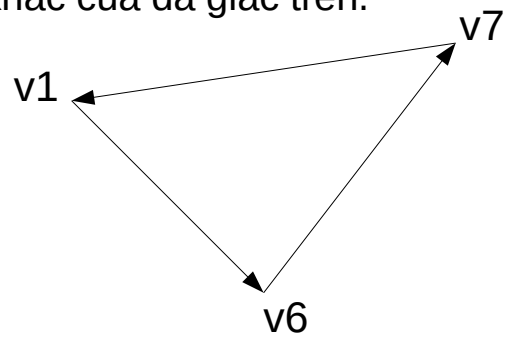
$y_{webgl} = y_{user} / height;$

$z_{webgl} = -z_{user} / depth;$

Với **width**, **heigth**, **depth** là **chiều rộng, chiều cao, chiều sâu** tối đa được thiết lập trên trục tọa độ của người sử dụng.

Mặt trong và mặt ngoài của đa giác

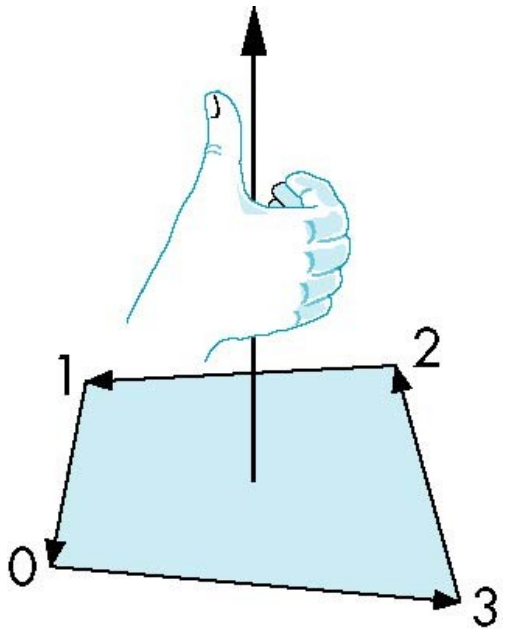
- Tất cả khối hình học 3D trong WebGL đều được tạo bởi một tập hợp các đa giác, trong đó mỗi đa giác đều thuộc một mặt phẳng nào đó, do đó các đa giác đều tồn tại 2 mặt được gọi là **mặt trong** và **mặt ngoài**.
- Khi WebGL vẽ đa giác, mặt trong hay mặt ngoài của đa giác được thể hiện bởi **thứ tự của các đỉnh**.
  - Ví dụ:
    - Các tập hợp đỉnh {v1, v6, v7} hay {v6, v7, v1} hay {v7, v1, v6} là tương đương và đều thể hiện cùng một mặt của một đa giác.
    - Tuy nhiên tập hợp đỉnh {v1, v7, v6} thì lại thể hiện mặt khác của đa giác trên.



- WebGL có thể **xử lý mặt trong, mặt ngoài của một đa giác là khác nhau**
  - Ví dụ: Trong một khối 3D, mặt trong sẽ bị ẩn đi do bị che khuất còn của mặt ngoài thì hiện ra.

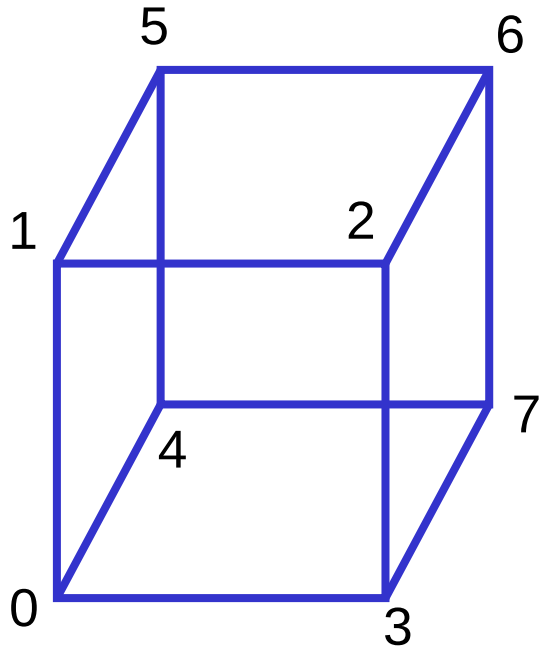
Mặt trong và mặt ngoài của đa giác

- Sử dụng **quy tắc bàn tay phải** để xác định thứ tự tập đỉnh cho **mặt ngoài** của đa giác
  - Để ngón trỏ của bàn tay phải về hướng vuông góc với mặt phẳng bạn coi đó là mặt ngoài.
  - Khum, chụm và tạo độ cong 4 ngón tay còn lại. Hướng của 4 ngón tay, chính là hướng đi thứ tự của các đỉnh.



Mặt trong và mặt ngoài của đa giác

- Ví dụ: Sử dụng quy tắc bàn tay phải để đưa ra thứ tự đỉnh xác định mặt ngoài của các fragments trong hình hộp sau

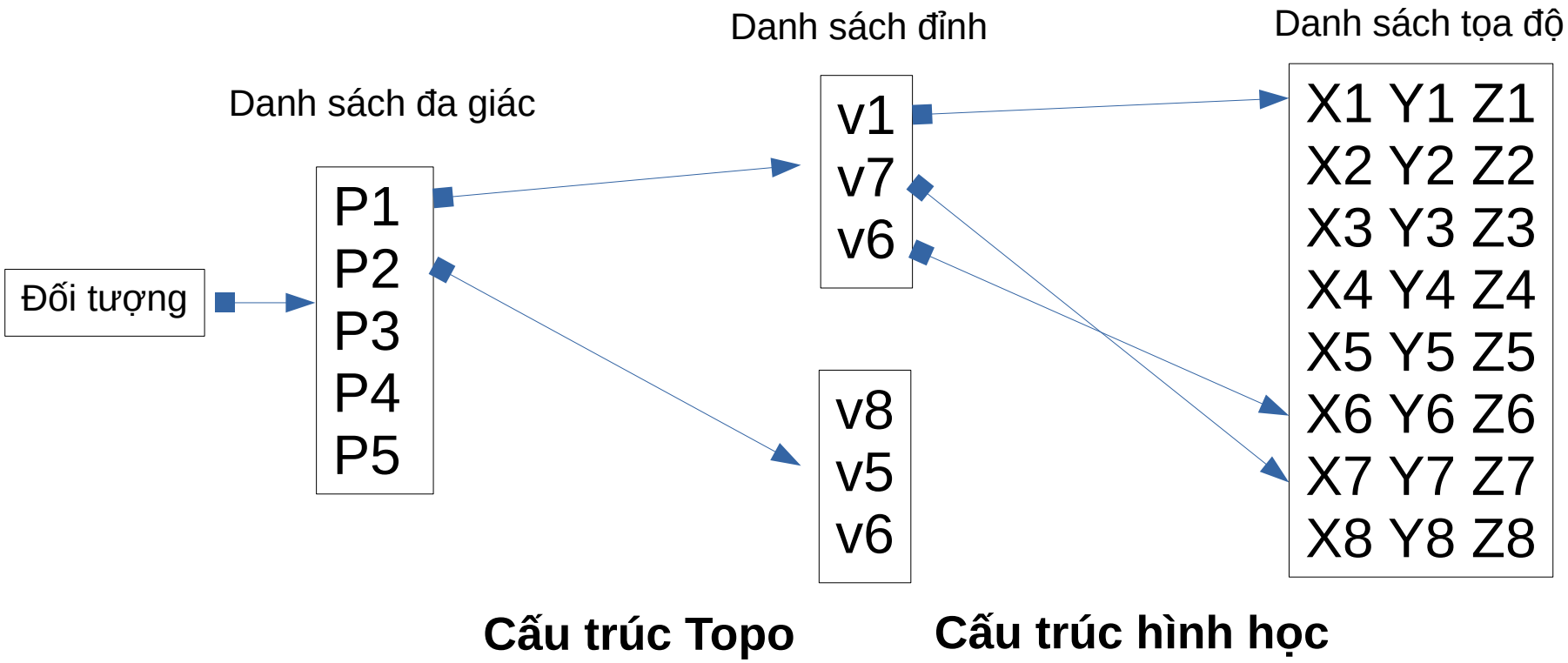


Cấu trúc dữ liệu xây dựng mô hình 3D

- Khi xây dựng cấu trúc dữ liệu cho đối tượng, ta cần cố gắng chia tách riêng rẽ **cấu trúc hình học** và **cấu trúc tôpô** của đối tượng.
  - Cấu trúc hình học là vị trí của các đỉnh.
  - Cấu trúc tôpô thể hiện mối liên hệ của các đỉnh và các cạnh.
    - Ví dụ: Đa giác là một danh sách các đỉnh có thứ tự với cạnh nối các cặp đỉnh liên tiếp và nối đỉnh cuối cùng với đỉnh đầu tiên.
    - Cấu trúc tôpô không đổi ngay cả khi cấu trúc hình học thay đổi.
- Có 2 dạng cấu trúc dữ liệu cơ bản để dựng một mô hình 3D
  - **Dựa trên danh sách đỉnh**
  - **Dựa trên danh sách cạnh**

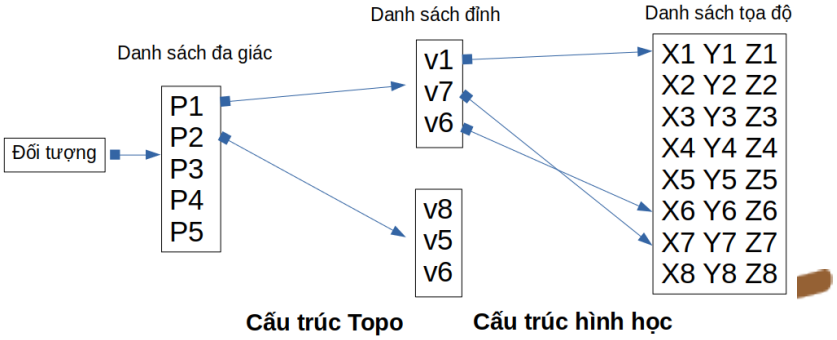
Cấu trúc dữ liệu xây dựng mô hình 3D

- Xây dựng cấu trúc dữ liệu dựa trên danh sách đỉnh



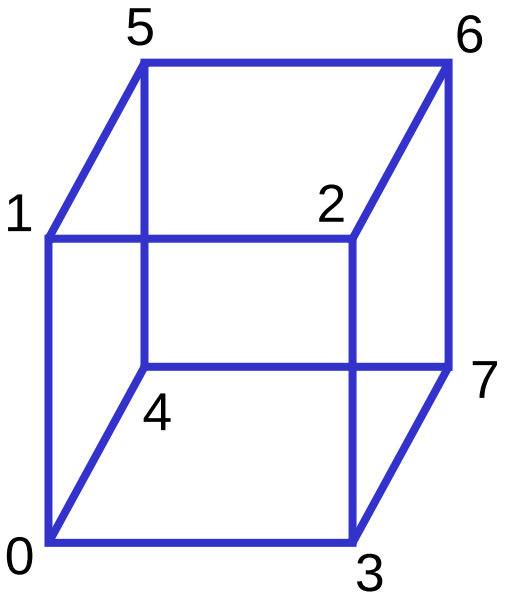
Cấu trúc dữ liệu xây dựng mô hình 3D

- Xây dựng cấu trúc dữ liệu dựa trên danh sách đỉnh
  - Giả sử ta cần xây dựng một **đối tượng** được tạo bởi 5 đa giác P1, P2, P3, P4, P5
  - Mỗi **đa giác** được tạo bởi một tập đỉnh nào đó
    - Ví dụ:
      - Đa giác P1 được tạo bởi các đỉnh v1, v7, v6
      - Đa giác P2 được tạo bởi các đỉnh v8, v5, v6
  - Mỗi **đỉnh** có một **tọa độ 3D** cụ thể
    - Ví dụ:
      - Đỉnh v1 có tọa độ (X1, Y1, Z1)
      - Đỉnh v6 có tọa độ (X6, Y6, Z6)



Cấu trúc dữ liệu xây dựng mô hình 3D

- Cấu trúc dữ liệu dựa trên danh sách đỉnh
  - Ví dụ: Vẽ hình hộp như trong hình bên
    - Hình hộp được tạo bởi 6 đa giác
    - Mỗi đa giác có chứa 4 đỉnh
    - Mỗi đỉnh sẽ có một tọa độ cụ thể nào đó
  - Thực hiện
    - Bước 1: Xác định tọa độ cho các đỉnh
    - Bước 2: Xây dựng hàm để vẽ hình dựa trên các đỉnh
      - Chú ý: thứ tự các đỉnh khi vẽ đa giác để xác định mặt ngoài cho chính xác.
      - Ví dụ: Xây dựng hàm quad(...) để vẽ một đa giác gồm 4 đỉnh.
        - quad(0, 3, 2, 1);
        - quad(2, 3, 7, 6);
        - quad(0, 4, 7, 3);
        - quad(1, 2, 6, 5);
        - quad(4, 5, 6, 7);
        - quad(0, 1, 5, 4);

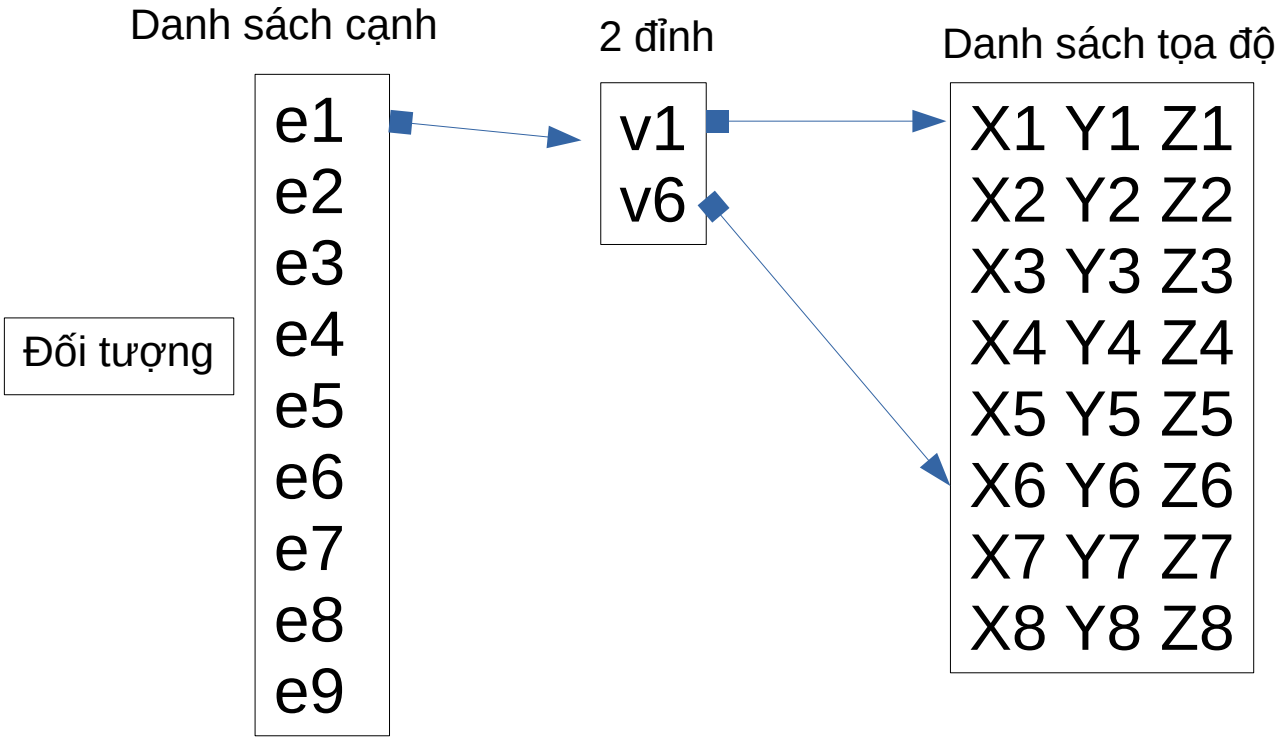


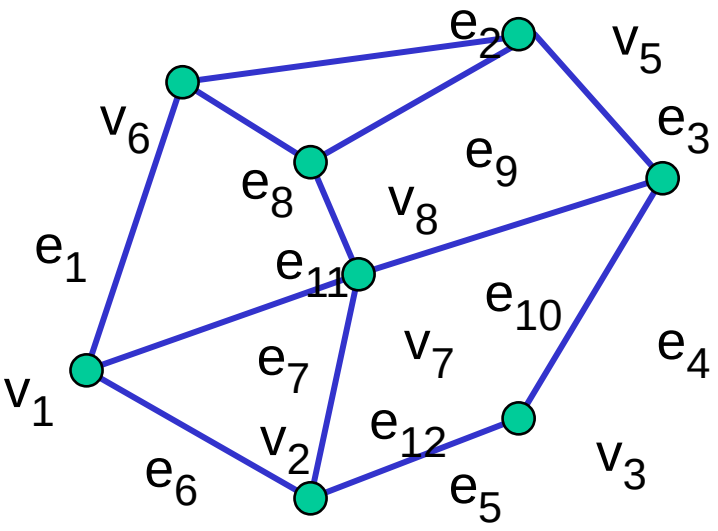
Ví dụ 2

- Vẽ một khối lập phương lên màn hình với tọa độ của mỗi đỉnh dựa trên tọa độ người sử dụng.
  - Mỗi mặt của hình lập phương là một màu khác nhau.
  - Sử dụng cấu trúc dữ liệu dựa trên danh sách đỉnh.

Cấu trúc dữ liệu xây dựng mô hình 3D

- Cấu trúc dữ liệu dựa trên danh sách cạnh

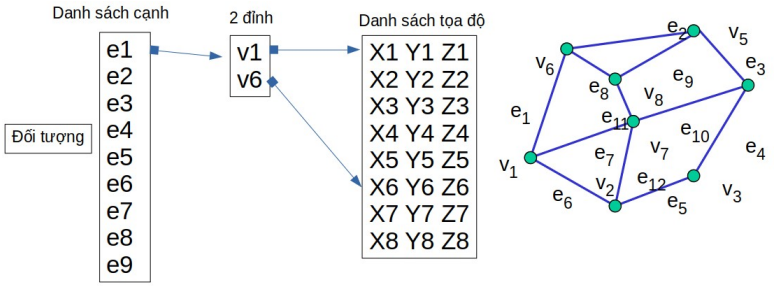




Trong cấu trúc này ta không cần mô tả các đa giác

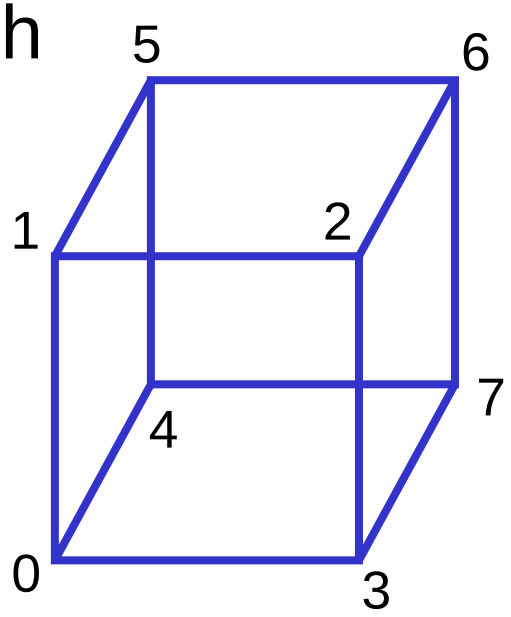
Cấu trúc dữ liệu xây dựng mô hình 3D

- Xây dựng cấu trúc dữ liệu dựa trên danh sách cạnh
  - Giả sử ta cần xây dựng một **đối tượng** được tạo bởi 9 cạnh e1, e2, e3, e4, e5, e6, e7, e8, e9
  - Mỗi **cạnh** được tạo bởi 2 đỉnh bất kỳ
    - Ví dụ:
      - Cạnh e1 được tạo bởi 2 đỉnh v1, v6
  - Mỗi **đỉnh** có một **tọa độ 3D** cụ thể
    - Ví dụ:
      - Đỉnh v1 có tọa độ (X1, Y1, Z1)
      - Đỉnh v6 có tọa độ (X6, Y6, Z6)



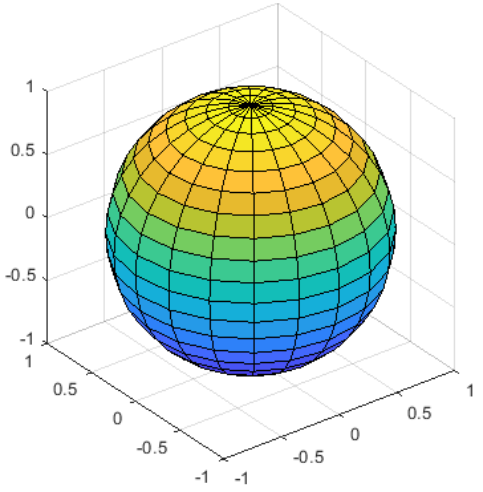
Cấu trúc dữ liệu xây dựng mô hình 3D

- Cấu trúc dữ liệu dựa trên danh sách cạnh
  - Ví dụ: Vẽ hình hộp như trong hình bên
    - Hình hộp được tạo bởi 12 cạnh
    - Mỗi cạnh được tạo bởi 2 đỉnh
    - Mỗi đỉnh sẽ có một tọa độ cụ thể nào đó
  - Thực hiện
    - Bước 1: Xác định tọa độ cho các đỉnh
    - Bước 2: Xây dựng hàm để vẽ cạnh dựa trên các đỉnh
      - Ví dụ: Sử dụng tham số gl.LINES để vẽ các cạnh dựa trên các đỉnh nằm trong danh sách
  - Ta nên sử dụng cấu trúc dữ liệu dạng này cho các mô hình mà khi vẽ hình không cần tô màu hoặc sử dụng texture.

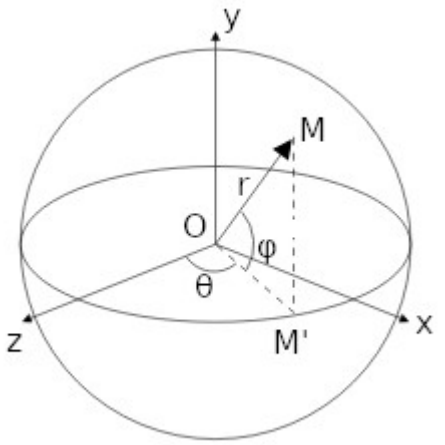


Ví dụ về dựng hình cầu

- Sử dụng các đường **kinh tuyến** và **vĩ tuyến**.
  - Tính toán để vẽ các ô hình chữ nhật **dựa trên các đỉnh giao cắt giữa kinh tuyến và vĩ tuyến**.
  - Khi khoảng cách giữa các đường kinh tuyến và vĩ tuyến càng nhỏ thì hình cầu sẽ càng mịn và càng chính xác.



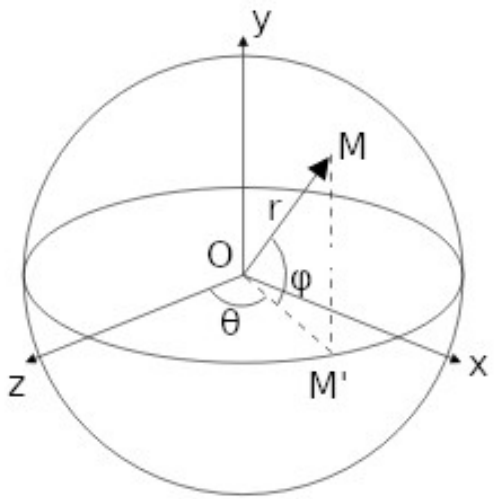
Vẽ hình cầu dựa trên kinh tuyến và vĩ tuyến



- Giả sử
  - **O** là gốc tọa độ.
  - **M** là một điểm nằm trên mặt cầu.
  - **M’** là hình chiếu của điểm M lên mặt phẳng được tạo bởi trục X và trục Z (MM’ vuông góc với mặt phẳng này)
  - **r** là bán kính của hình cầu, cũng có độ dài bằng với đoạn thẳng OM.
  - **φ** là góc được tạo bởi góc MOM’.
  - **θ** là góc được tạo bởi góc giữa tia OM’ và trục tọa độ Z.

Vẽ hình cầu dựa trên kinh tuyến và vĩ tuyến

- Nhận xét
  - Tất cả các điểm trên mặt cầu đều có thể được vẽ bằng cách **thay đổi giá trị của các góc θ và φ**.
    - **Góc φ** sẽ thay đổi từ -90 độ đến 90 độ.
    - **Góc θ** sẽ thay đổi từ 0 độ đến 360 độ.



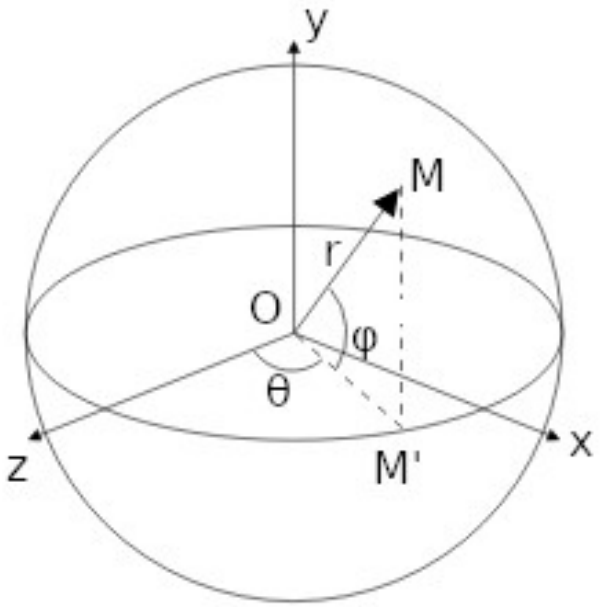
Vẽ hình cầu dựa trên kinh tuyến và vĩ tuyến

- Ta có công thức để tính **tọa độ (x, y, z) của điểm M theo bán kính r và 2 góc θ và φ** như sau:

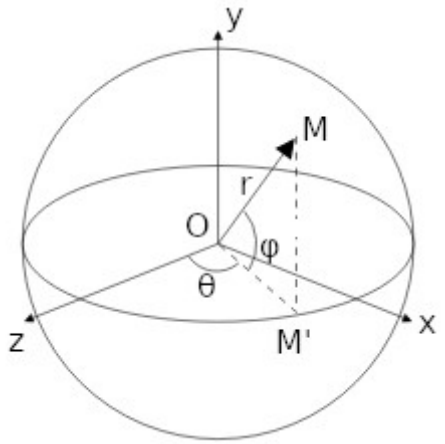
$$M_x = r * \sin(\theta) * \cos(\varphi)$$

$$M_y = r * \sin(\varphi)$$

$$M_z = r * \cos(\theta) * \cos(\varphi)$$



Vẽ hình cầu dựa trên kinh tuyến và vĩ tuyến



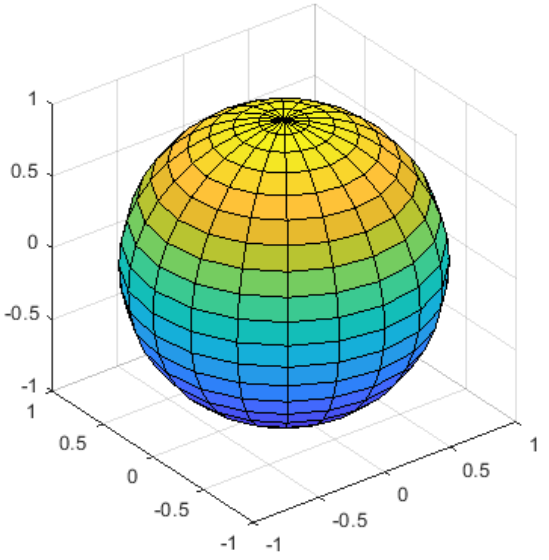
- Thuật toán

```
for ( -90 <= φ <= (90 - φjump); φ += φjump)
    if (φ == -90)
        for(0 <= θ <= (360 - θjump); θ += θjump)
            veTamGiac({r, θ, φ}, {r, θ, φ+φjump}, {r, θ+θjump, φ+φjump});
    else if (φ == (90 – φjump))
        for(0 <= θ <= (360 - θjump); θ += θjump)
            veTamGiac({r, θ, φ}, {r, θ+θjump, φ}, {r, θ, φ+φjump});
    else
        for(0 <= θ <= (360 - θjump); θ += θjump)
            veTuGiac( {r, θ, φ}, {r, θ+θjump, φ}, {r, θ+θjump, φ+φjump}, {r, θ, φ+φjump} );
```

- Chú ý:
  - Thủ tục **VeTuGiac** có 4 đỉnh đưa vào và phải tuân theo quy tắc bàn tay phải.

Ví dụ 3

- Vẽ một hình cầu theo thuật toán đã nêu ở trên, với các đỉnh tại kinh tuyến và vĩ tuyến cách đều nhau một khoảng 10 độ và mỗi mảnh tứ giác tạo nên hình cầu sẽ có một màu ngẫu nhiên.



Phép biến đổi Affine

- Giả sử trên không gian 3 chiều ta có một điểm P có tọa độ $\mathbf{(P_x, P_y, P_z)}$.
- Thông qua một phép biến đổi **T** nào đó, ta cần biến đổi điểm P thành một điểm Q có tọa độ là $\mathbf{(Q_x, Q_y, Q_z)}$
- Trong không gian Affine, tọa độ của Q là **tổ hợp tuyến tính** của tọa độ P.

$$Q_x = m_{11}P_x + m_{12}P_y + m_{13}P_z + m_{14}$$

$$Q_y = m_{21}P_x + m_{22}P_y + m_{23}P_z + m_{24}$$

$$Q_z = m_{31}P_x + m_{32}P_y + m_{33}P_z + m_{34}$$

Phép biến đổi Affine

- Có 2 cách biểu diễn dựa trên ma trận để thể hiện mối quan hệ giữa P và Q
  - Cách 1:

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} + \begin{pmatrix} m_{14} \\ m_{24} \\ m_{34} \end{pmatrix}$$

  - Cách 2 (Ma trận đồng nhất):

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

  - Trong khóa học này ta chọn cách sử dụng **ma trận đồng nhất.**

Phép biến đổi hình 3D

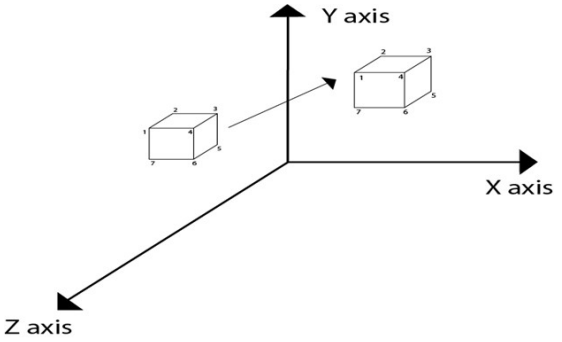
- Phép tịnh tiến

$$Q_x = P_x + m_{14}$$

$$Q_y = P_y + m_{24}$$

$$Q_z = P_z + m_{34}$$

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & m_{14} \\ 0 & 1 & 0 & m_{24} \\ 0 & 0 & 1 & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

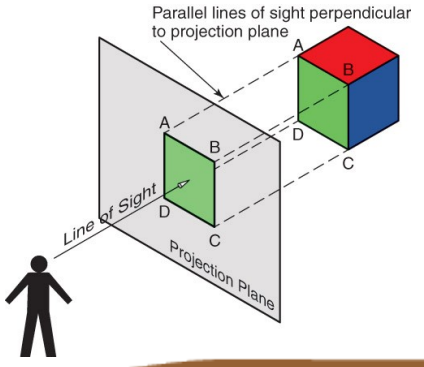


  - Ví dụ: Dịch chuyển một điểm theo trục X 5 đơn vị, trục Y 2 đơn vị và trục Z 6 đơn vị ta sử dụng ma trận biến đổi sau

$$\begin{pmatrix} 1 & 0 & 0 & \mathbf{5} \\ 0 & 1 & 0 & \mathbf{2} \\ 0 & 0 & 1 & \mathbf{6} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

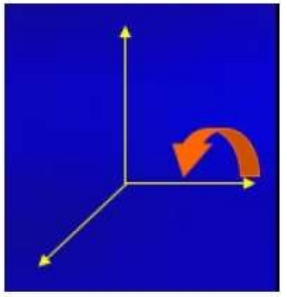
Ví dụ 4

- Vẽ một hình hộp chữ nhật lên màn hình, sau đó **tịnh tiến hình hộp chữ nhật** bằng việc thiết lập các giá trị dịch chuyển cho trục X, trục Y và trục Z.
  - Chú ý: Do trục Z mặc định nằm trên đường thẳng nối mắt người sử dụng với gốc tọa độ, và phép chiếu mặc định là phép chiếu song song nên ta không cảm nhận được phép tịnh tiến khi thay đổi giá trị dịch chuyển cho trục Z.

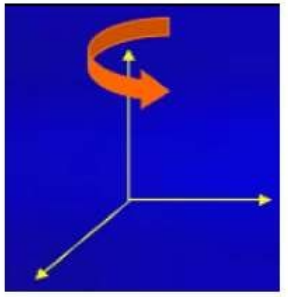


Phép biến đổi hình 3D

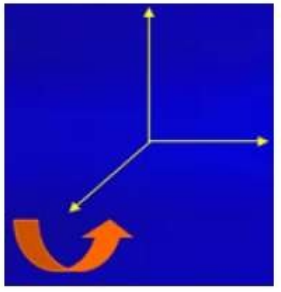
- Phép quay



Rotation about x-axis



Rotation about y-axis



Rotation about z-axis

- Phép quay quanh trục X

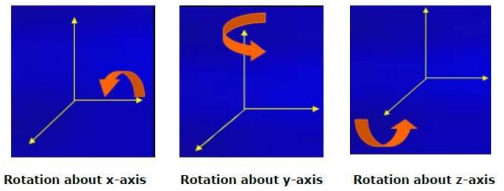
$Q_x = P_x$

$Q_y = \cos(\theta)P_y - \sin(\theta)P_z$

$Q_z = \sin(\theta)P_y + \cos(\theta)P_z$

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

Phép biến đổi hình 3D



- Phép quay quanh trục Y

$$Q_x = \cos(\theta)P_x + \sin(\theta)P_z$$

$$Q_y = P_y$$

$$Q_z = -\sin(\theta)P_x + \cos(\theta)P_z$$

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

- Phép quay quanh trục Z

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

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

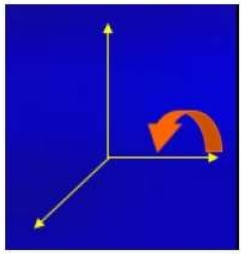
$$Q_z = P_z$$

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

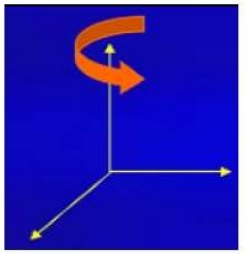
Phép biến đổi hình 3D

- Ví dụ: Quay một điểm quanh trục X một góc 45 độ, ta sử dụng ma trận sau

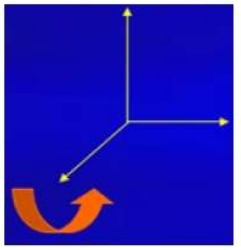
$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$



Rotation about x-axis



Rotation about y-axis



Rotation about z-axis

Ví dụ 5

- Vẽ một hình hộp chữ nhật lên màn hình, sau đó **quay hình hộp chữ nhật** quanh trục X, trục Y hoặc trục Z bằng việc thiết lập các góc quay.
  - Chú ý:
    - Thiết lập góc quay từ 0 độ cho đến 360 độ.
    - Sử dụng các hàm **cos**, **sin** có sẵn trong thư viện của GLSL. Đầu vào của các hàm này là các tham số góc theo Radian.
    - Sử dụng các hàm **radians** có sẵn trong thư viện của GLSL để chuyển đổi một từ độ sang radian.

Ví dụ 6

- Vẽ một hình hộp chữ nhật lên màn hình, sau đó cho **hình hộp chữ nhật** quay liên tục. Quay quanh trục nào sẽ do người sử dụng điều khiển thông qua các Button.

Quay quanh trục X | Quay quanh trục Y | Quay quanh trục Z

Ví dụ 7

- Dựa trên ví dụ 6, nhưng cho **hình cầu** quay liên tục. Quay quanh trục nào sẽ do người sử dụng điều khiển thông qua các Button.
  - Thay đổi các giá trị φ_{jump} và θ_{jump} trong mã nguồn để thấy sự thay đổi của hình cầu.

Quay quanh trục X | Quay quanh trục Y | Quay quanh trục Z

Phép biến đổi hình 3D

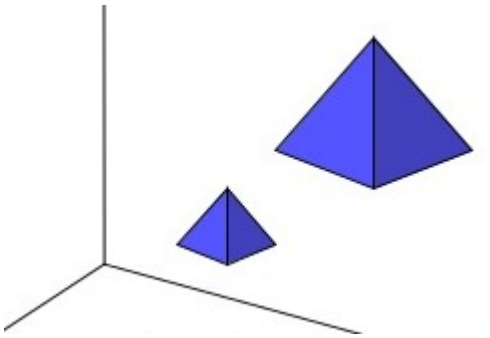
- Phép biến đổi tỉ lệ

$$Q_x = S_x P_x$$

$$Q_y = S_y P_y$$

$$Q_z = S_z P_z$$

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



  - Ví dụ: Biến đổi tỷ lệ một hình theo tỉ lệ đối với trục X là 1.5, với trục Y là 0.6 và với trục Z là 2.1 ta sử dụng ma trận sau

$$\begin{pmatrix} \mathbf{1.5} & 0 & 0 & 0 \\ 0 & \mathbf{0.6} & 0 & 0 \\ 0 & 0 & \mathbf{2.1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ví dụ 8

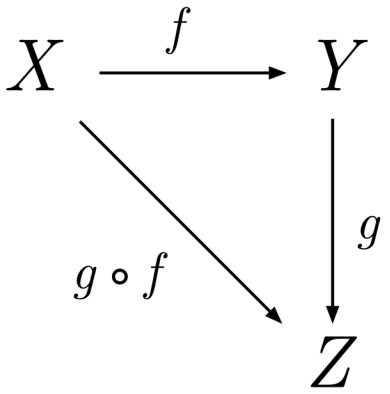
- Vẽ một hình hộp chữ nhật lên màn hình, sau đó **biến đổi tỷ lệ hình hộp** bằng việc thiết lập các giá trị tỷ lệ cho trục X, trục Y và trục Z.
  - Chú ý: Với giá trị nằm trong khoảng (0, 1), hình sẽ bị thu nhỏ.

Ví dụ 9

- Dựa trên Ví dụ 6, thêm chức năng phóng to và thu nhỏ hình hộp chữ nhật trong khi đang quay.

Phép biến đổi hình 3D

- Phép biến đổi tổ hợp
  - Cũng giống như phép biến đổi hình 2D, trong thực tế, ta thường phải thực hiện **các phép biến đổi phức tạp** bằng các **phép biến đổi tổ hợp**.
    - Ví dụ: Quay hình quanh một trục bất kỳ.
  - Công thức để xây dựng các phép biến đổi tổ hợp trong mô hình 3D cũng giống như trong mô hình 2D.



Hết Tuần 5

Cảm ơn các bạn đã chú ý lắng nghe !!!