TỔNG QUAN CHUNG – Phần 2

Tổng quan chung

- Bits và Bytes
- Màu sắc
- Đồ họa 2D & 3D
- Hình học cơ bản
- Toán học trong đồ họa máy tính

Bits và Bytes

Bit (Binary Digit)

Đơn vị thông tin nhỏ nhất, chỉ nhận giá trị 0 hoặc 1. Nó tương ứng với các trạng thái logic như True/False hay On/Off.

Byte

Tập hợp 8 bits, là đơn vị cơ sở để biểu diễn các kiểu dữ liệu như ký tự, màu sắc, hay giá trị số.

VD: Một kênh màu (Red, Green hoặc Blue) thường được lưu bằng **1 byte**
Giá trị 0 biểu thị không có cường độ
Giá trị 255 biểu thị cường độ tối đa

Name	Size
Byte (char)	8 bits (1 byte)
Integers	32 bits (4 bytes)
Short	16 bits (2 bytes)
Floating-point values	32 bits (4 bytes)

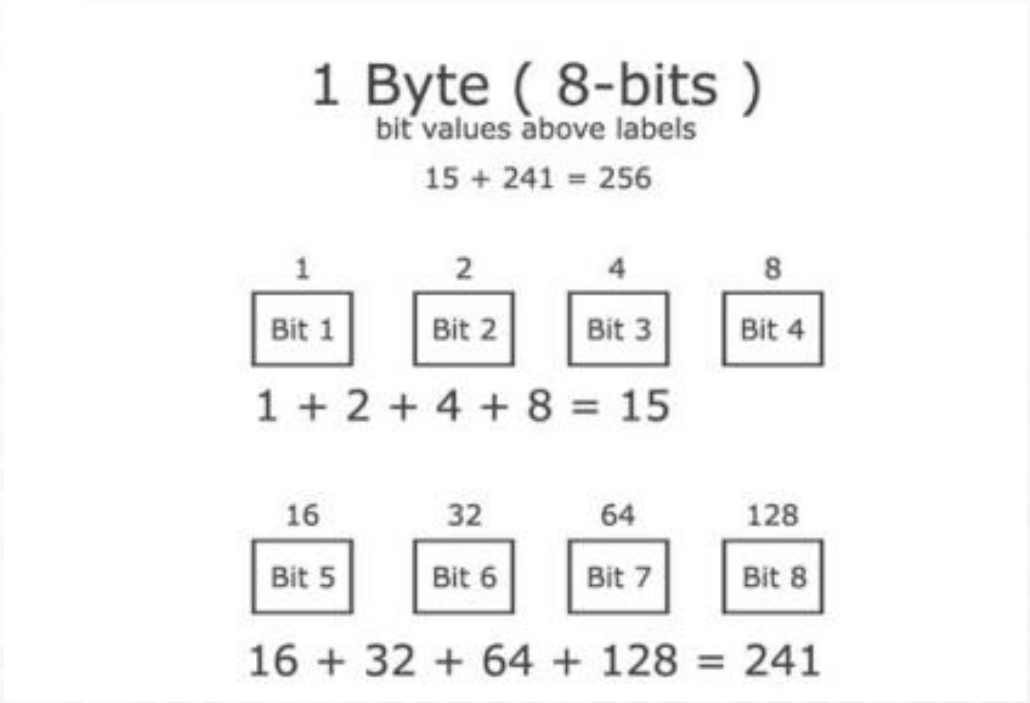
Mối quan hệ giữa số Bits và độ chính xác

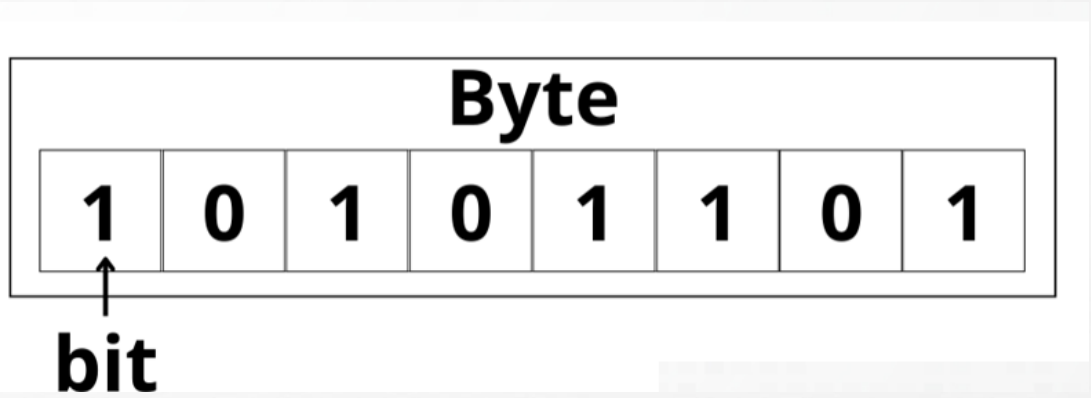
Càng nhiều bits thì độ chính xác càng cao

- Tốn nhiều bộ nhớ hơn
- Tốn nhiều băng thông truyền dữ liệu hơn
- Gây áp lực lên hiệu năng, đặc biệt trong - đồ họa thời gian thực

*Đồ họa luôn là sự đánh đổi giữa **độ chính xác** và **hiệu năng***

Bits và Bytes



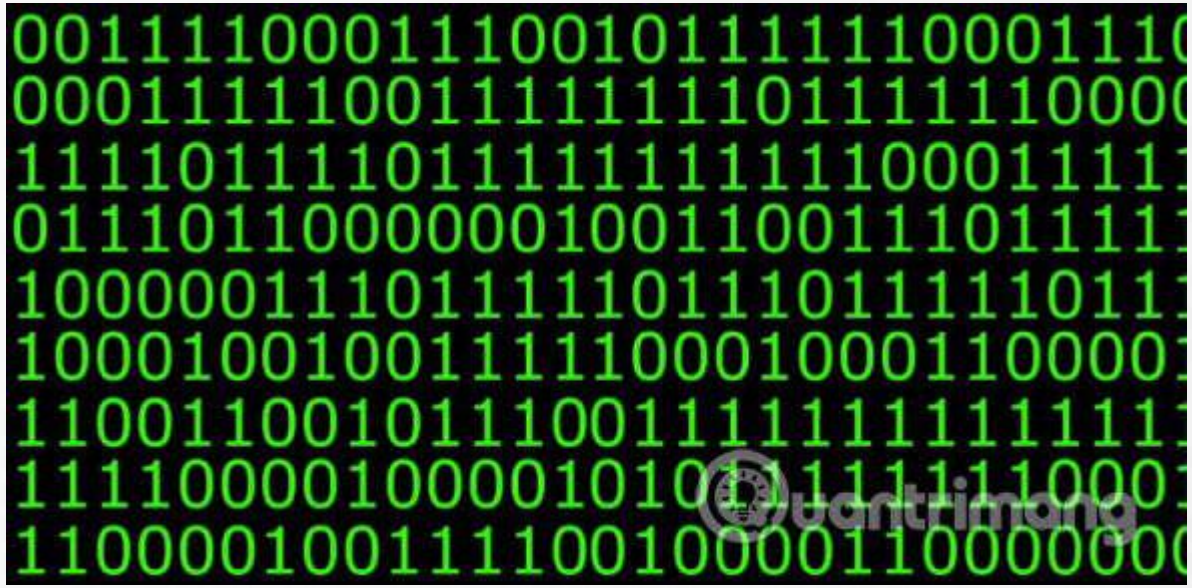


$$Value = \sum_{i=0}^{7} 2^i * b_i = 2^0 * 1 + 2^1 * 0 + 2^2 * 1 + 2^3 * 1 + 2^4 * 0 + 2^5 * 1 + 2^6 * 0 + 2^7 * 1 = 173$$

Biểu diễn số và Hệ cơ số

Hệ Nhị phân (Base-2)

- ✓ Hệ nhị phân là hệ đếm **cơ số 2**, chỉ sử dụng hai chữ số 0 và 1
- ✓ Máy tính sử dụng **hệ nhị phân** làm ngôn ngữ cơ bản.
- ✓ Mỗi bit đại diện cho một lũy thừa của 2, tạo nên các giá trị số.



Biểu diễn số và Hệ cơ số

Hệ Nhị phân (Base-2)

Máy tính dung hệ nhị phân vì:

Phần cứng điện tử chỉ dễ dàng phân biệt **hai trạng thái:**

- Có điện/ Không có điện
- Bật/ Tắt

Việc dung nhiều mức điện áp (như hệ thập phân) sẽ:

- Phức tạp
- Kém ổn định
- Dễ sinh lỗi

Hệ nhị phân là lựa chọn **đơn giản, bền vững và đáng tin cậy** cho máy tính

Biểu diễn số và Hệ cơ số

Hệ Thập lục phân (Hexadecimal)

- Hệ thập lục phân là hệ đếm **cơ số 16**, sử dụng Các chữ số từ **0 đến 9 và Các chữ cái từ A đến F (đại diện cho 10–15)**
- Hệ thập lục phân được dùng để biểu diễn các giá trị phức tạp một cách gọn gàng hơn

VD: Dùng để **Biểu diễn màu sắc như** màu trắng là 0xFFFFFF, đen là 0x000000.

→ Hexadecimal dễ đọc hơn nhị phân, đặc biệt khi làm việc với các giá trị lớn.

Mối quan hệ giữa Binary và Hexadicimal

1 chữ số hex = 4 bit nhị phân

Hexadecimal chỉ là cách viết gọn và dễ đọc của Binary

VD: $1111_2 = F_{16}$
$0000_2 = 0_{16}$
$10101100_2 = AC_{16}$

Lập trình viên không phải đọc chuỗi bit dài và việc debug/đọc dữ liệu đồ họa trở nên dễ dàng hơn

Binary là ngôn ngữ của máy

Hexadecimal là ngôn ngữ trung gian cho con người

Đồ họa máy tính không chỉ là vẽ hình, mà là **quản lý và biến đổi dữ liệu số**

Màu sắc trong đồ họa máy tính

Pixel và màn hình raster

Pixel (picture element) là đơn vị nhỏ nhất của hình ảnh raster. Mỗi pixel chứa:

- Màu sắc (RGB, RGBA)
- Có thể có thông tin độ sâu hoặc alpha

Màn hình raster

Hầu hết màn hình hiện đại là raster display:

Hình ảnh được tạo từ lưới pixel

Ví dụ: 1920 × 1080 = hơn 2 triệu pixel

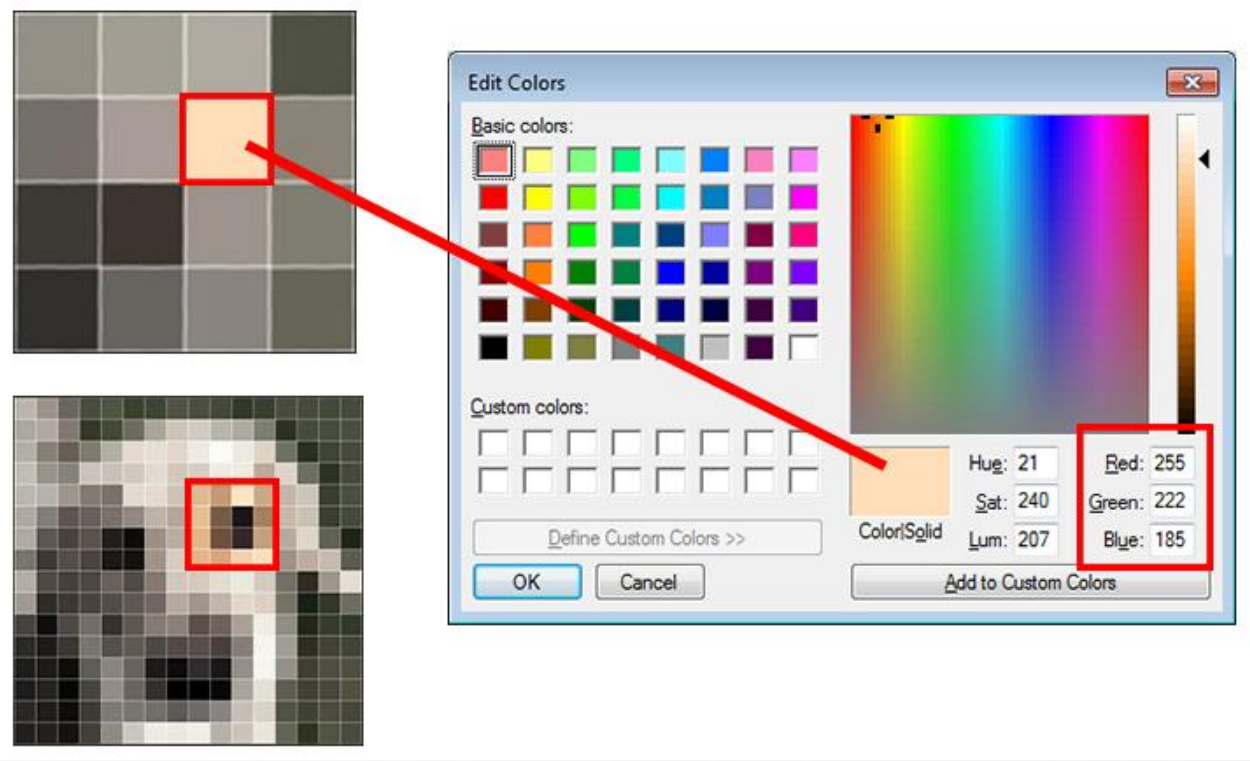
Trong đồ họa 2D:

- Mỗi pixel được vẽ trực tiếp lên framebuffer

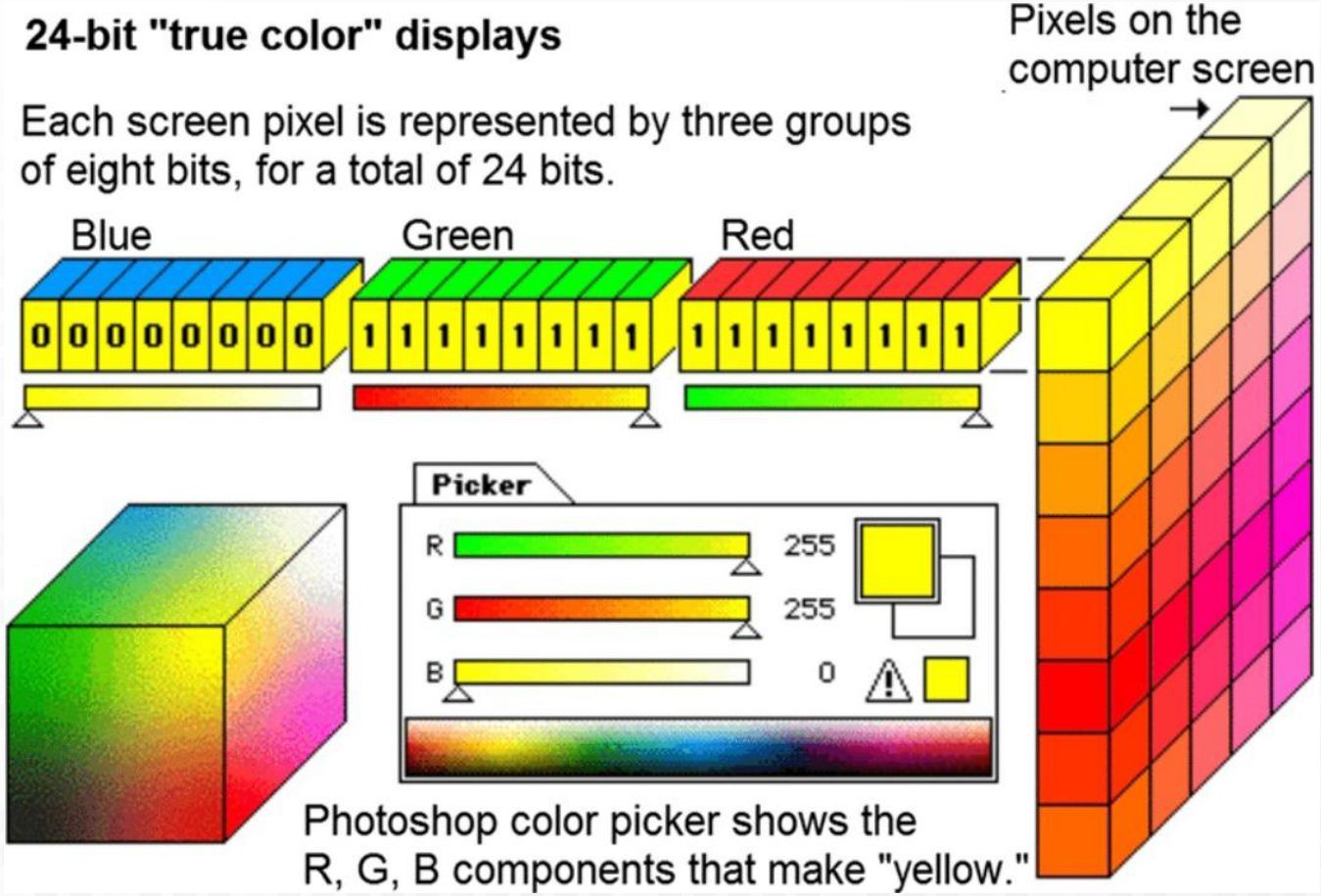
Biểu diễn về màu sắc trong đồ họa máy tính

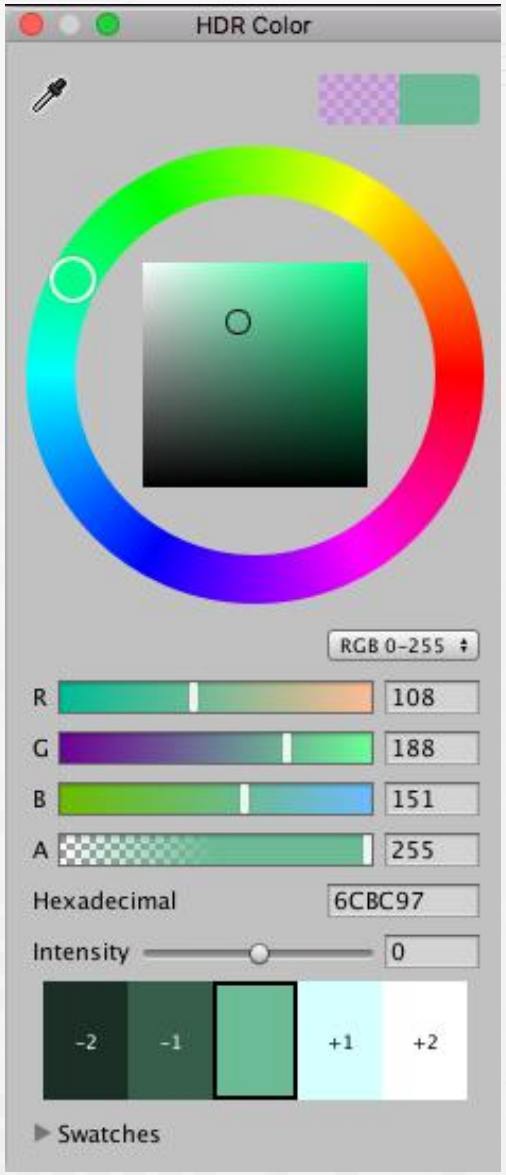
Trong đồ hoạ máy tính, màu sắc là tập hợp các giá trị số được lưu trữ và xử lý bởi phần cứng

Mỗi pixel trên màn hình được gán 1 giá trị màu, và giá trị này được xác định bởi mô hình màu



Cách biểu diễn dữ liệu màu sắc



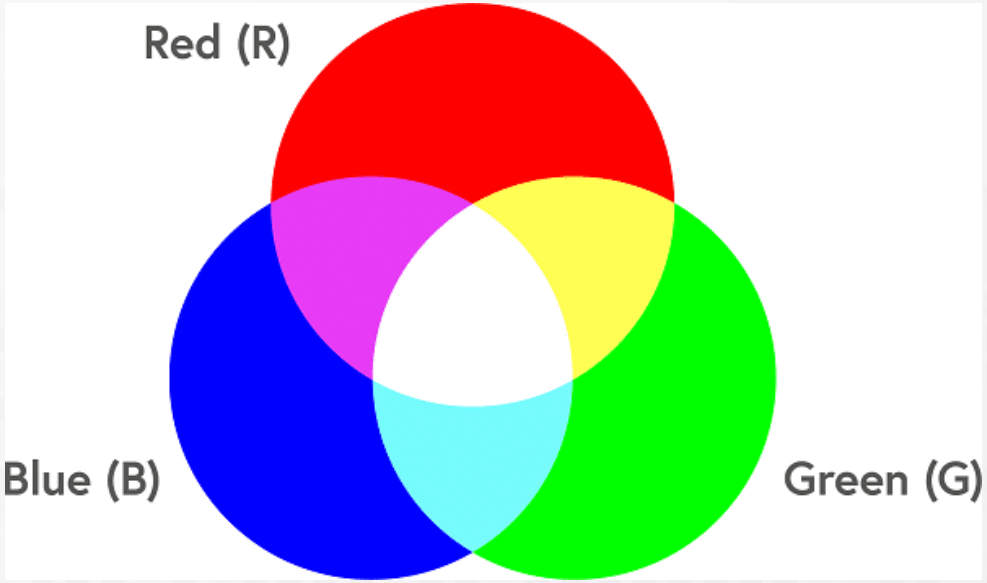


https://www.researchgate.net/figure/In-the-RGB-color-space-each-pixel-24-bits-is-represented-by-Red-Green-and-Blue-of_fig3_356729886

Mô hình màu RGB

Mô hình RGB dựa trên **trộn màu ánh sáng** (additive color model), gồm ba thành phần:

- ✓ R (Red – đỏ)
- ✓ G (Green – xanh lá)
- ✓ B (Blue – xanh dương)



Mỗi màu được tạo ra bằng cách:

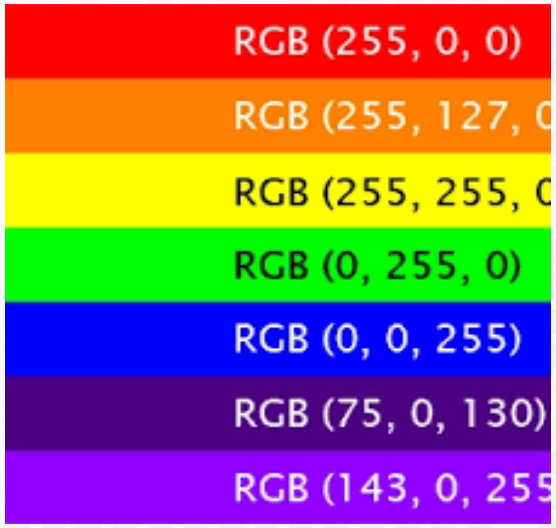
- ✓ Trộn ba thành phần R, G, B với các cường độ khác nhau

Biểu diễn RGB bằng số

Thông thường:

- Mỗi kênh màu dùng 1 byte (8 bit)
- Giá trị từ 0 đến 255)

Tổng số màu có thể biểu diễn:
256^3 = 16.777.216 màu



Ví dụ:

(255, 0, 0) → đỏ thuần

(0, 255, 0) → xanh lá

(0, 0, 255) → xanh dương

(255, 255, 255) → trắng

(0, 0, 0) → đen

Alpha Channel – Kênh trong suốt

Khái niệm

Ngoài RGB, một pixel thường có thêm:

- **Alpha (A):** độ trong suốt

RGBA:

- R, G, B: màu
- A: mức độ hiển thị (0 = trong suốt, 1 hoặc 255 = đục)

Alpha	hsla(H,S,L,A)	Color
0.0	rgba(255, 0, 0, 0.0)	fully transparent
0.2	rgba(255, 0, 0, 0.2)	
0.4	rgba(255, 0, 0, 0.4)	
0.6	rgba(255, 0, 0, 0.6)	
0.8	rgba(255, 0, 0, 0.8)	
1.0	rgba(255, 0, 0, 1.0)	fully opaque

Ứng dụng Alpha

Sprite 2D

UI

Particle effect

Particle effect

Cách biểu diễn dữ liệu màu sắc

4 Kênh RGBA

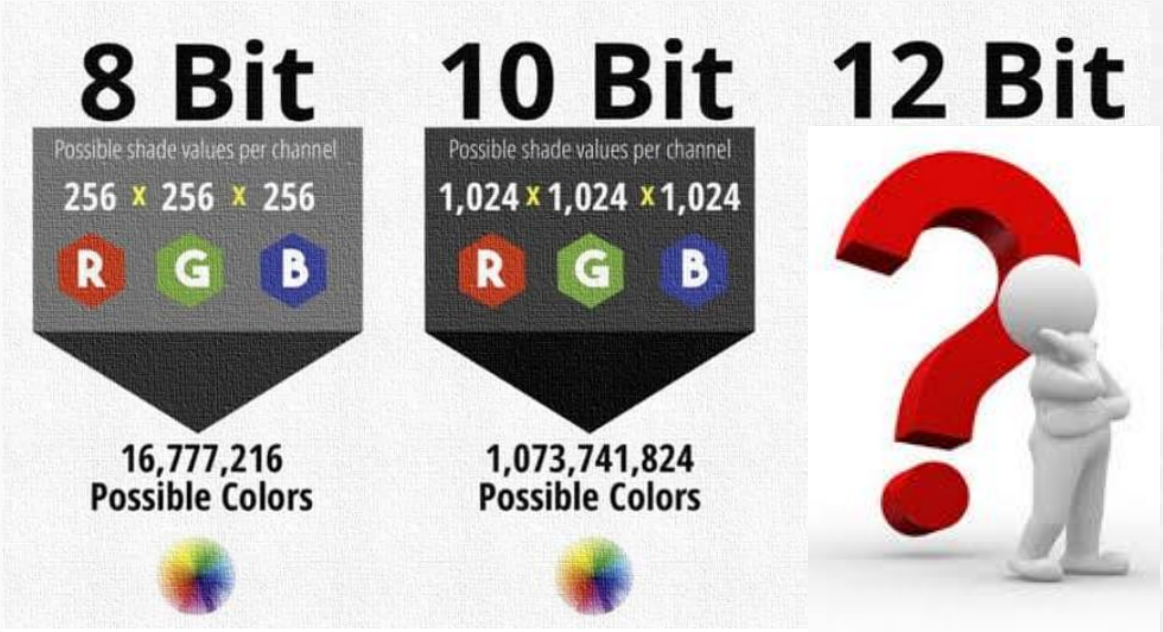
Chuẩn phổ biến trong game hiện nay, bao gồm R, G, B và kênh **Alpha (A)** để quy định độ trong suốt.

8-bit (LDR)

Giới hạn giá trị màu từ 0-255 cho mỗi kênh (tương ứng với một biến `unsigned char`), thường dùng trong đồ họa thông thường

10-bit trở lên (HDR)

Có thể thể hiện được rất nhiều sắc độ khác nhau, dùng trong đồ họa chất lượng rất cao

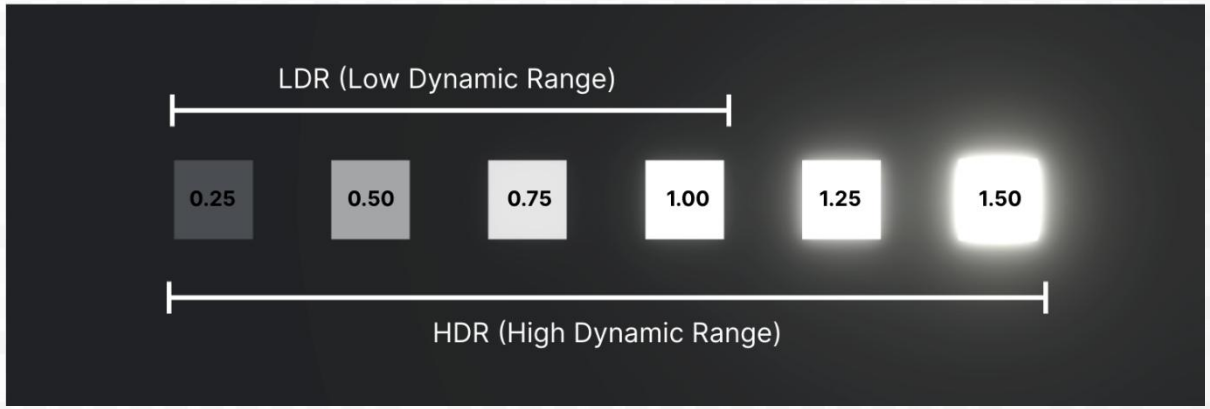


LDR – Low Dynamic Range

Khái niệm LDR

Là cách biểu diễn màu truyền thống:

- Giá trị màu bị giới hạn trong khoảng 0–255
- Không biểu diễn được cường độ ánh sáng vượt ngưỡng này



Hạn chế của LDR

- Ánh sáng thực tế có dải cường độ rất rộng
- LDR không thể mô phỏng:

✓ *Ánh sáng mặt trời chói*

✓ *Hiệu ứng bloom tự nhiên*

✓ *Sự thích nghi của mắt người*

➢ ***Hệ quả là hình ảnh thiếu chiều sâu ánh sáng***

HDR - High Dynamic Range

HDR ra đời để:

- Mô phỏng ánh sáng gần với thực tế hơn
- Lưu trữ giá trị màu lớn hơn 1.0

Trong HDR

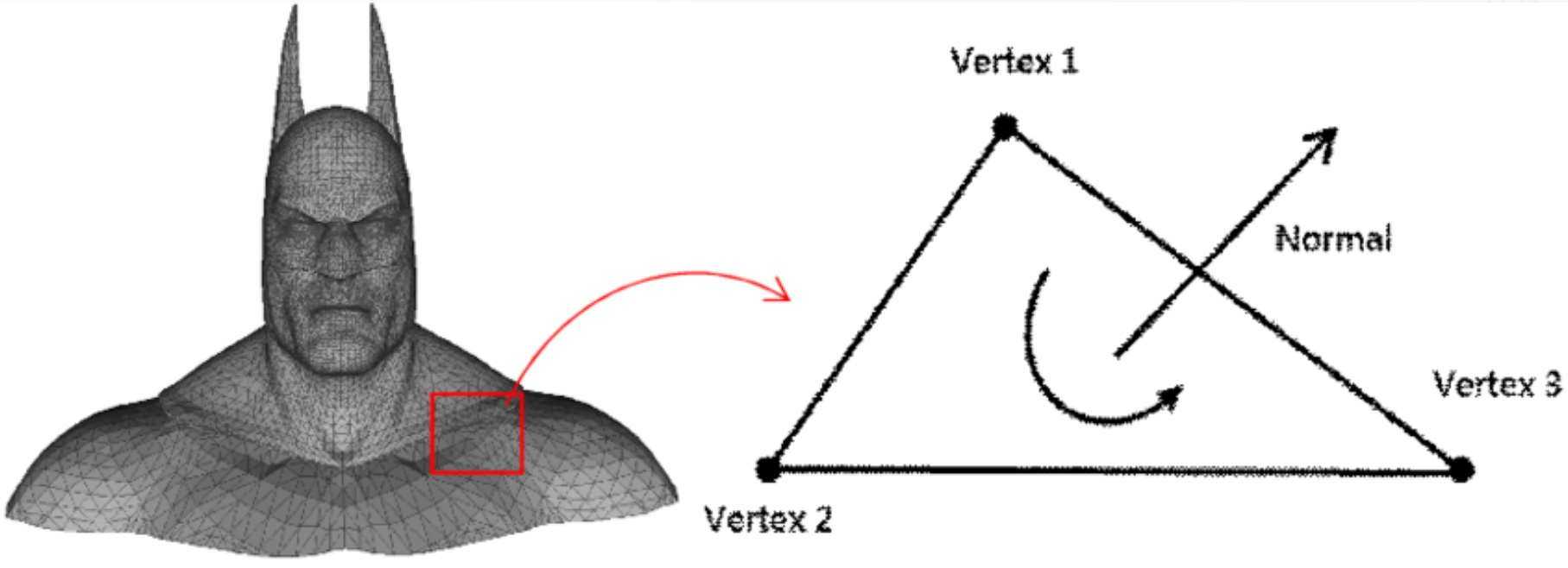
- Màu không còn bị giới hạn 0–255
- Thường dùng floating-point (số thực)

Ví dụ:

- Ánh sáng mặt trời có thể có giá trị: (5.0, 4.8, 4.6)
- Đèn yếu: (0.2, 0.2, 0.2)

→ Những giá trị này không thể lưu bằng LDR

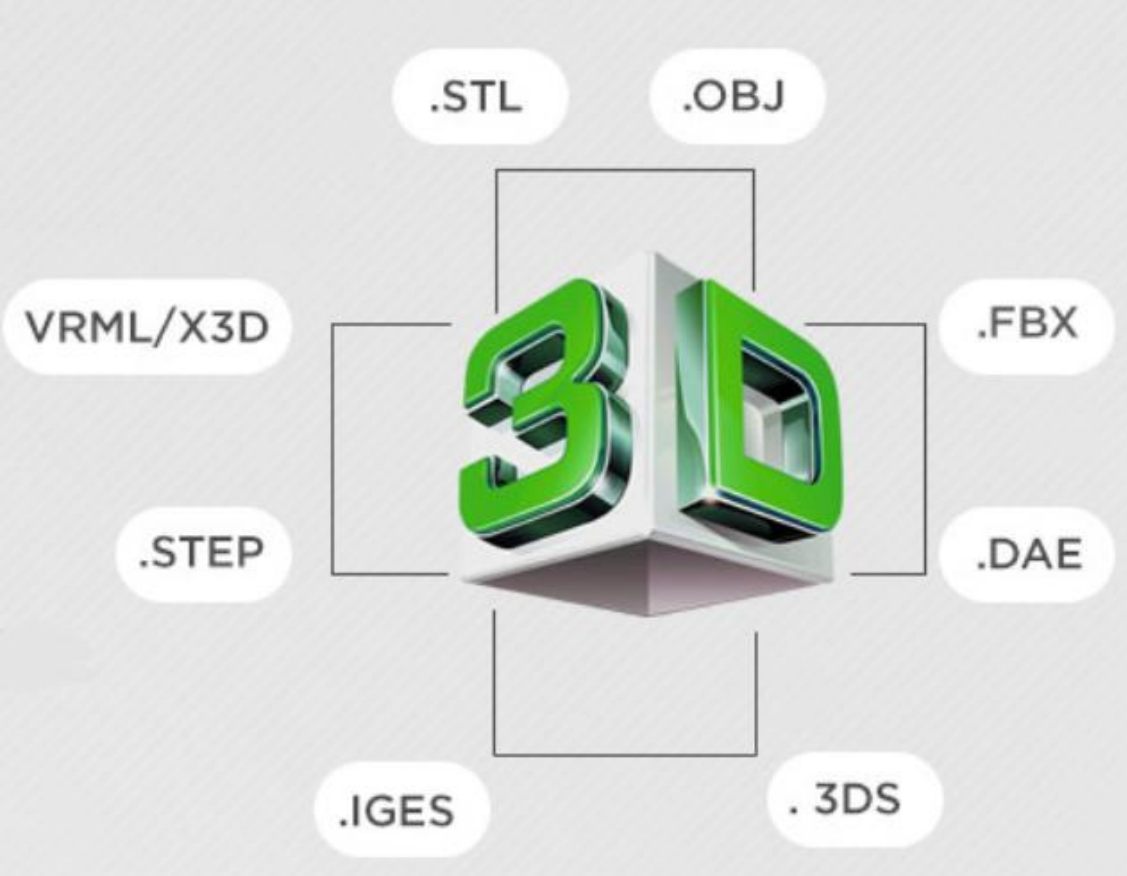
Model 3D



❓ Vì sao tam giác được dùng nhiều nhất trong đồ họa game?

- **Luôn đồng phẳng:** Ba điểm luôn tạo thành một mặt phẳng duy nhất, giúp tính toán và render dễ dàng.
- **Dễ nội suy:** Đơn giản để tính toán các giá trị màu, ánh sáng, và texture trên bề mặt.
- **GPU xử lý nhanh:** Các GPU hiện đại được tối ưu hóa để xử lý hàng tỷ tam giác mỗi giây, đảm bảo hiệu suất cao.
- **Tam giác hóa:** Mọi polygon (đa giác) đều có thể được chia nhỏ thành các tam giác (tam giác hóa).

Cấu trúc dữ liệu Model 3D



Một số định dạng file 3D

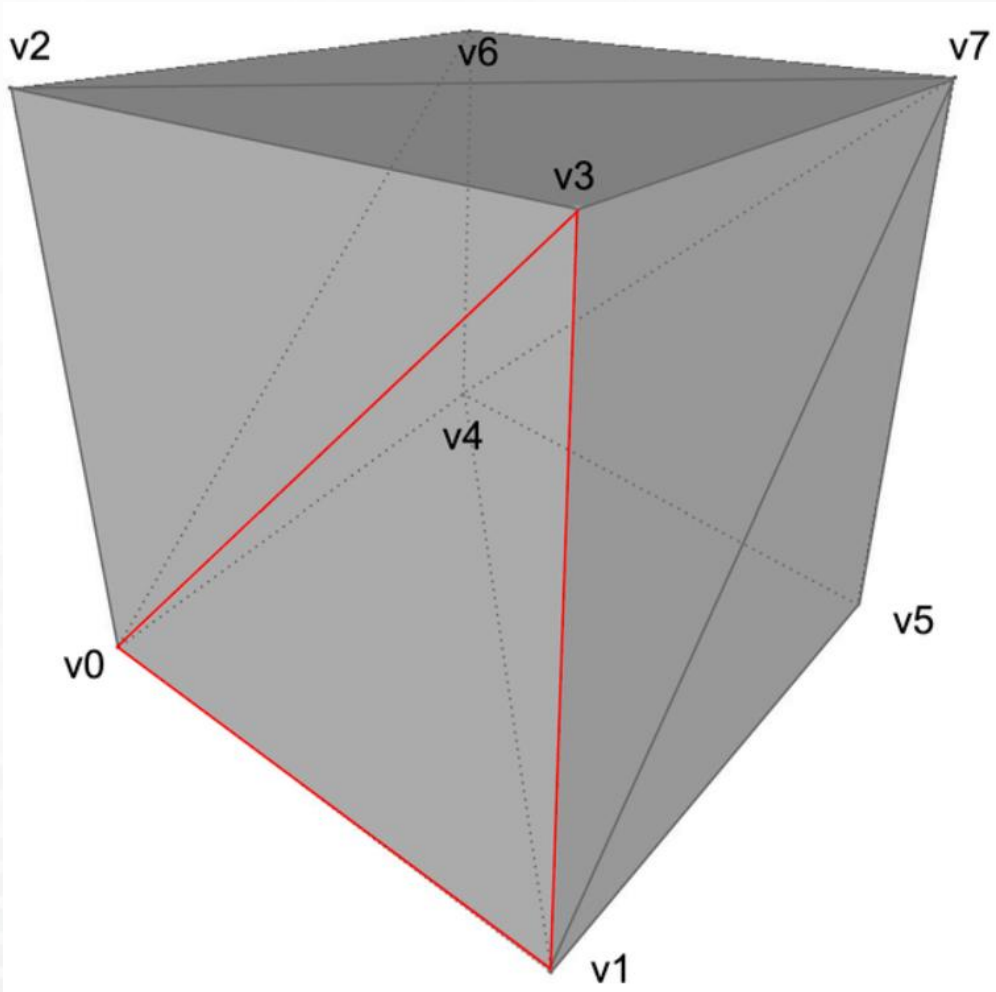
Vertices (Đỉnh)

Các điểm trong không gian 3D xác định vị trí của các góc của tam giác.

Tập hợp Tam giác

Một Mesh được định nghĩa bởi một tập hợp các tam giác, mỗi tam giác lại được tạo thành từ ba đỉnh (Vertices).
Được sắp xếp theo chiều ngược kim đồng hồ (Counter Clockwise)

Cấu trúc dữ liệu Model 3D




```
v 0.0 0.0 0.0
v 0.0 0.0 1.0
v 0.0 1.0 0.0
v 0.0 1.0 1.0
v 1.0 0.0 0.0
v 1.0 0.0 1.0
v 1.0 1.0 0.0
v 1.0 1.0 1.0
f 0 6 4
f 0 2 6
f 0 3 2
f 0 1 3
f 2 7 6
f 2 3 7
f 4 6 7
f 4 7 5
f 0 4 5
f 0 5 1
f 1 5 7
f 1 7 3
```

Định dạng file OBJ

https://en.wikipedia.org/wiki/Wavefront_.obj_file

Đồ họa 2D và Đồ họa 3D

Đồ họa 2D – Đồ họa 2 chiều

Khái niệm

Đồ họa 2D là hệ thống biểu diễn hình ảnh trên **mặt phẳng hai chiều**, sử dụng:

- Trục x (ngang)
- Trục y (dọc)

Không có chiều sâu (z), do đó:

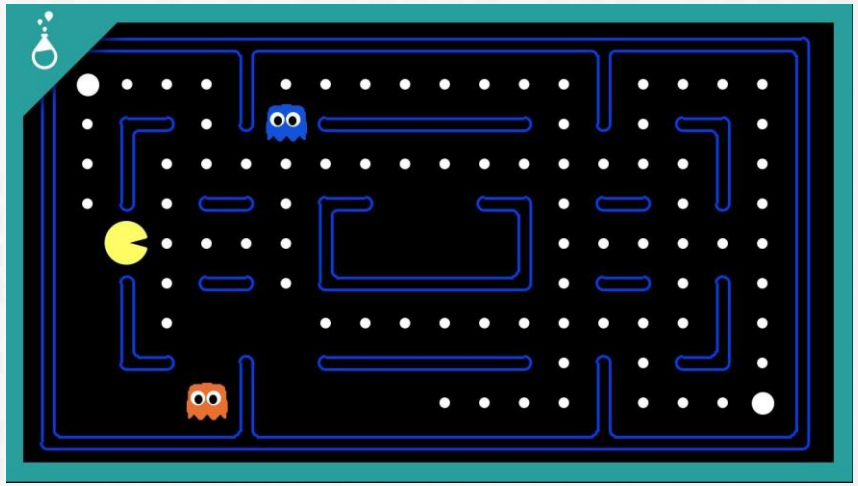
- Mọi đối tượng đều nằm trên mặt phẳng phẳng
- Không có phối cảnh thực

Đồ Họa 2D trong game

Đồ họa 2D là nền tảng của nhiều tựa game kinh điển và vẫn giữ vững vị thế trong ngành công nghiệp game hiện đại. Nó tập trung vào việc tạo ra hình ảnh trên một mặt phẳng, mang lại phong cách nghệ thuật độc đáo và trải nghiệm chơi game riêng biệt.

Đặc Điểm Đồ Họa 2D

- Làm việc trong **không gian 2 chiều** (chiều rộng và chiều cao).
- Không có chiều sâu thật sự, các đối tượng được sắp xếp theo lớp để tạo cảm giác gần/xa.
- Các trò chơi 3D cũng không thể tách rời đồ họa 2D



Khái niệm về Sprite

Sprite là một khái niệm trung tâm trong đồ họa 2D, là các hình ảnh đại diện cho từng phần tử trong game nhân vật, vật thể hoặc môi trường. Các hoạt ảnh của sprite được lưu dưới dạng các Sprite Sheet giúp game 2D trở nên sống động và tăng tính tương tác.

Đại Diện Nhân Vật

Sprite thường được dùng để hiển thị các nhân vật chính và phụ, với nhiều tư thế và hoạt ảnh khác nhau.

Vật Thể & Môi Trường

Các vật phẩm, chướng ngại vật, hoặc các yếu tố môi trường tĩnh và động cũng được tạo từ sprite.

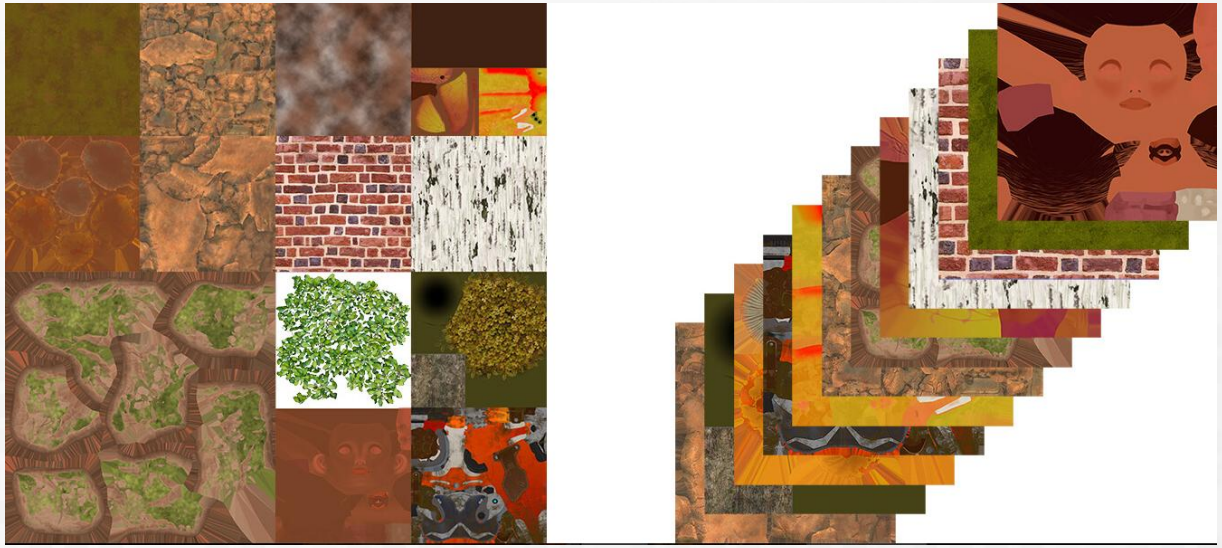
Linh Hoạt Biến Đổi

Sprite có thể được di chuyển, thay đổi kích thước (scale) và xoay (rotate) để tạo ra các hiệu ứng hình ảnh và tương tác đa dạng trong game.

Sprite Sheet

Sprite sheet là một hình ảnh lớn chứa nhiều Sprite nhỏ bên trong, mỗi một sprite có mục đích sử dụng khác nhau

- Nhỏ gọn, tiết kiệm dung lượng
- Tối ưu trong quá trình đọc
- Có rất nhiều ứng dụng trong thiết kế và phát triển game



Sprite Sheet - Animation 2D

Sprite sheet lưu trữ một chuỗi các sprite liên tiếp nhau của một hoạt ảnh 2D cùng một hình ảnh. Mỗi sprite sẽ tương ứng với một frame của hoạt ảnh.

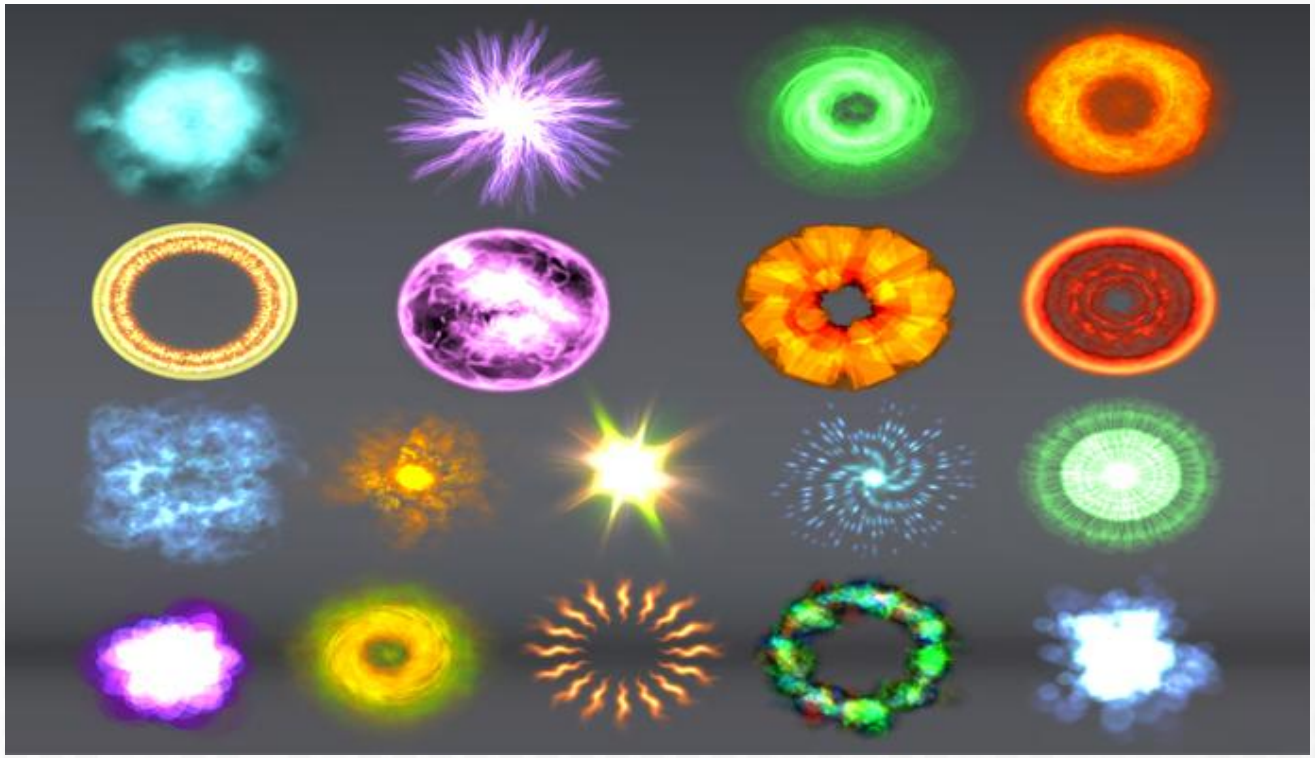
Sprite Sheet - Tile và Background

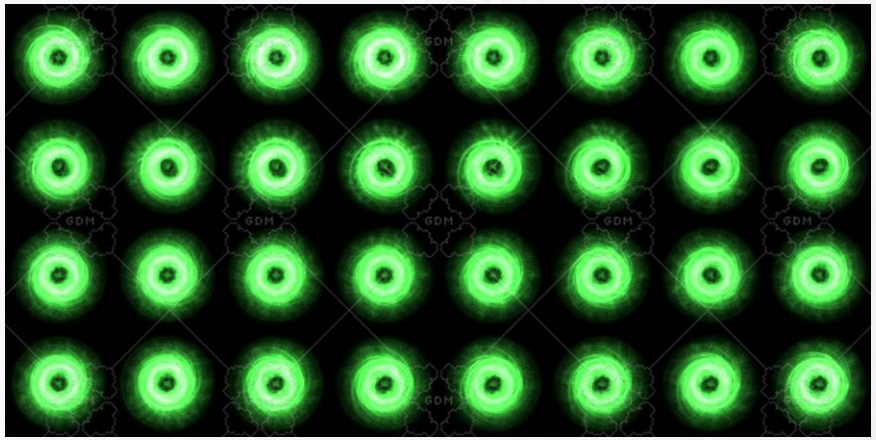
Kỹ thuật Tile và Background là nền tảng cho việc xây dựng môi trường trong game 2D, giúp tối ưu hóa tài nguyên và tạo ra những thế giới rộng lớn một cách hiệu quả.

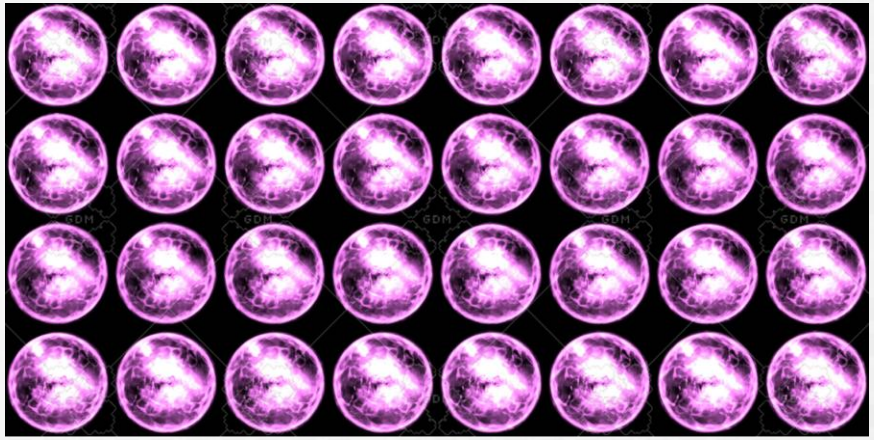
- **Sử dụng ảnh lặp (tiles):** Các mảnh hình ảnh nhỏ (tiles) được sắp xếp cạnh nhau để tạo thành các cấu trúc lớn hơn như tường, sàn nhà, hoặc địa hình.
- **Tiết kiệm bộ nhớ:** Thay vì vẽ toàn bộ background dưới dạng một hình ảnh lớn, việc tái sử dụng các tiles nhỏ giúp giảm đáng kể lượng bộ nhớ cần thiết.
- **Phổ biến trong game cổ điển và casual:** Kỹ thuật này đã được sử dụng rộng rãi trong các game NES, SNES và vẫn rất hữu ích cho các game indie và di động hiện đại.

> “Không phải game đơn giản thì là 2D, và không phải game 2D thì đơn giản.”

Sprite Sheet – VFX, Partical System







ĐỒ HỌA 3D

Khái niệm

Đồ họa 3D là phương pháp biểu diễn đối tượng trong không gian ba chiều, sử dụng ba trục:

- X – chiều ngang
- Y – chiều dọc
- Z – chiều sâu

Không giống 2D, đồ họa 3D:

- ✓ Không hiển thị trực tiếp
- ✓ Phải chiếu (project) lên màn hình 2D

→ 3D graphics là nghệ thuật đánh lừa thị giác thông qua toán học.

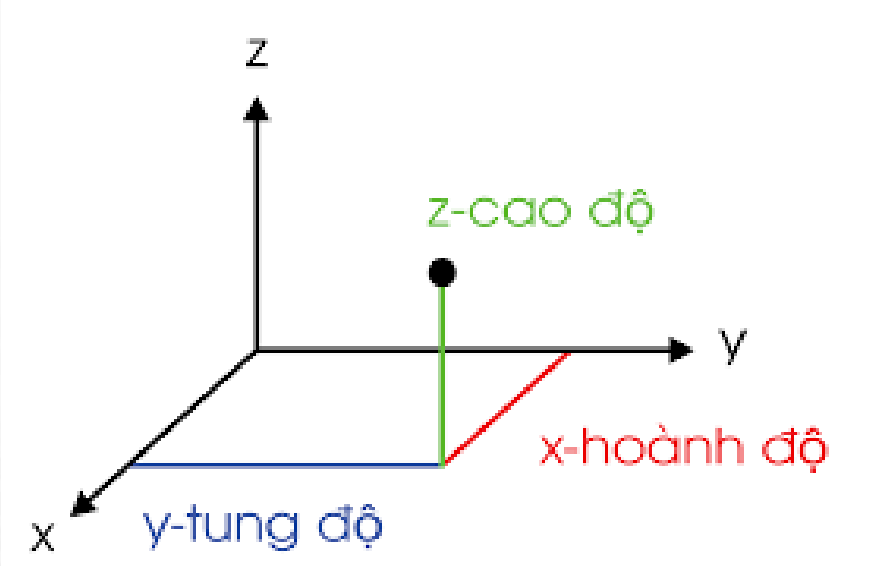
Không gian tọa độ 3D

Hệ tọa độ Descartes 3D

Mỗi điểm được xác định bởi (x, y, z)

Dùng cho:

- Vị trí đỉnh (vertex)
- Vị trí đối tượng
- Camera và ánh sáng



Đồ Họa 3D trong Game

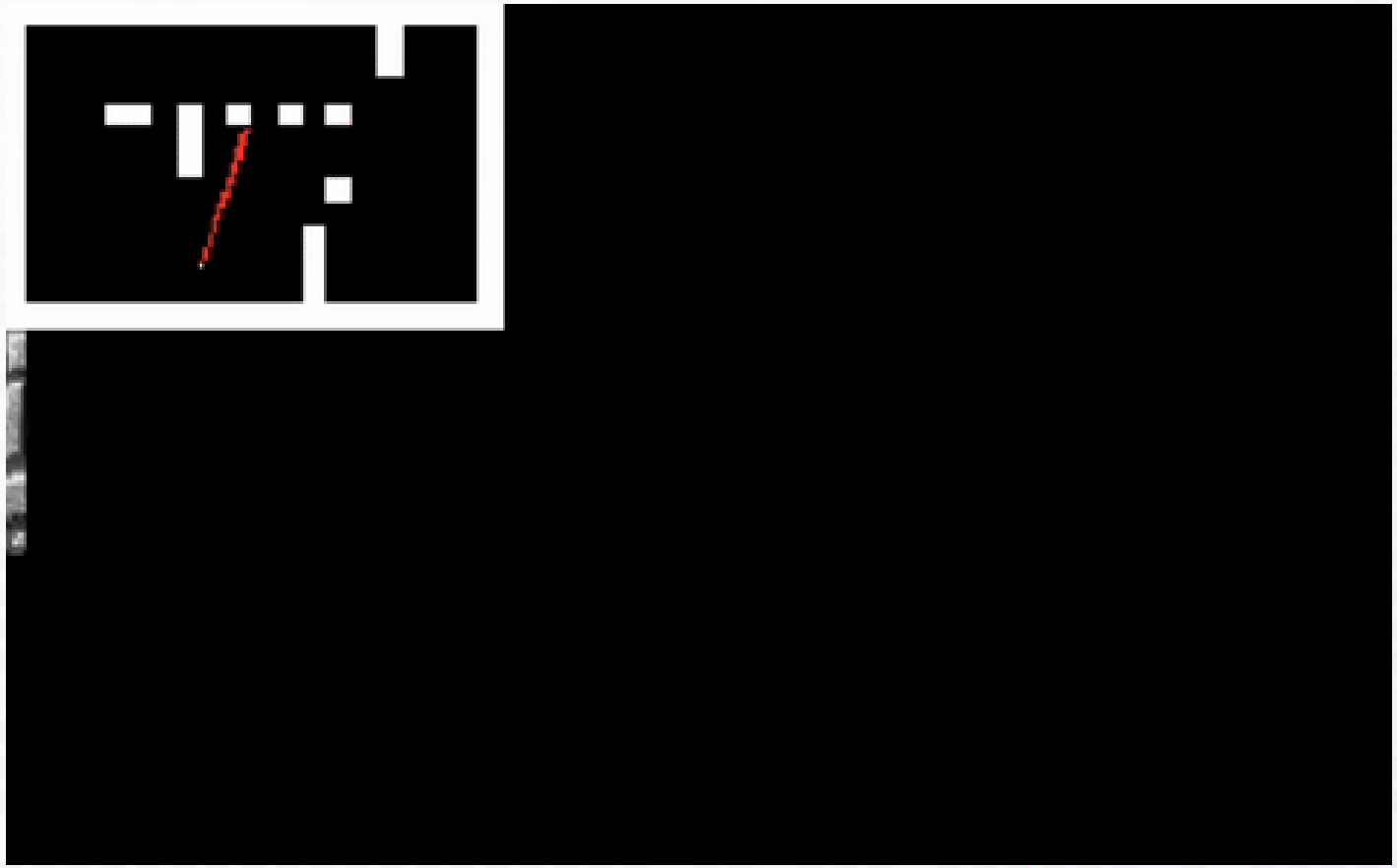
Đồ họa 3D đã cách mạng hóa ngành công nghiệp game, mang đến những trải nghiệm hình ảnh sống động, chân thực và khả năng tương tác sâu sắc mà đồ họa 2D khó lòng sánh được. Từ những tựa game bắn súng góc nhìn thứ nhất đến các thế giới mở rộng lớn, 3D graphics đã định hình lại cách chúng ta chơi và trải nghiệm game.

Không Gian 3D

- **Sử dụng các mô hình toán học trong không gian 3 chiều:** Các vật thể được xây dựng từ lưới đa giác (polygon mesh) trong hệ tọa độ X, Y, Z, cho phép biểu diễn các đối tượng với chiều sâu và khối lượng.
- **Cho phép người chơi thay đổi góc nhìn tự do:** Khả năng xoay camera, di chuyển trong môi trường 3D mang lại cảm giác nhập vai và kiểm soát cao hơn.
- **Tạo ra sự chân thực nhờ ánh sáng và đổ bóng động:** Hệ thống chiếu sáng và đổ bóng phức tạp mô phỏng thế giới thực, tăng tính thẩm mỹ và độ tin cậy của hình ảnh.

HIỂN THỊ HÌNH ẢNH 3D TRÊN MÀN HÌNH 2D

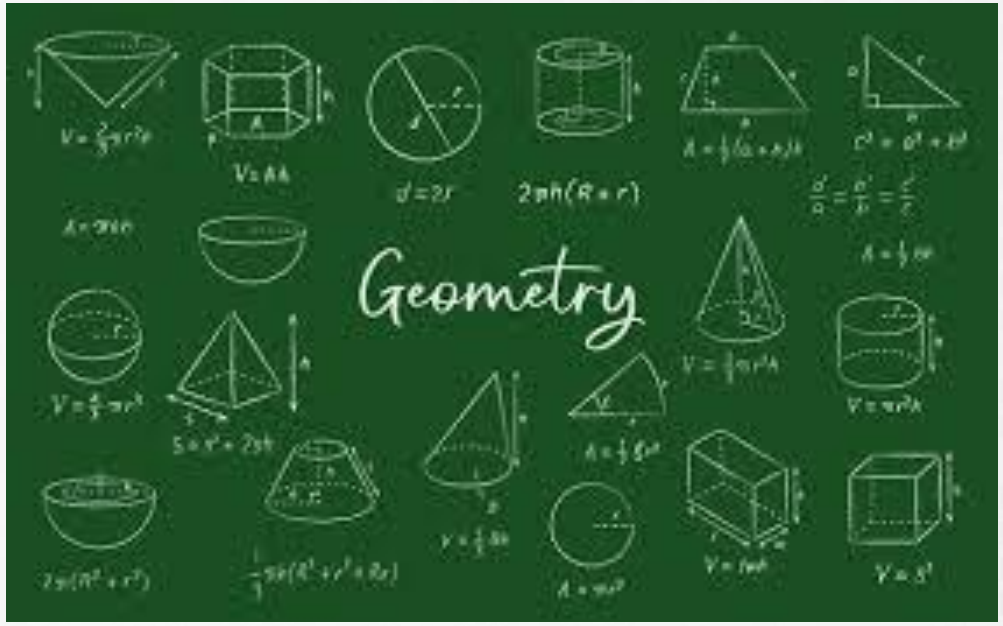
- Dựa vào khoảng cách từ camera đến các đối tượng 3D để vẽ lại hình ảnh 3D trên màn hình
- RayTracing, Rasterization



Hình học và các hình khối cơ bản

Hình học (Geometry) trong đồ họa máy tính

- Trong đồ họa máy tính, hình học dùng để mô tả **hình dạng và cấu trúc không gian** của đối tượng
- Hình học không mô tả màu sắc hay ánh sáng, mà chỉ tập trung vào:
  - Vị trí
  - Hình dạng
  - Kích thước
  - Quan hệ không gian giữa các phần tử



Các hình khối cơ bản Primitives

- **Primitive** là **đơn vị hình học cơ bản nhất** mà hệ thống đồ họa có thể xử lý trực tiếp.
- Mọi đối tượng phức tạp đều được xây dựng từ tập hợp các primitive.
- Trong hệ thống đồ họa, GPU:
  - Không hiểu “đối tượng”
  - Không hiểu “mô hình”
  - Chỉ xử lý **primitive**

Điểm (Point)

- Point biểu diễn **một vị trí trong không gian**
- Không có kích thước
- Không có diện tích

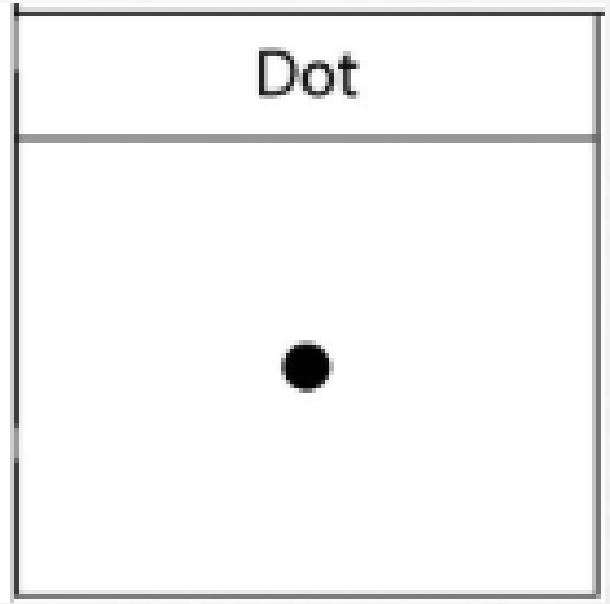
- Trong thực tế hiển thị:

  Một point thường được vẽ thành **một pixel**

- Point là:

  Thành phần cơ bản nhất

  Cơ sở để xây dựng line và polygon



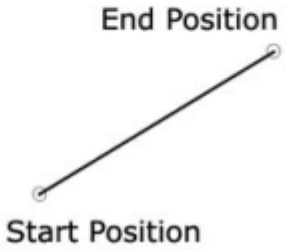
Đoạn thẳng (Line)

- Line được xác định bởi **hai point**
- Biểu diễn một đoạn thẳng trong không gian

- Line:
- Có chiều dài
- Không có diện tích

- Trong đồ họa:
- Line thường dùng cho:
  - Wireframe
  - Minh họa hình học
- Không phù hợp để biểu diễn bề mặt vật thể

```
struct Point
{
   int x;
   int y;
}

struct Line
{
   Point start;
   Point end;
}
```



Đa giác (Polygon)

Polygon là:
- Một tập hợp các line khép kín
- Tạo thành một bề mặt phẳng

Polygon có:
- Diện tích
- Hướng (orientation)

```
struct Polygon
{
   int total_points;
   array<Point> points;
}
```

FIGURE 2.10 A simple polygon (left) and a complex polygon (right).

Tam giác (Triangle)

- Triangle được tạo từ **ba point**
- Ba point **luôn xác định một mặt phẳng**
- Triangle không bị suy biến hình học như polygon nhiều cạnh

Do đó:

- Triangle là primitive ổn định
- Triangle là nền tảng của render 3D

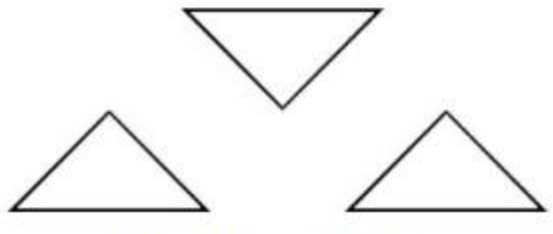
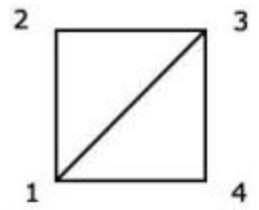


FIGURE 2.12 Triangle lists.



Triangle 1:
{ 1, 2, 3 }

Triangle 2:
{ 3, 4, 1 }

FIGURE 2.13 Using indices to specify triangles.

Hình cầu (Sphere) và hình hộp (Boxs)

- Một **sphere** được xác định bởi:
- **Tâm (center point)**
- **Bán kính (radius)**

- Về mặt hình học:
- Mọi điểm trên bề mặt sphere đều cách tâm một khoảng bằng bán kính

- Sphere là đối tượng:
- Đối xứng theo mọi hướng
- Không phụ thuộc vào hướng quay

```
struct Sphere
{
   int radius;
   Point position;
}
```

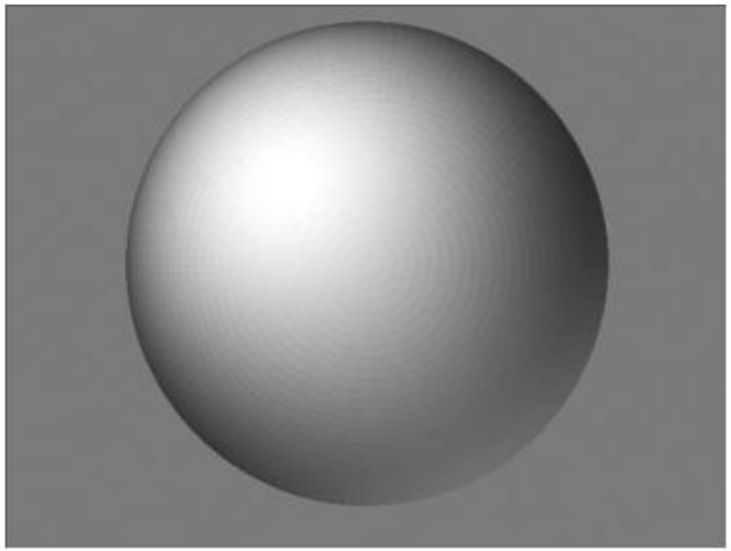


FIGURE 2.17 A sphere.

Hình hộp (Boxs)

- Một **box** là đối tượng hình học:
  - Có sáu mặt phẳng
  - Các mặt vuông góc với nhau
- Trong đồ họa, box thường được hiểu là:

Hình hộp chữ nhật (rectangular box)

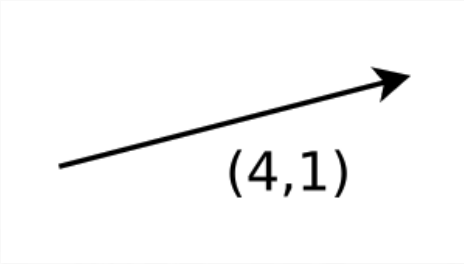
```
struct Box
{
    int width, height, depth;
    Point position;
}

struct Cube
{
    int size;
    Point position;
}
```

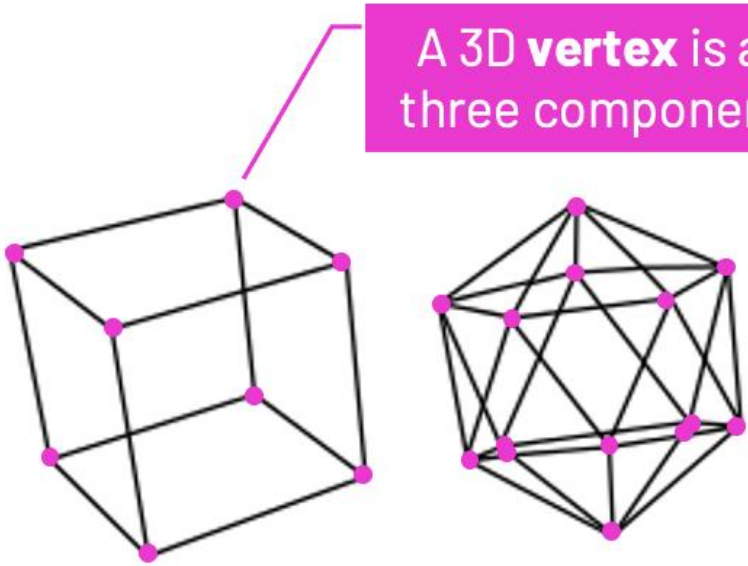
Các khái niệm toán học trong đồ họa máy tính

VECTOR

Vector

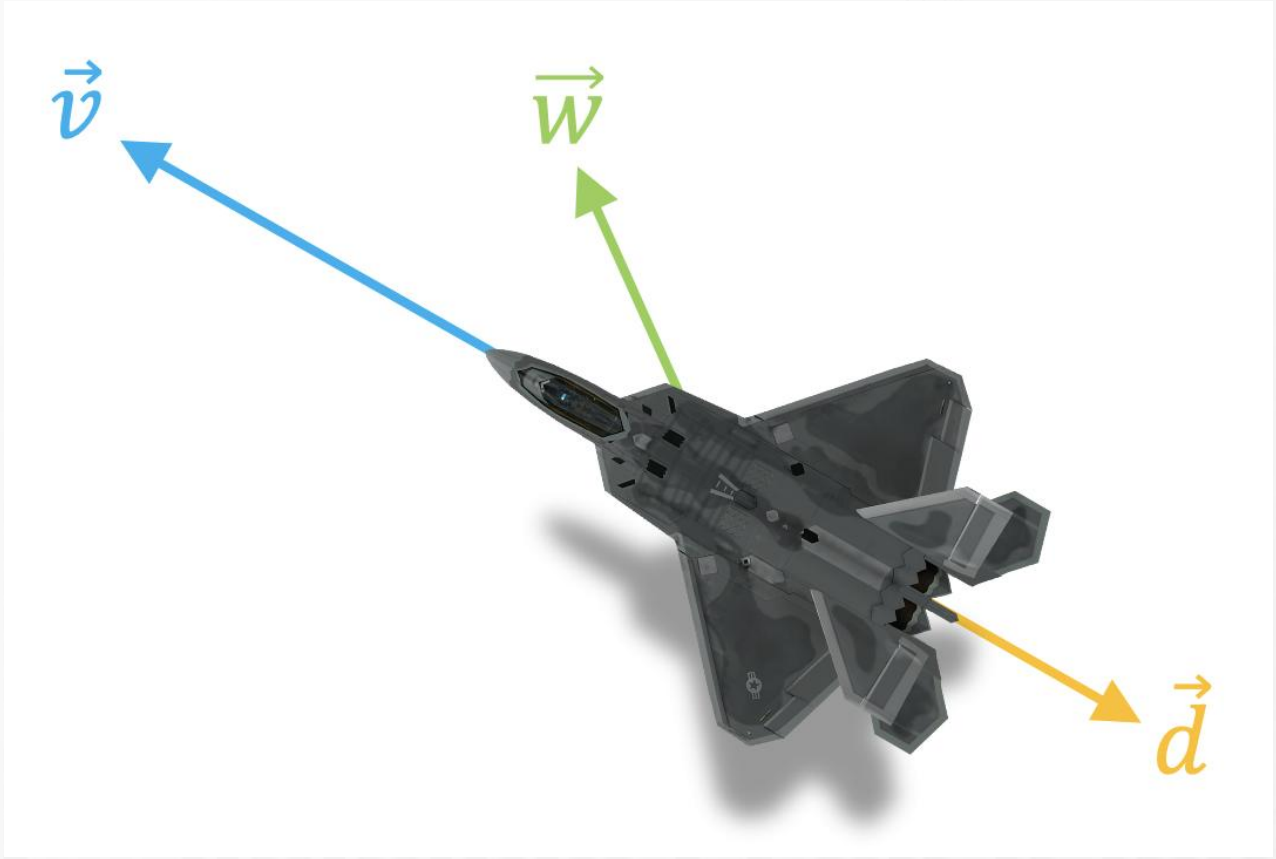
- Là đối tượng toán học cơ bản được sử dụng trong hầu hết các game và game engine
- Phần lớn các vector trong lập trình game là 2D (x,y), 3D (x,y,z), trong một số trường hợp đặc biệt là 4D (x,y,z,w).





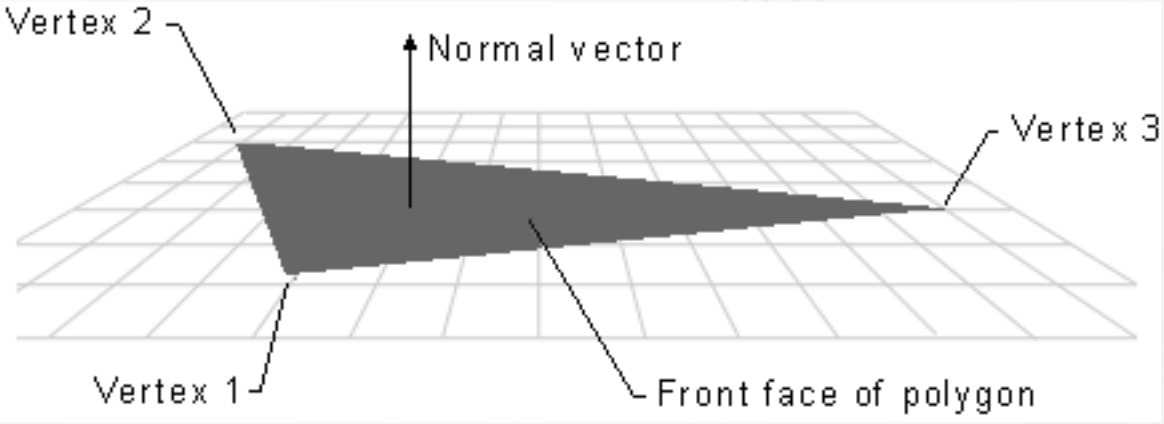
Vector

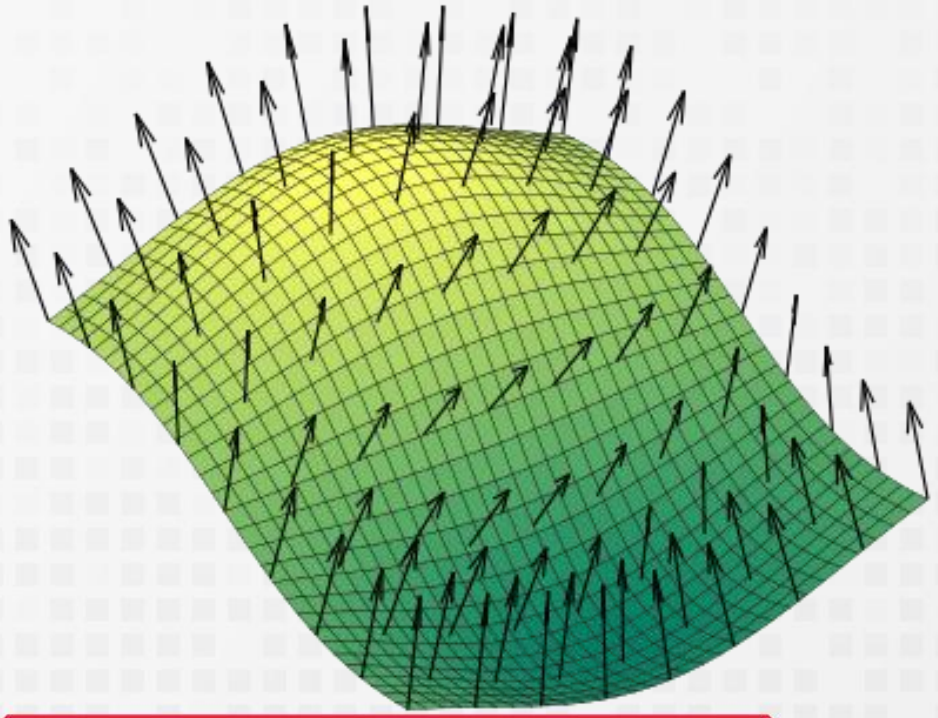
- Dùng để biểu diễn một lực, vận tốc, gia tốc, ma sát,... của đối tượng trong thế giới game
- Dễ thực hiện các phép cộng, trừ, nhân vô hướng,...



Vector

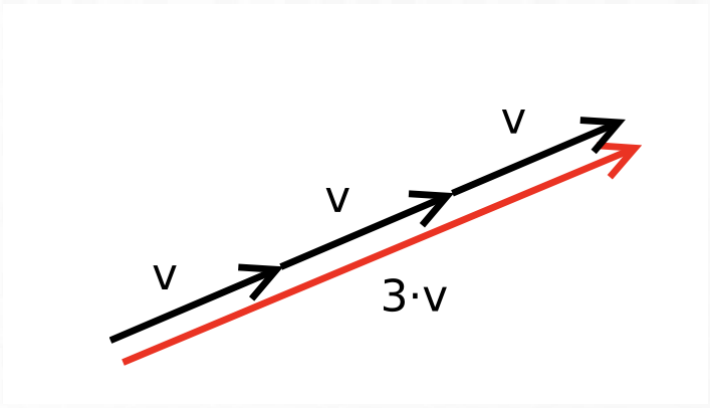
- Còn được sử dụng để biểu diễn các bề mặt 3D. Trong đó mỗi phần tử của bề mặt là một đa giác, mỗi đa giác này có một normal vector biểu diễn hướn của bề mặt

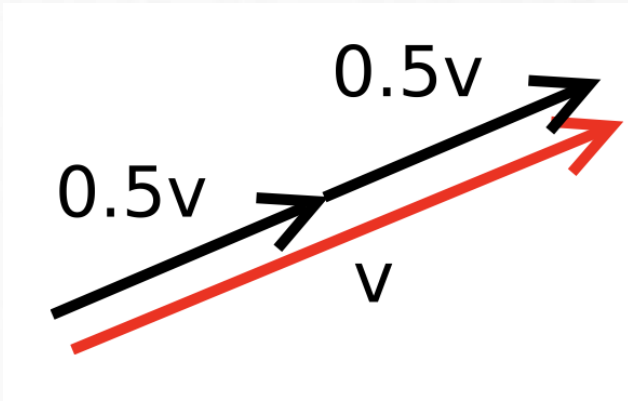


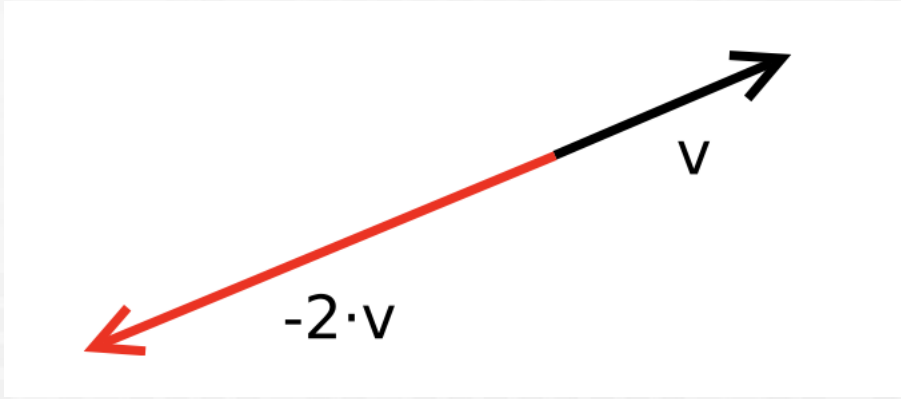


Vector

- Phép nhân
  - Tăng giảm độ lớn vector (scaling)
  - Đảo hướng vector (xoay 180 độ)





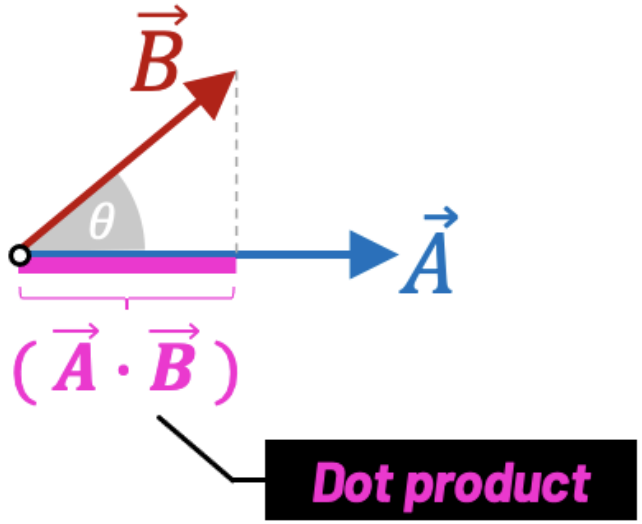


Tích vô hướng Dot - Product

Tích vô hướng

- Được sử dụng để tính độ dài của một vector khi chiếu lên một vector khác
- Được sử dụng để so sánh về hướng giữa 2 vector:
  - Cùng hướng nếu tích > 0
  - Vuông góc nếu tích = 0
  - Ngược hướng nếu tích < 0

$$\vec{A} \cdot \vec{B} = |\vec{A}||\vec{B}| \cos\theta$$

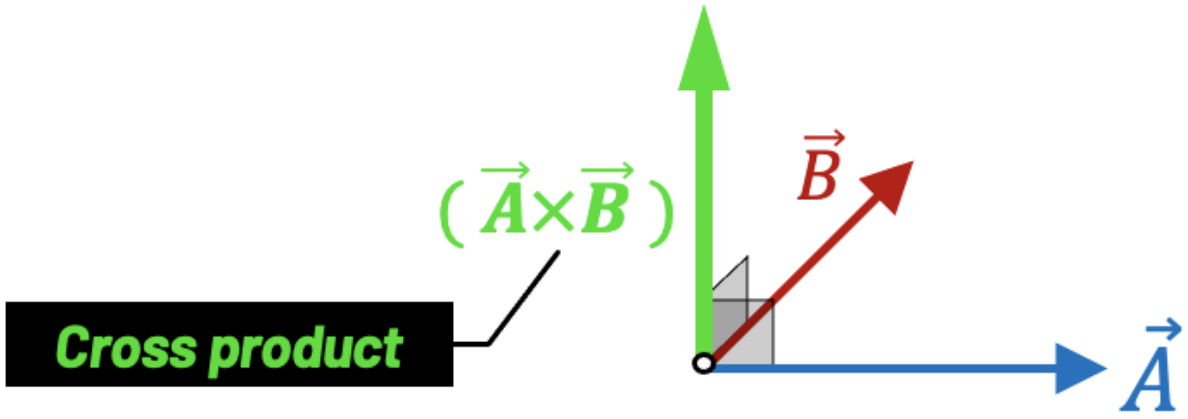


Tích có hướng (Cross – Product)

Dùng để tính vector pháp tuyến của mặt phẳng được đại diện bởi 2 vector cho trước.

$$\vec{A} \times \vec{B} = |\vec{A}||\vec{B}| \sin\theta$$

Tập các vector pháp tuyến sử dụng để biểu diễn các bề mặt cong.



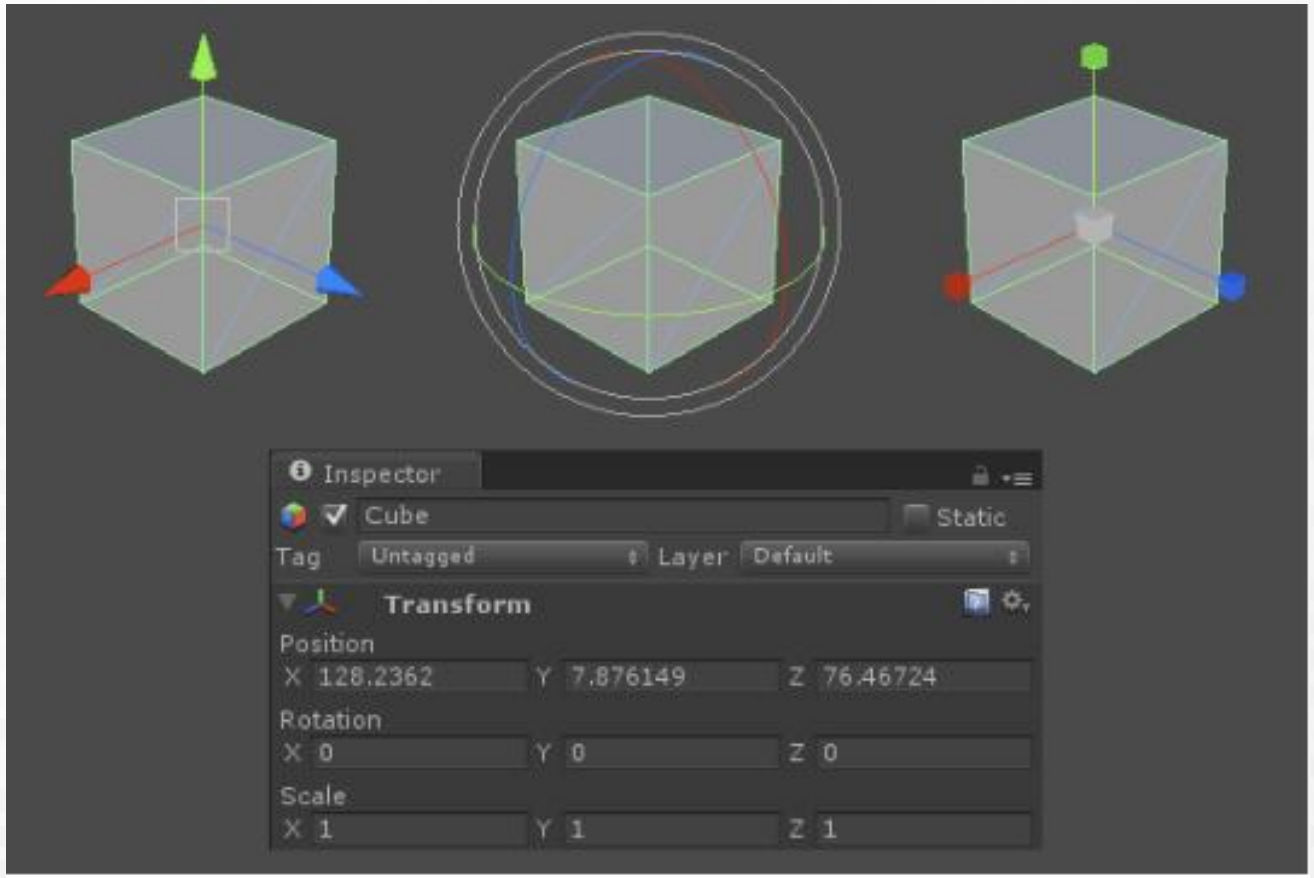
MA TRẬN

CÁC ỨNG DỤNG

Là công cụ được sử dụng rất rộng rãi trong game, cụ thể:

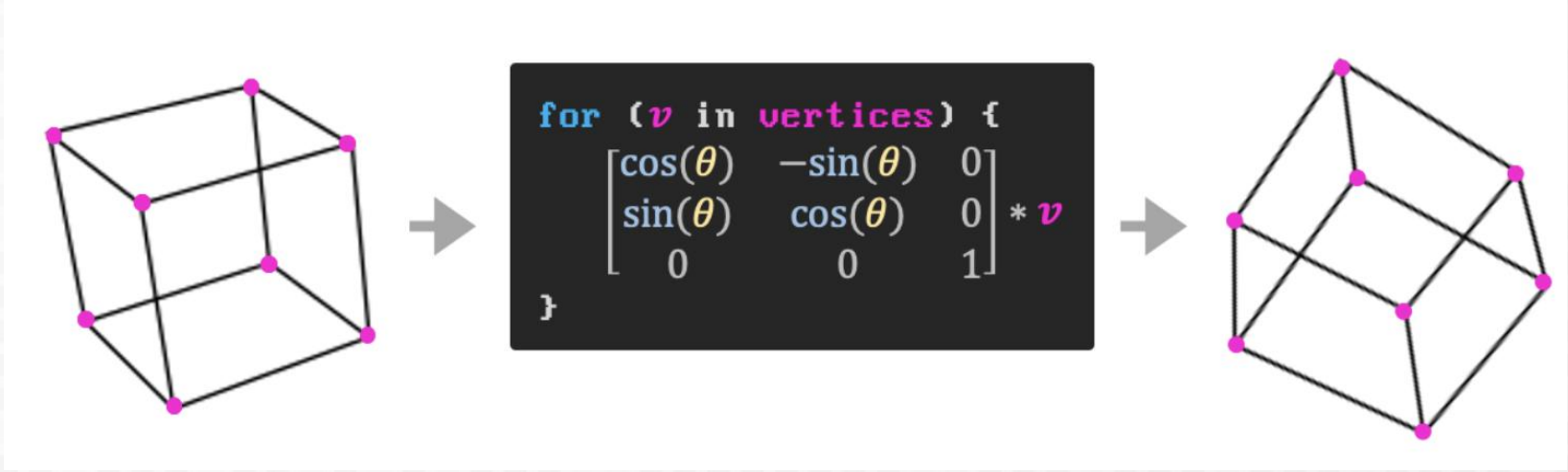
- Phép chiếu, chuyển vị, tỉ lệ cho các đối tượng hình
- Thực hiện phép xoay đối tượng trong không gian
- Chuyển đổi các hệ màu
- Tạo dữ liệu mật mã

CÁC ỨNG DỤNG


Inspector
Cube
Static
Tag
Untagged
Layer
Default
Transform
Position
X 128.2362
Y 7.876149
Z 76.46724
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1

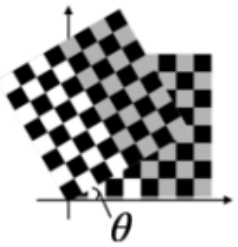
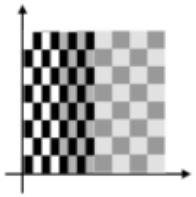
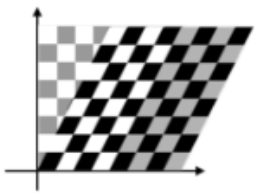
PHÉP XOAY

- Nhân từng đỉnh của vật thể với ma trận biến đổi (rotate, scale, translate) để tính toán tọa độ mới của đỉnh



ỨNG DỤNG PHÉP BIẾN ĐỔI AFFINE

- 3 ứng dụng chính: Xoay (rotation), tỉ lệ (scale), chuyển vị (translation)

Name	Rotate	Translate	Scale	Shear
M	$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Example				

$$\begin{bmatrix} Translation\ Matrix \end{bmatrix} * \begin{bmatrix} Rotation\ Matrix \end{bmatrix} * \begin{bmatrix} Scale\ Matrix \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

BIỂU DIỄN SỰ CHUYỂN ĐỔI TRỤC TỌA ĐỘ

- Là tập hợp các phép xoay, dịch chuyển và tỉ lệ để chuyển đổi đối tượng giữa 2 không gian toạ độ.
- Được sử dụng bởi các engines