

Đồ họa



Tuần 1

Giảng viên: Trần Đức Minh

Nội dung bài giảng



- Đồ họa máy tính là gì ?
- Các ứng dụng đồ họa máy tính
- Hệ thống đồ họa máy tính
- Tạo lập hình ảnh
- Kiến trúc đường ống đồ họa
- Phương pháp lập trình đồ họa
- Các công cụ hỗ trợ lập trình đồ họa trong khóa học
- Ví dụ



Đồ họa máy tính là gì ?



- Đồ họa máy tính giải quyết tất cả những khía cạnh của việc **tạo ra hình ảnh bằng máy tính**.
- Đồ họa máy tính là sự kết hợp giữa 3 thành phần
 - Đối tượng đồ họa: Các đối tượng được sáng tạo bởi người dùng (đồ vật, nhân vật trong games, ...)
 - Phần mềm:
 - Mức cao: Các phần mềm được dùng để tạo dựng hình ảnh như 3DS Studio, Lightwave,
 - Mức thấp: Các thư viện hỗ trợ lập trình đồ họa như DirectX, OpenGL, ...
 - Phần cứng: Máy tính và card đồ họa dùng để lưu trữ và hiển thị các đối tượng đồ họa.

Các ứng dụng đồ họa máy tính



- Biểu diễn thông tin

- Đồ thị, mặt cong thay cho biểu bảng
- Biểu diễn khác



- Ví dụ:

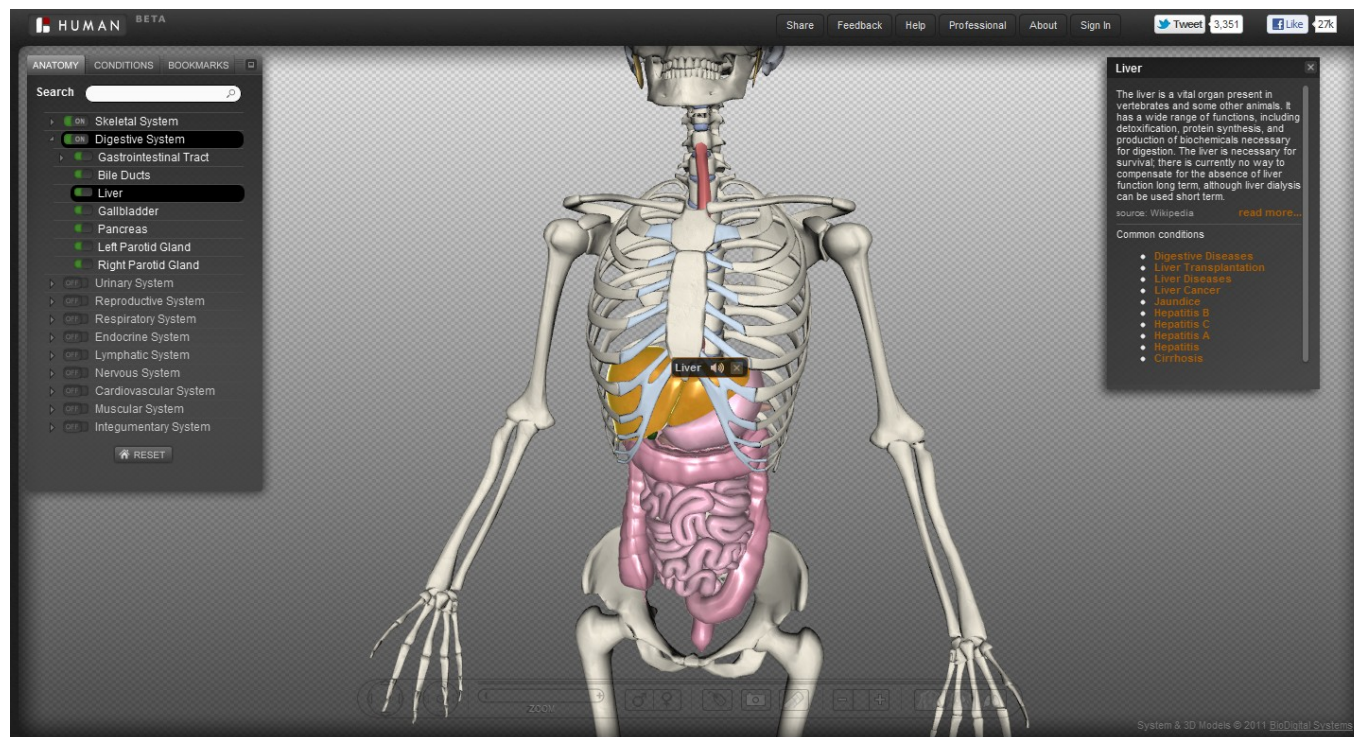
<https://artsexperiments.withgoogle.com/freefall/wave>



Các ứng dụng đồ họa máy tính



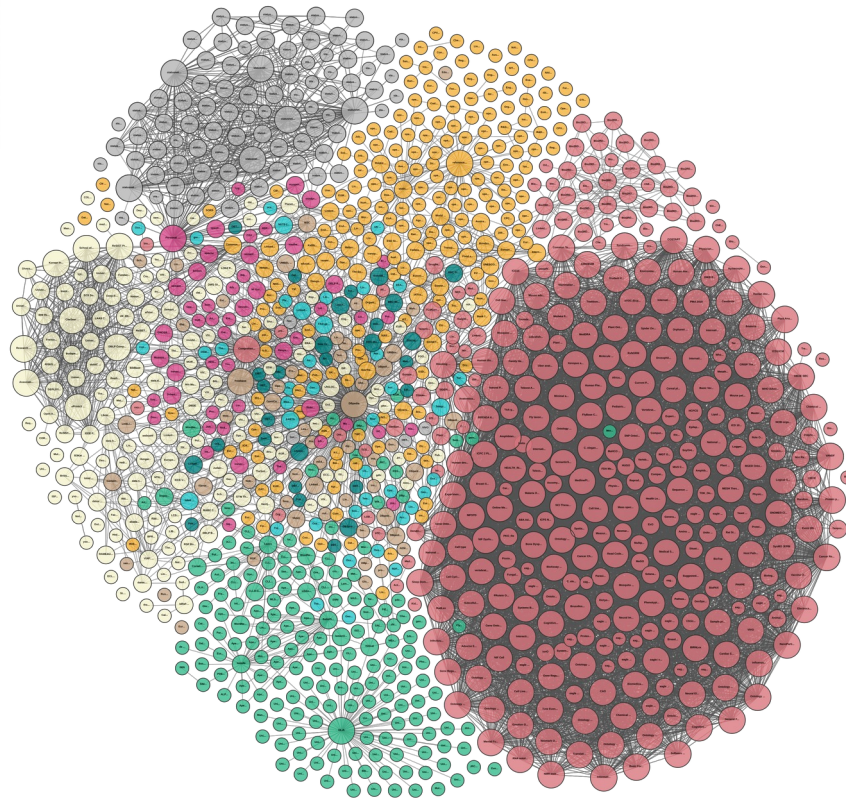
- Hiện thị thông tin
 - Hình ảnh giải phẫu cơ thể người



Các ứng dụng đồ họa máy tính



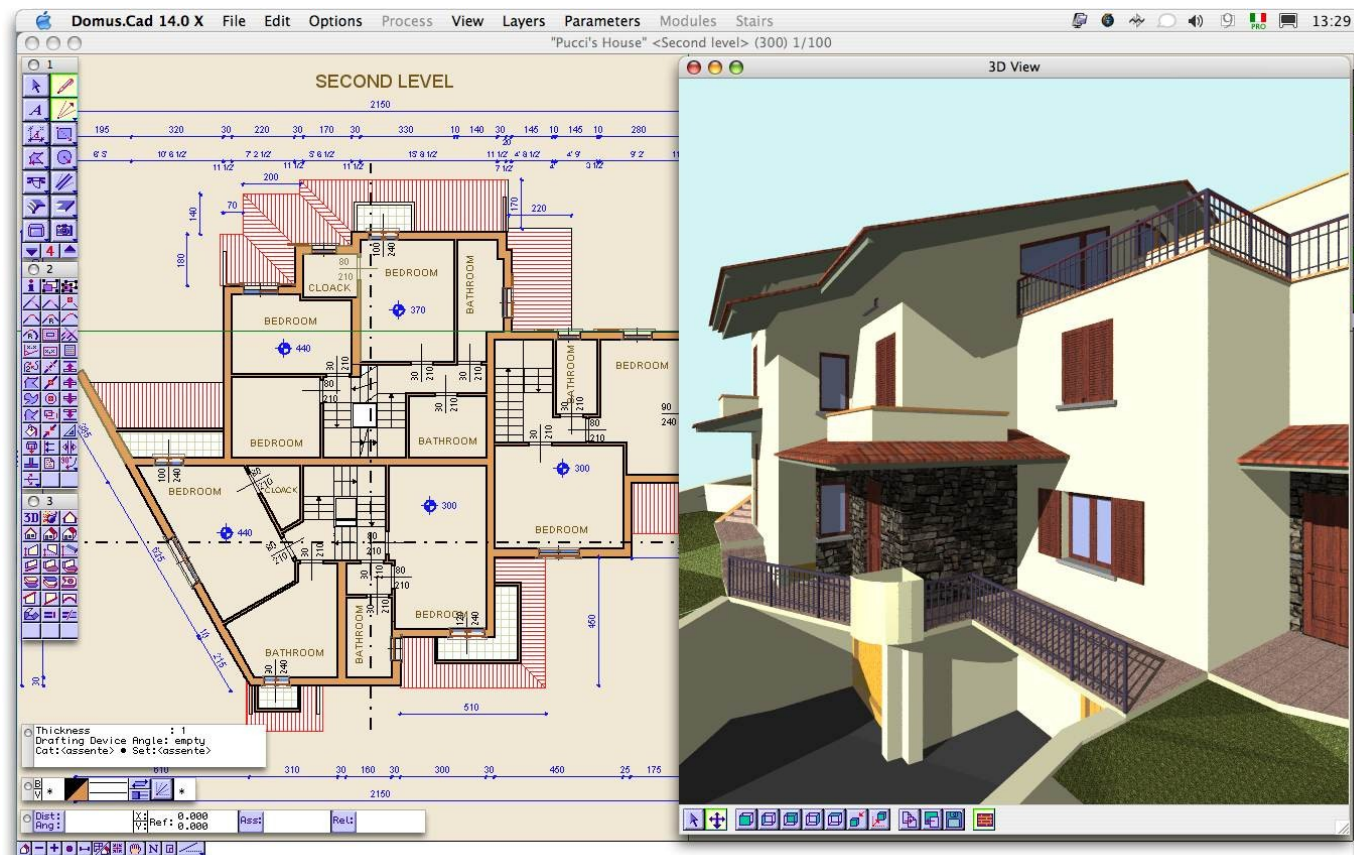
- Hiện thị thông tin
 - Linked Open Data



Các ứng dụng đồ họa máy tính



- Máy tính hỗ trợ thiết kế



Các ứng dụng đồ họa máy tính



- Các hệ thống giả lập



Máy bay



Tramway

Các ứng dụng đồ họa máy tính



- Thực tại ảo



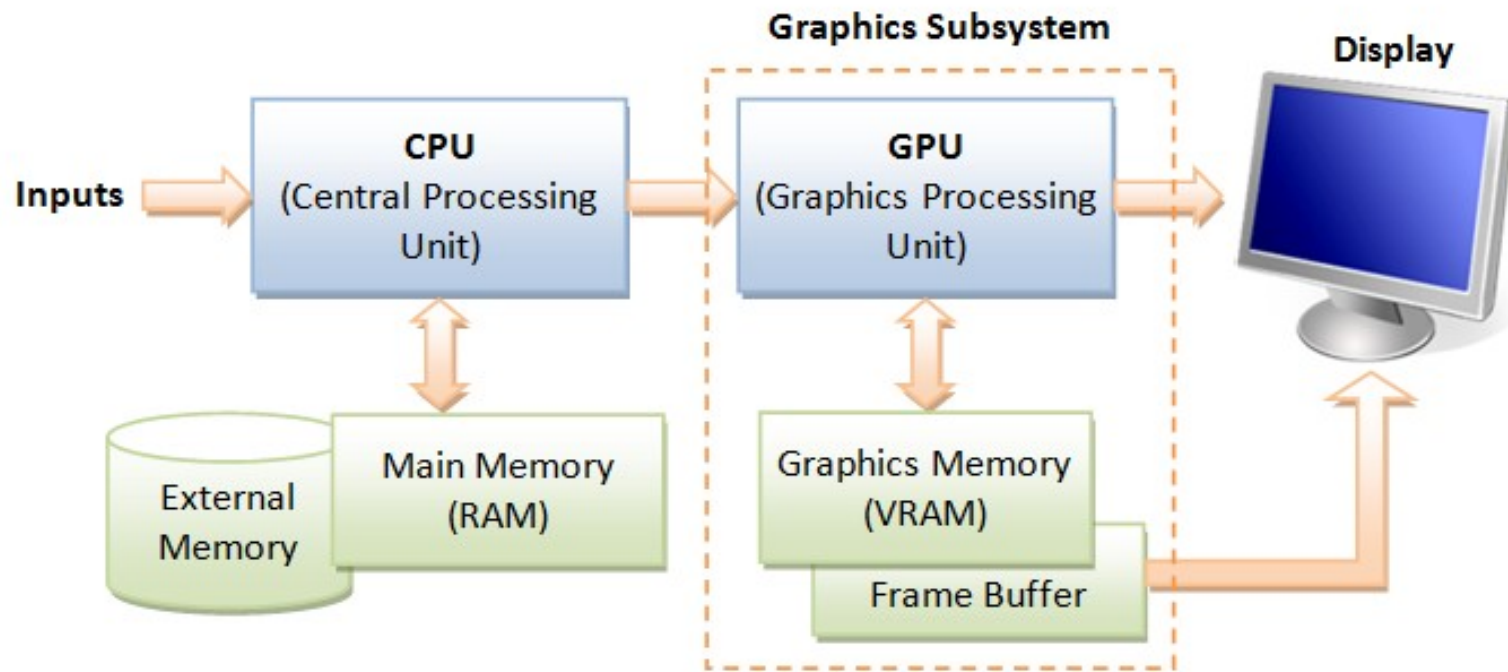
Các ứng dụng đồ họa máy tính



- Augmented reality (thực tế tăng cường)



Hệ thống đồ họa máy tính

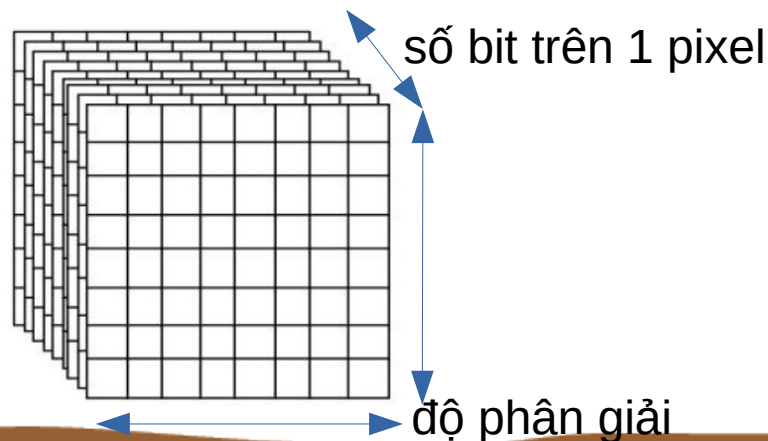


Hình ảnh trong Frame buffer là hình ảnh hiển thị lên màn hình

Frame Buffer



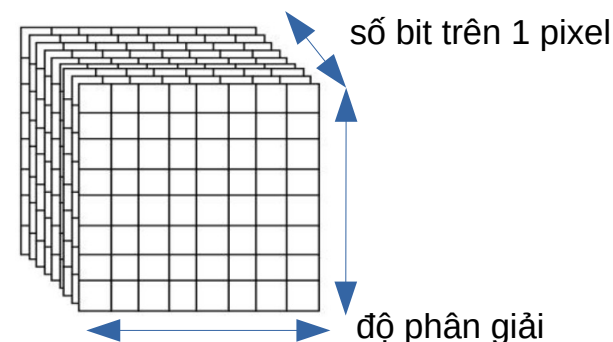
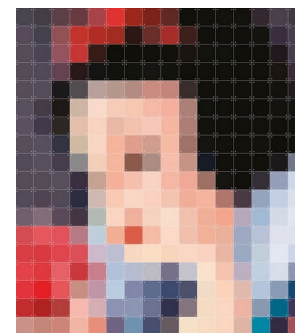
- Nằm trong **bộ nhớ máy tính** hoặc bên trong **GPU** (Graphics Processing Unit).
- Được sử dụng để lưu trữ thông tin các điểm ảnh (pixel) trên màn hình máy tính.
- Độ phân giải của Frame Buffer và độ phân giải của màn hình là giống nhau.



Pixel



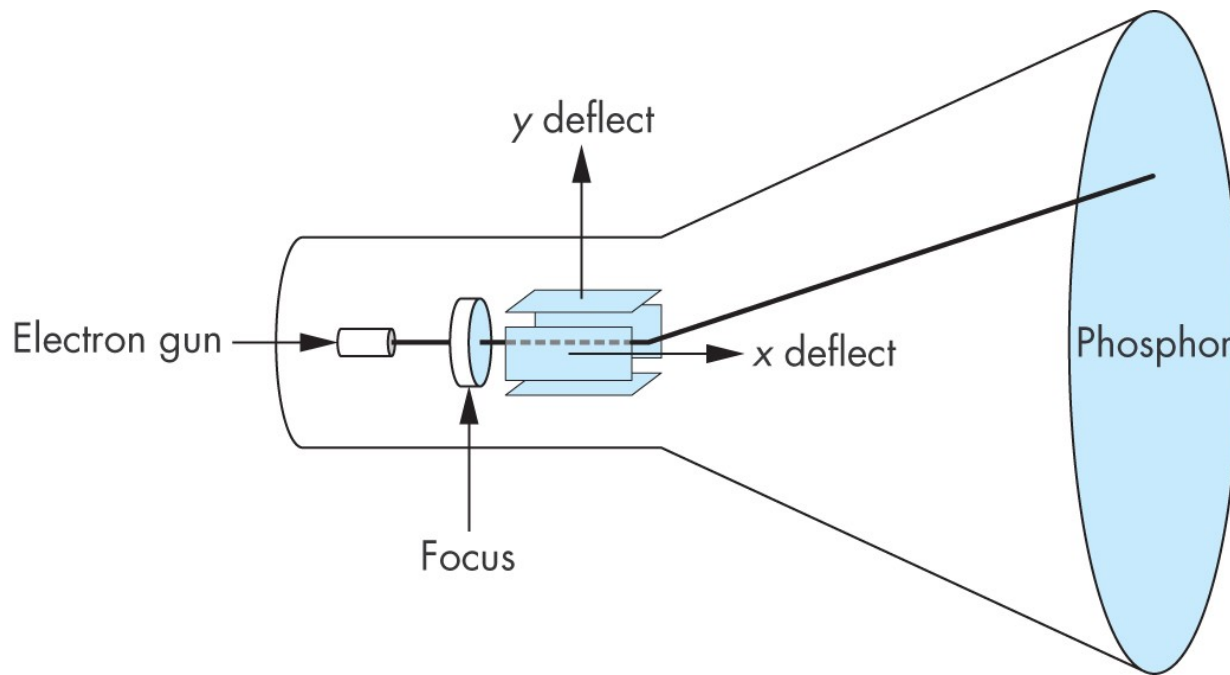
- Pixel là đơn vị nhỏ nhất của ảnh, trong đó ảnh là một mảng 2 chiều của các điểm ảnh.
- Mỗi pixel có 2 thuộc tính
 - Vị trí của pixel - Tọa độ x và y.
 - Giá trị của pixel - Màu của pixel
- Số lượng bit được sử dụng cho mỗi pixel sẽ xác định số lượng màu có thể được tạo ra cho ảnh.
- Ví dụ:
 - 1 bit : Số lượng màu của ảnh là 2
 - 8 bits : Số lượng màu của ảnh là 256



Thiết bị xuất



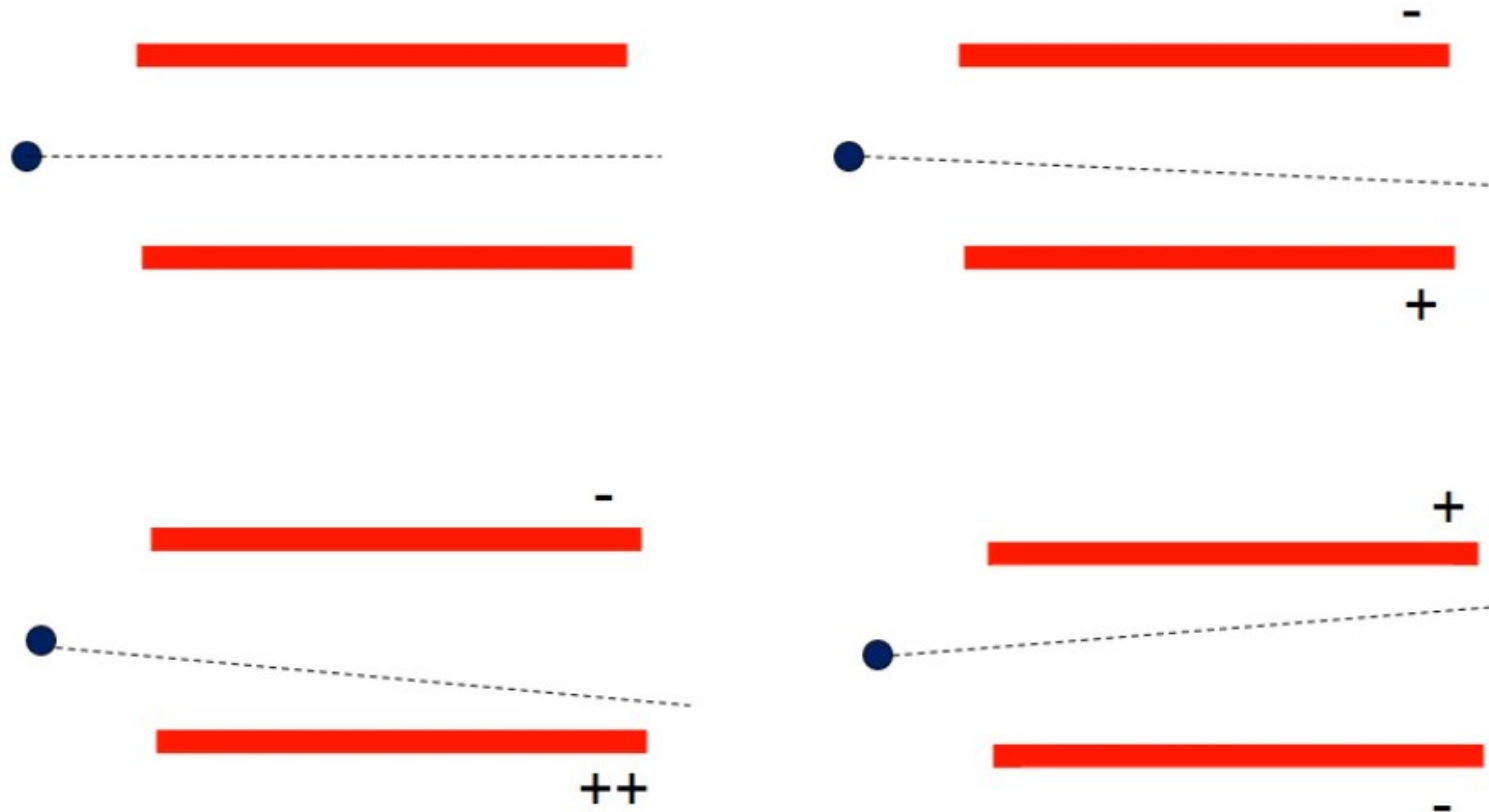
- Cathode Ray Tube (CRT)



Thiết bị xuất



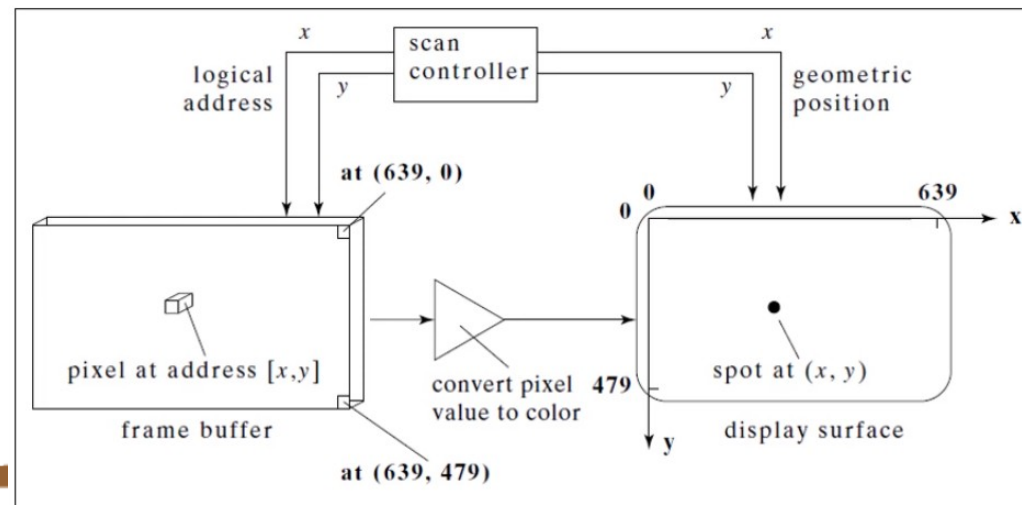
- Cathode Ray Tube (CRT)



Thiết bị xuất



- Cathode Ray Tube (CRT)
 - Cơ chế chuyển đổi dữ liệu từ Frame Buffer ra màn hình hiển thị
 - **Scan controller:** Hướng chùm tia điện tử đến đúng vị trí cần hiển thị điểm ảnh trên màn hình.
 - **Convert pixel:** Chuyển đổi giá trị bên trong Frame buffer thành màu sắc tương ứng lên màn hình.



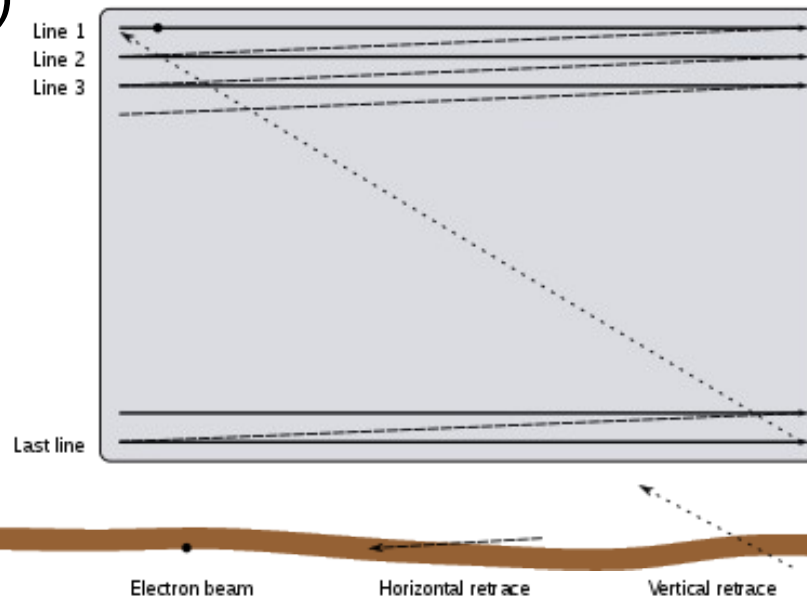
Thiết bị xuất



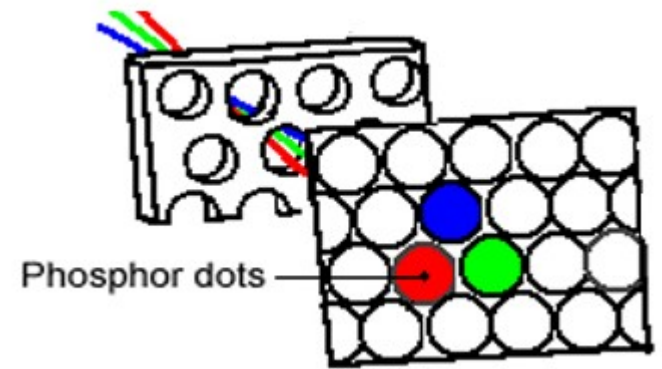
- Cathode Ray Tube (CRT)

- **Scan controller:**

- Do màn hình phosphor chỉ sáng trong một khoảng thời gian nhất định, do đó ta phải làm tươi màn hình liên tục.
 - **Tần số làm tươi:** xHz = Trong 1 giây làm tươi x lần (Mỗi 1 giây thì cần x lần đọc dữ liệu từ Frame Buffer để đưa lên màn hình)



Thiết bị xuất

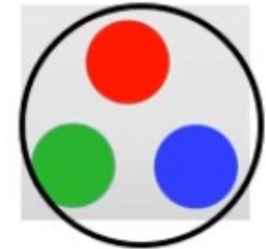
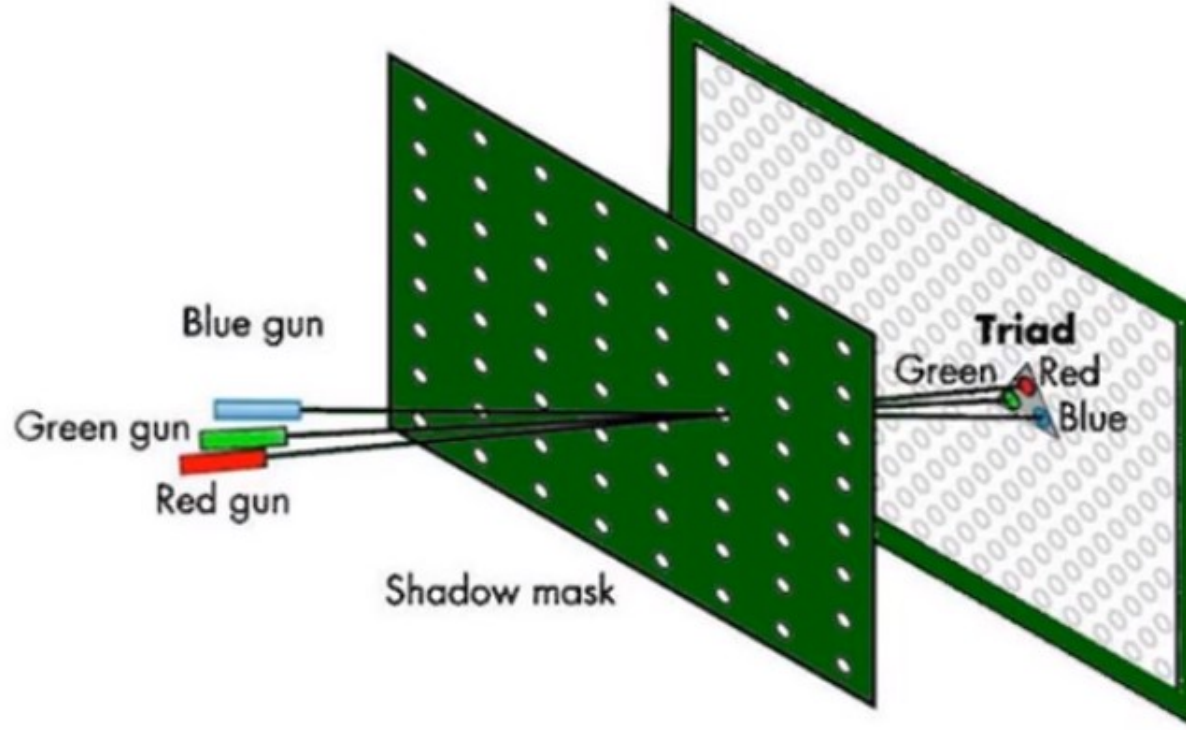


- Màn hình CRT màu
 - Mỗi pixel trên màn hình chứa 3 chấm nhỏ, tương ứng với 3 màu **Red**, **Green** và **Blue**.
 - Màu sắc của pixel sẽ dựa trên cường độ màu của 3 chấm nhỏ này.
 - Ví dụ: Khi cường độ của 3 chấm nhỏ **Red**, **Green** và **Blue** tất cả đều tối đa (đỏ tối đa, xanh lá cây tối đa và xanh da trời tối đa) thì ta sẽ nhận được màu của pixel là **màu trắng**.
 - Scan controller sử dụng 3 súng điện tử.
 - Mỗi súng điện tử có nhiều mức độ bắn khác nhau. Tương ứng với mỗi mức độ, sẽ hiển thị cường độ màu tương ứng đối với chấm nhỏ.

Thiết bị xuất



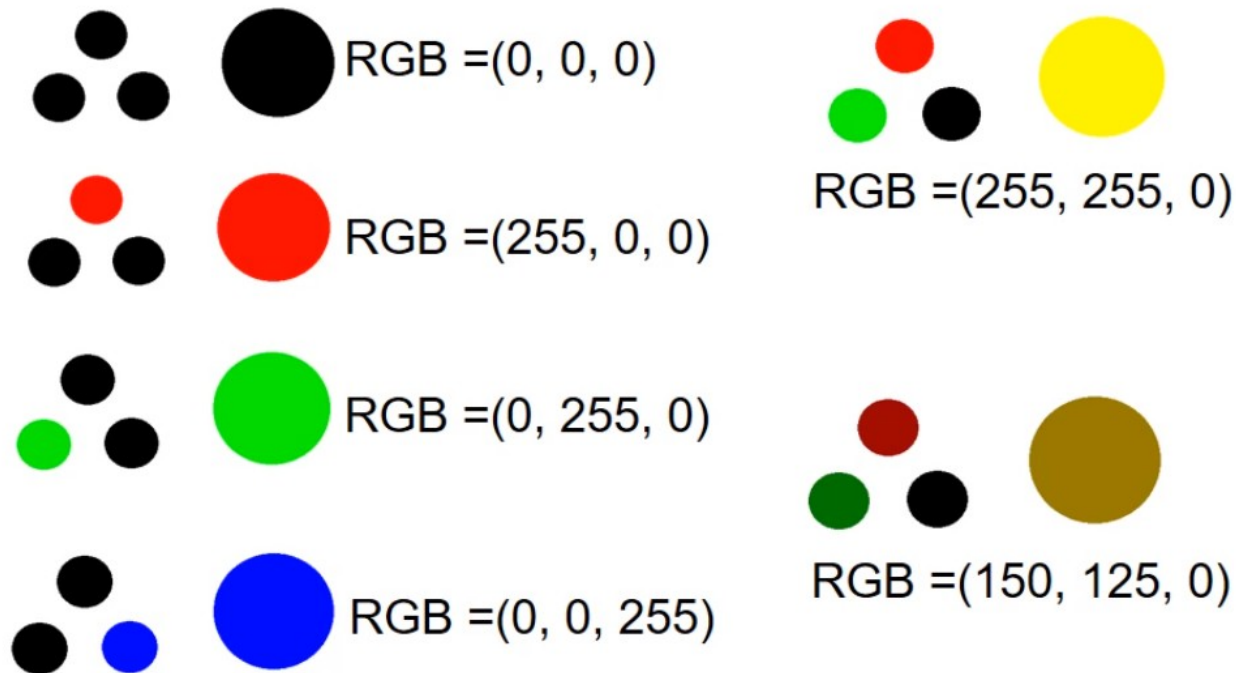
- Màn hình CRT màu



Thiết bị xuất



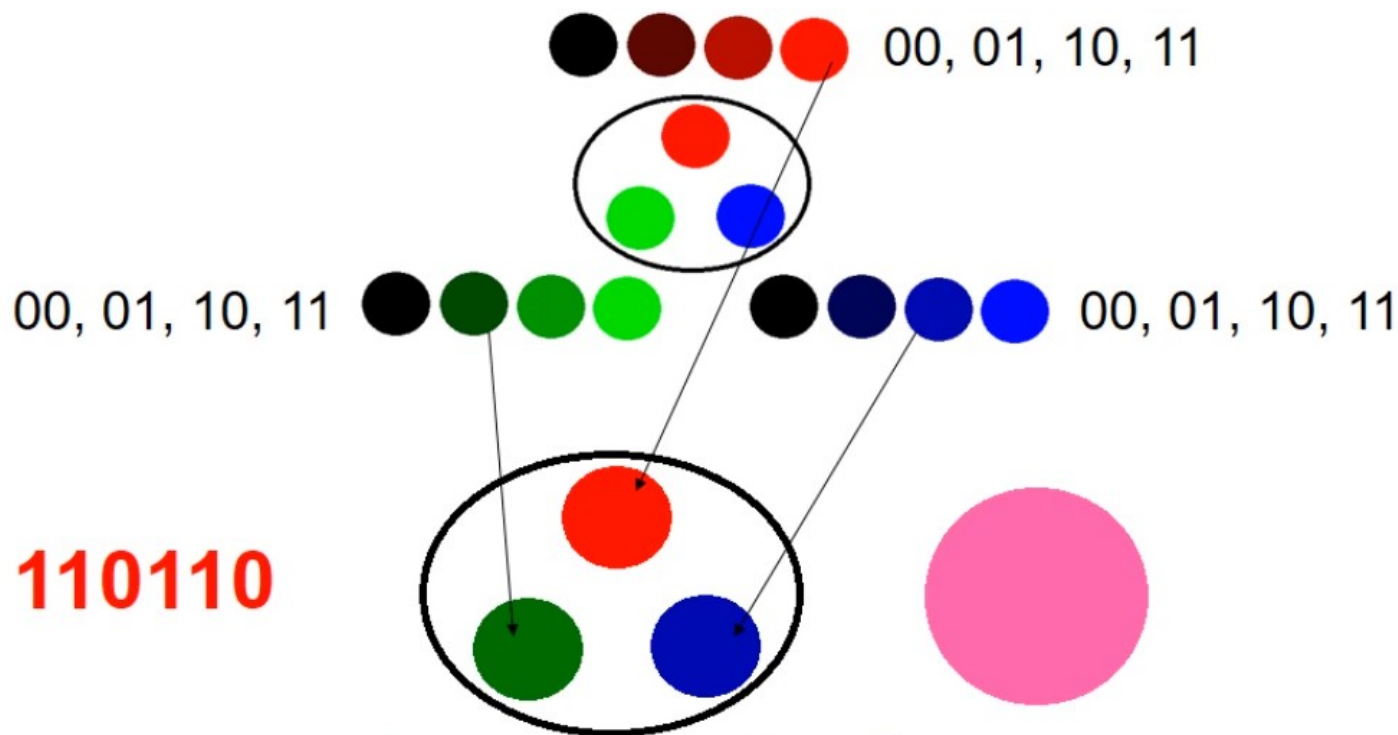
- Ví dụ: Màn hình CRT màu với mỗi chấm nhỏ trong pixel chia thành 256 cường độ (Full Color).



Thiết bị xuất



- Ví dụ: Sử dụng 6 bit để lưu trữ màu, tức là mỗi chấm nhỏ trong pixel sẽ sử dụng 2 bit để mã hóa cường độ.



Thiết bị xuất

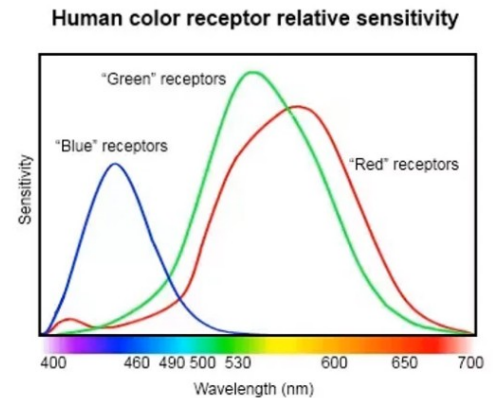


- Ví dụ: Một màn hình có độ phân giải 640 x 480 điểm ảnh, mỗi điểm ảnh chứa 6 bit màu. Ta có một số nhận xét như sau:
 - Số lượng màu có thể hiển thị tại mỗi điểm ảnh là
$$2^6 = 64 \text{ màu.}$$
 - Dung lượng bộ nhớ frame buffer cần sử dụng cho màn hình là $640 \times 480 \times 6 = 1.843.200 \text{ bits} = 230.400 \text{ bytes} = 225 \text{ KBytes}.$

Thiết bị xuất



- **Độ nhạy cảm về màu của thị giác** được xác định bằng việc nhận diện số lượng màu nhiều hay ít.
- Trong trường hợp mỗi điểm ảnh trên màn hình có thể hiển thị được 2^n màu với n không chia hết cho 3.
 - Độ nhạy cảm về màu trong mắt người được sắp xếp theo **thứ tự từ cao đến thấp là Green, Red và Blue**. Do đó ta **không cần chia đều các bit**.
 - Màu nào mắt con người nhạy cảm, ta sẽ phân bổ số bit nhiều hơn.





Thiết bị xuất



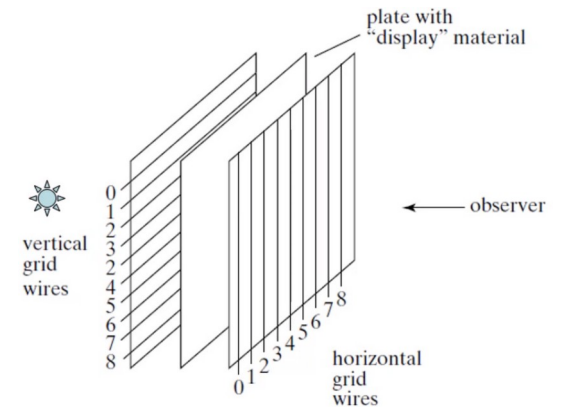
- Màn hình tinh thể lỏng đen trắng

- Cấu tạo

- Hai lớp điện cực nằm phía bên ngoài.
 - Một tấm dùng để xác định tọa độ theo chiều ngang.
 - Một tấm dùng để xác định tọa độ theo chiều dọc.
- Lớp ở giữa chứa các thành phần tinh thể lỏng.
- Có một nguồn phát sáng nằm phía sau màn hình.

- Hoạt động

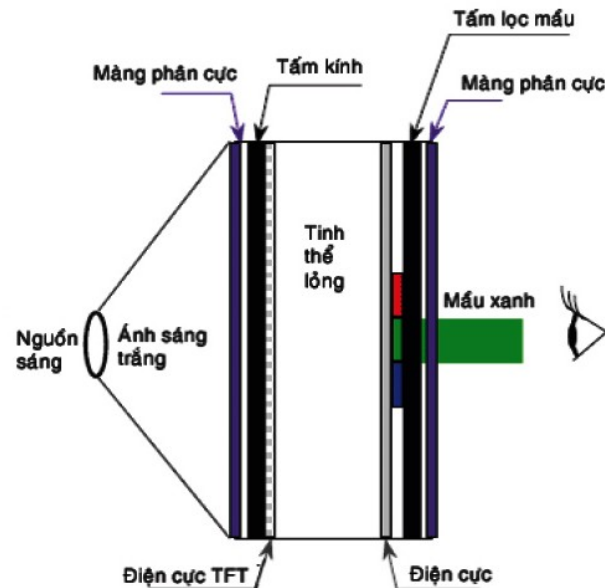
- Khi ta không tác động hiệu điện thế tại một tọa độ bất kỳ trên hai lớp điện cực bên ngoài thì các tinh thể lỏng sẽ nằm ngang và che mất ánh sáng đi từ phía sau ra phía trước màn hình (màu đen).
- Khi ta tác động tại một tọa độ bất kỳ một hiệu điện thế thì tinh thể lỏng tại vị trí đó sẽ xoay và làm cho ánh sáng đi từ sau ra trước và ta có thể thấy được điểm ảnh đó (màu trắng).



Thiết bị xuất



- Màn hình tinh thể lỏng màu
 - Cấu tạo và hoạt động giống màn hình tinh thể lỏng đen trắng đối với các điểm nhỏ bên trong pixel.
 - Nguyên lý tạo màu giống như với màn hình CRT.



Tạo lập hình ảnh



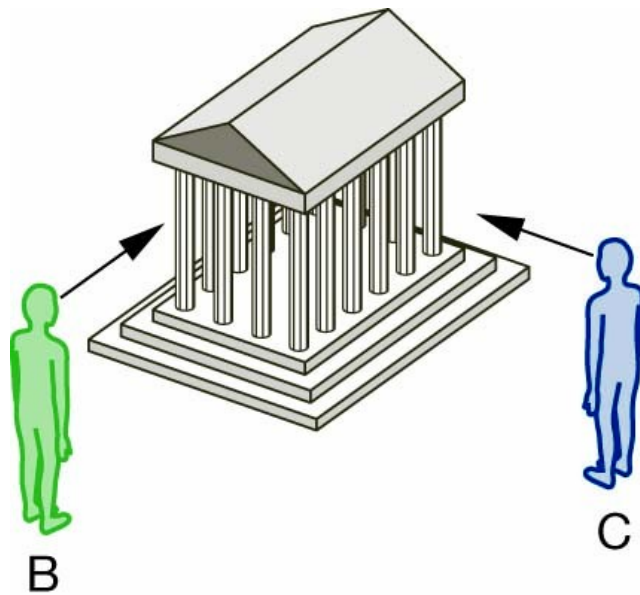
- Để tạo lập hình ảnh bằng thư viện đồ họa, ta cần xác định các thành phần sau:
 - Các đối tượng (objects)
 - Các vật liệu (materials): màu sắc, chất liệu, ...
 - Góc nhìn (viewer)
 - Nguồn sáng (light source)



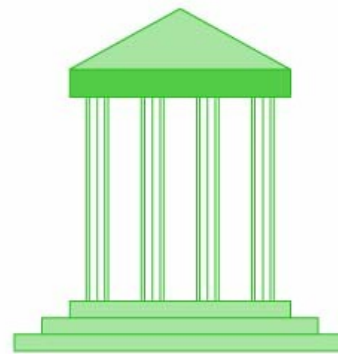
Tạo lập hình ảnh



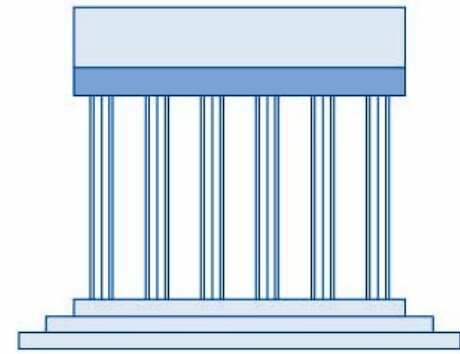
- Ví dụ đối tượng và góc nhìn



(a)



(b)

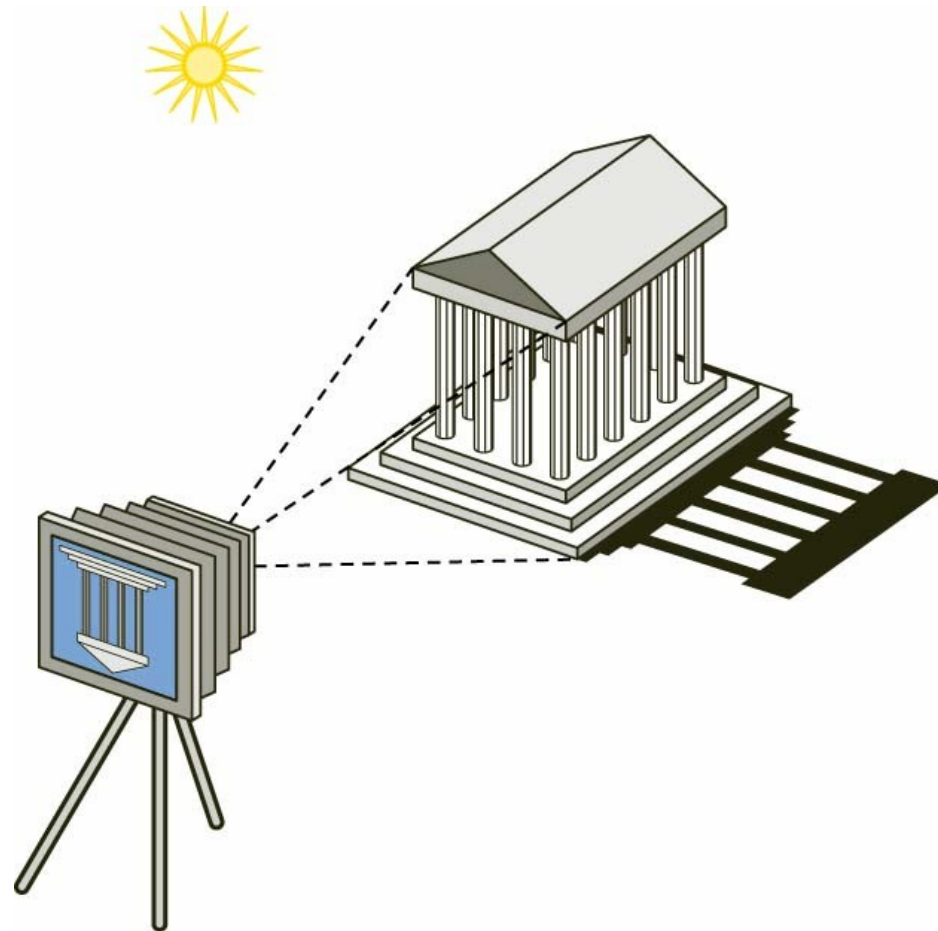


(c)

Tạo lập hình ảnh



- Ví dụ nguồn sáng



Kiến trúc đường ống đồ họa (pipeline)



- Kiến trúc pipeline chia xử lý ra thành nhiều khâu độc lập với nhau để đơn giản hóa.
- Tất cả các bước bên trong **đường ống đồ họa** có thể được **thực hiện bên trong phần cứng** (cụ thể là bên trong **card đồ họa**)
- Xử lý thông tin ở các đỉnh có thể **xử lý song song** để tăng tốc độ chương trình.

Vertex processor



- **Xử lý đỉnh** (vertex processor) là chuyển đổi mô tả đối tượng từ hệ trục tọa độ này sang hệ trục tọa độ khác.
 - Tọa độ của đối tượng
 - Tọa độ của camera
 - Tọa độ của màn hình
- Tất cả việc chuyển đổi tọa độ này tương đương với việc biến đổi ma trận.
- Việc xử lý đỉnh cũng bao gồm cả việc tính toán màu sắc của đỉnh.

Primitive Assembly



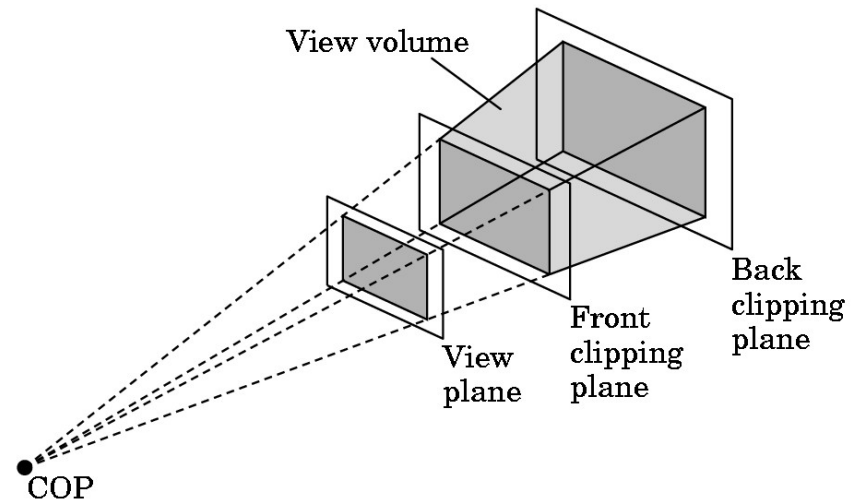
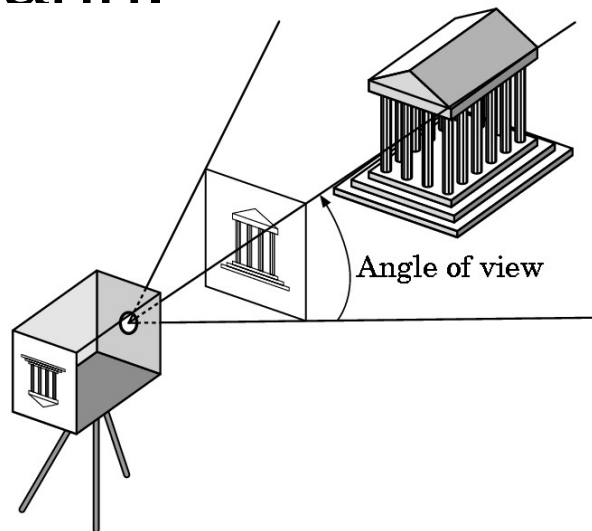
- Dựa trên các đỉnh, các đối tượng hình học sẽ được tạo ra.
 - Các đoạn thẳng
 - Các đa giác
 - Các đường cong
 - Các bề mặt



Clipping



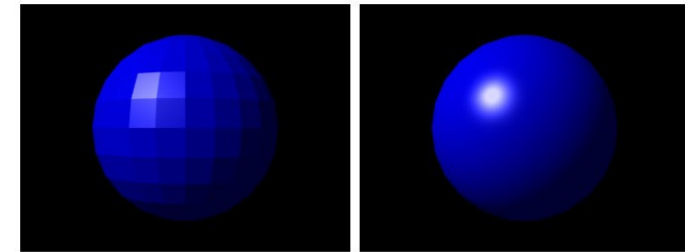
- Các đối tượng không nằm trong vùng **View Volume** sẽ bị **cắt xén** (clipped out) ra khỏi bối cảnh.



Rasterization



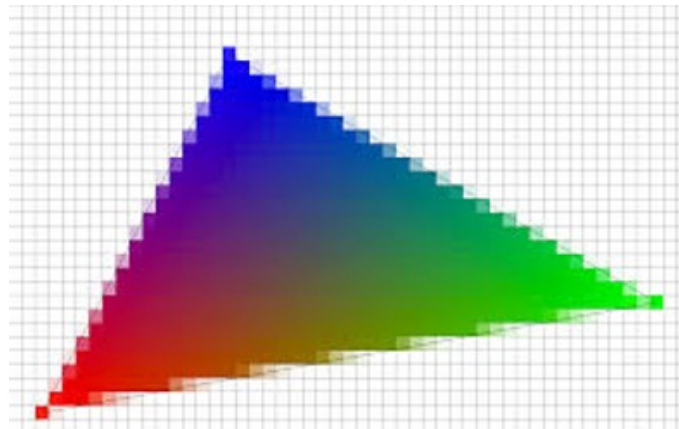
- Trong WebGL các đối tượng được tạo lập bởi nhiều mặt phẳng (**fragments**) kết hợp với nhau. Các fragments này được tạo ra bởi sự kết hợp giữa các đỉnh.
- Rasterizer sinh ra các fragments cho mỗi đối tượng
- Fragments chứa các pixels mà
 - Nằm trong frame buffer
 - Có thuộc tính màu và chiều sâu (số bit màu)



Fragment Processing



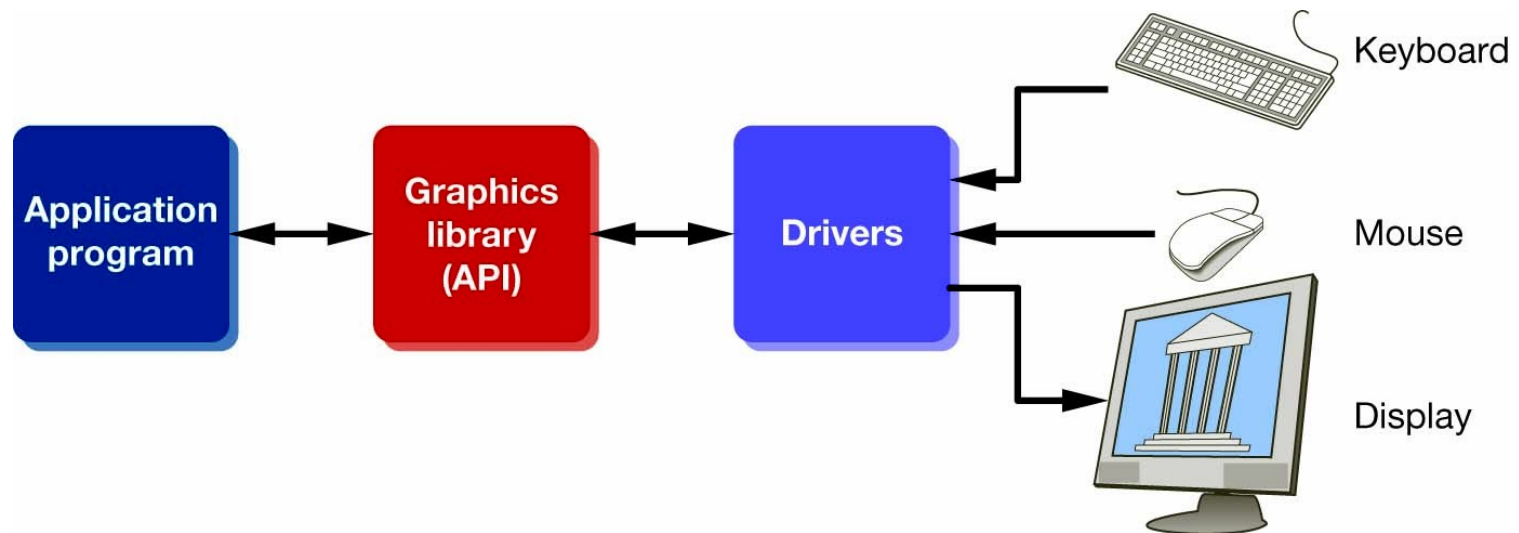
- Các fragments được xử lý để xác định màu của pixel trên màn hình tương ứng với giá trị trong frame buffer.
- Các màu được xác định bởi texture hoặc nội suy theo màu của đỉnh.



Lập trình đồ họa



- Người lập trình sẽ lập trình đồ họa thông qua một giao tiếp phần mềm **API** (Application Programmer Interface)



Lập trình đồ họa



- Thư viện đồ họa cung cấp cho người lập trình các hàm cần thiết để **tạo dựng và đặc tả các thành phần**
 - Các đối tượng (objects)
 - Mất thời gian nhất
 - Các vật liệu (materials)
 - Góc nhìn (viewer)
 - Nguồn sáng (light source)
- Ngoài ra, thư viện đồ họa cũng cung cấp các hàm hỗ trợ cho xử lý
 - Thiết bị đầu vào (chuột, bàn phím)
 - Các khả năng khác liên quan đến hệ thống

Lập trình đồ họa



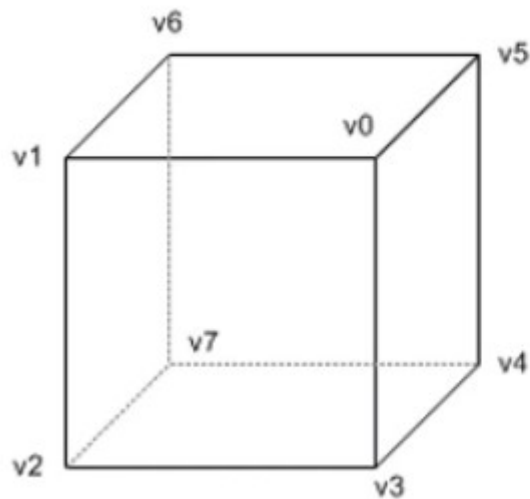
- Đặc tả đối tượng
 - Thư viện đồ họa **KHÔNG** hỗ trợ các hàm để vẽ các đối tượng phức tạp, do đó để vẽ các đối tượng phức tạp ta cần phải tách nó thành các đối tượng đơn giản.
 - Thư viện đồ họa chỉ hỗ trợ vẽ các đối tượng sau
 - Điểm
 - Đường thẳng
 - Đa giác (WebGL chỉ hỗ trợ vẽ tam giác)



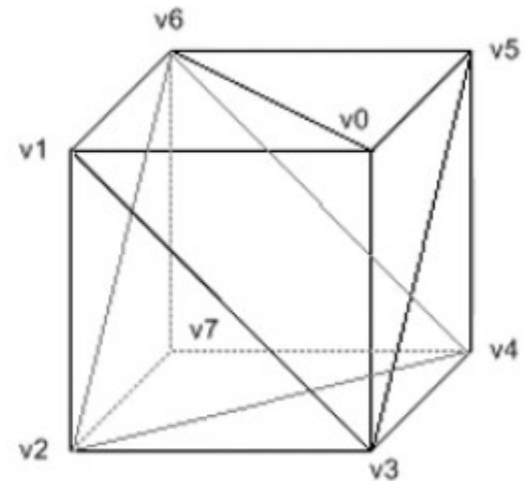
Lập trình đồ họa



- Đặc tả đối tượng
 - Cần xây dựng thuật toán thật tốt để tăng tốc độ xử lý vẽ hình.
 - Ví dụ vẽ hình lập phương



Chia thành 6 hình vuông

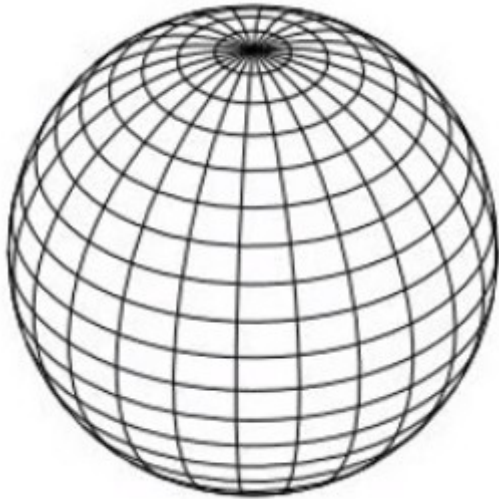


Chia thành 12 hình tam giác

Lập trình đồ họa



- Đặc tả đối tượng
 - Ví dụ vẽ hình cầu



**Chia thành các hình thang
(chia theo kinh tuyến - vĩ tuyến)**



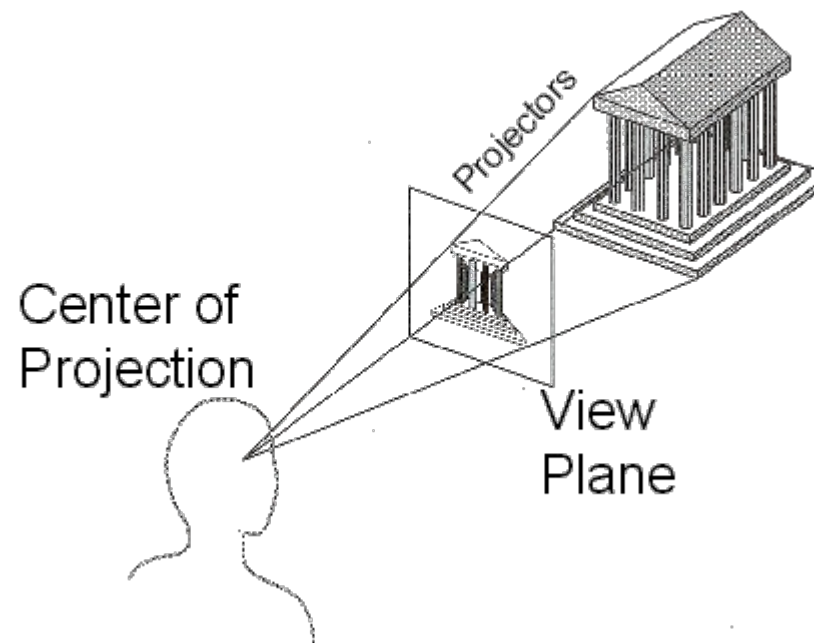
Chia thành các hình tam giác



Lập trình đồ họa



- Đặc tả góc nhìn (viewer)
 - Việc thiết lập khá đơn giản đối với thư viện đồ họa.
 - Người lập trình cần nghiên cứu về phương pháp để có thể hiểu tham số và thiết lập cho đúng.



Lập trình đồ họa



- Đặc tả nguồn sáng và vật liệu (light & material)
 - Việc thiết lập khá đơn giản đối với thư viện đồ họa.
 - Người lập trình cần nghiên cứu về phương pháp để có thể hiểu tham số và thiết lập cho đúng.
 - Các kiểu nguồn sáng
 - Nguồn sáng từ một điểm và nguồn sáng phân tán
 - Nơi chiếu sáng
 - Nguồn sáng xa và gần
 - Các thuộc tính màu
 - Các đặc tính vật liệu
 - Hấp thụ: các đặc tính màu
 - Phân tán
 - Khuyếch tán
 - Tạo bóng

Các công cụ hỗ trợ lập trình đồ họa trong khóa học




- HTML (Hypertext Markup Language)
- Ngôn ngữ lập trình JavaScript
- Thư viện đồ họa WebGL
- Ngôn ngữ lập trình GLSL (OpenGL Shader Language)
- Thư viện JavaScript định nghĩa các kiểu dữ liệu hỗ trợ cho lập trình đồ họa và hỗ trợ tính toán véc-tơ, ma trận.



Giới thiệu về HTML



- **HTML** (Hypertext Markup Language) là ngôn ngữ đánh dấu siêu văn bản có sử dụng các **tag** để định nghĩa các thành phần bên trong trang Web.

**THANG LONG UNIVERSITY**

Ngành Khoa học máy tính

Khoa học máy tính là ngành học đóng vai trò rất quan trọng trong nghiên cứu và giải quyết các vấn đề lý luận cũng như các kỹ thuật nền tảng của CNTT và truyền thông. Ngành học này nghiên cứu những kiến thức cơ bản trong tin học nhằm mục đích áp dụng các thuật toán và toán học vào các bài toán trong thực tế. Cụ thể, những vấn đề được ngành học quan tâm nghiên cứu như: Nguyên lý làm việc của máy tính, biểu diễn cấu trúc của dữ liệu trong máy tính cùng các lý thuyết cơ bản trong lĩnh vực tin học, v.v. Các vấn đề này là cơ sở cho các ngành công nghệ thông tin - truyền thông khác.

Cử nhân khoa học máy tính là những người có hiểu biết rất cơ bản về các vấn đề nêu ở trên. Với kiến thức nền tảng của ngành học này, họ có đầy đủ tri thức để có thể tham gia nhiều mảng trong lĩnh vực tin học như nghiên cứu chuyên sâu, trở thành những nhà phát triển ứng dụng phần mềm chuyên nghiệp, những nhà phát triển các ứng dụng trí tuệ nhân tạo cũng như kiến trúc sư trưởng về hệ thống công nghệ thông tin và truyền thông, v.v.

Cử nhân khoa học máy tính do vậy là những người có kiến thức chuyên sâu và khả năng phân tích giải quyết vấn đề trong tin học một cách cơ bản. Họ có tư duy sâu sắc về hoạt động của máy tính, bao gồm cả phần cứng lẫn phần mềm, họ có khả năng tạo ra các sản phẩm phần mềm đáp ứng nhu cầu của thị trường một cách chuyên nghiệp.

Mã ngành
7340201

Thời gian học
04 năm

Tổ hợp môn thi
A00, A01

Học phí
22.000.000vnd/năm

Điểm trúng tuyển các năm gần đây (thang điểm 30)

| | |
|----------|-------|
| Năm 2019 | 16.00 |
| Năm 2018 | 15.00 |

Đối tác của TLU

view-source:https://thanglong.edu.vn/khoa-toan-tin-hoc/nganh-khoa-hoc-may-tinh

```
<li><a href="/khoa-toan-tin-hoc/nganh-cong-nghe-thong-tin">Ngành Công nghệ thông tin</a></li>
</div>
</div>
<div class="col-md-8 col-xl-6">
<div class="layout_content">
<h1 class="page-big-title d-none d-md-block">Ngành Khoa học máy tính</h1>
<div class="page-single-entry">
<div class="paragraph paragraph--type--section-text paragraph--view-mode--default">
<div class="clearfix text-formatted field field--name-field-paragraph-text field--type-text-long field--labe
"><p>Khoa học máy tính là ngành học đóng vai trò rất quan trọng trong nghiên cứu và giải quyết
truyền thông. Ngành học này nghiên cứu những kiến thức cơ bản trong tin học nhằm mục đích áp dụng các thuật to
để được ngành học quan tâm nghiên cứu như: Nguyên lý làm việc của máy tính, biểu diễn cấu trúc của dữ liệu tro
.v.v. Các vấn đề này là cơ sở cho các ngành công nghệ thông tin - truyền thông khác.</p>
</div>
<div class="clearfix text-formatted field field--name-field-paragraph-text field--type-text-long field--labe
"><p>Cử nhân khoa học máy tính là những người có hiểu biết rất cơ bản về các vấn đề nêu ở trên. Với kiến thức nê
nhiều mảng trong lĩnh vực tin học như nghiên cứu chuyên sâu, trở thành những nhà phát triển ứng dụng phần mề
cũng như kiến trúc sư trưởng về hệ thống công nghệ thông tin và truyền thông, .v.v.</p>
</div>
<div class="clearfix text-formatted field field--name-field-paragraph-text field--type-text-long field--labe
"><p>Cử nhân khoa học máy tính do vậy là những người có kiến thức chuyên sâu và khả năng phân tích giải quyết à
động của máy tính, bao gồm cả phần cứng lẫn phần mềm, họ có khả năng tạo ra các sản phẩm phần mềm đáp ứng nhu
</div>
</div>
<div class="paragraph paragraph--type--section-text paragraph--view-mode--default">
<div class="clearfix text-formatted field field--name-field-paragraph-text field--type-text-long field--labe
"><h2>Mục tiêu đào tạo</h2>
</div>
</div>
<div class="paragraph paragraph--type--section-text paragraph--view-mode--default">
<div class="clearfix text-formatted field field--name-field-paragraph-text field--type-text-long field--labe
"><p>Sau khi tốt nghiệp ngành Khoa học máy tính, sinh viên sẽ đạt được những yêu cầu sau:</p>
</div>
<div class="list-group">
<ul>
<li><i>Nắm vững các nguyên lý và lý luận về tính toán bên trong máy tính; Biểu diễn, quản lý dữ liệu và tri t
dụng vào các bài toán thực tế; Có hiểu biết sâu sắc về một số lĩnh vực trong tin học để có thể đưa ra giải ph
</li>
<li><i>Nắm vững một số ngôn ngữ lập trình phổ biến và hiện đại; Có kỹ năng lập trình vững vàng để giải quyết
các công cụ mới liên quan đến lĩnh vực lập trình.</i></li>
<li><i>Nắm bắt và vận dụng được các công cụ phát triển phần mềm để xây dựng được các sản phẩm phần mềm theo m
</li>
<li><i>Có khả năng tiếp cận các vấn đề liên quan đến trí tuệ nhân tạo theo cả hướng nghiên cứu chuyên sâu và
</li>
<li><i>Có đầy đủ tri thức để trở thành một kiến trúc sư trưởng về hệ thống thông tin; Tích hợp hệ thống, chủa
doanh nghiệp hay tổ chức trên nền tảng sử dụng các yếu tố công nghệ thông tin.</i></li>
</ul>
</div>
</div>
<div class="paragraph paragraph--type--section-text paragraph--view-mode--default">
```

Giới thiệu về ngôn ngữ Javascript



- Là một **ngôn ngữ lập trình kịch bản** (không đòi hỏi biên dịch trước khi thực thi) với cách viết tựa C.
- Mã lệnh của Javascript được **thực thi bởi Web Browser**.
- Được **sử dụng kết hợp với HTML** nhằm tăng hiệu quả thiết kế và hoạt động trên giao diện Web.



Giới thiệu về ngôn ngữ Javascript



- Cách nhúng Javascript vào HTML
 - **Cách 1**: Viết mã lệnh Javascript bên trong phần nội dung của tag

<script type="text/javascript"> ... </script>

```
<!DOCTYPE html>
<html>

<head>
  <title>Show warning</title>
</head>

<body>
  <button id="show-warning">Show</button>
</body>

<script type="text/javascript">

  let show_warning = function () {
    alert("Day la canh bao !!!");
  }

  let button = document.getElementById("show-warning");
  button.onclick = show_warning;

</script>

</html>
```

Giới thiệu về ngôn ngữ Javascript



- Cách nhúng Javascript vào HTML
 - **Cách 2**: Viết mã lệnh Javascript bên trong một **file có đuôi .js**, sau đó thiết **đặt giá trị của thuộc tính src** trong tag **script** là đường dẫn đến file này.

```
<script type="text/javascript" src="../../Common/initShaders.js"></script>  
<script type="text/javascript" src="../../Common/MVnew.js"></script>  
<script type="text/javascript" src="tlu.js"></script>
```

WebGL 2.0



- WebGL là một thư viện đồ họa triển khai trên ngôn ngữ Javascript.
- Được hỗ trợ bởi các trình duyệt web mới nhất.

| IE | Edge * | Firefox | Chrome | Safari | Opera | iOS Safari * | Opera Mini * | Android Browser * | Opera Mobile * | Chrome for Android | Firefox for Android | UC Browser for Android | Samsung Internet | QQ Browser | Baidu Browser | KaiOS Browser |
|------|--------|---------|--------|---------|---------|--------------|--------------|-------------------|----------------|--------------------|---------------------|------------------------|------------------|------------|---------------|---------------|
| | | 2-3.6 | 4-7 | 3.1-5 | 10-11.5 | | | | | | | | | | | |
| | 12-18 | 4-23 | 8-32 | 5.1-7.1 | 12.1-18 | 3.2-7.1 | | | | | | | | | | |
| 6-10 | 79-84 | 24-79 | 33-84 | 8-13.1 | 19-69 | 8-13.7 | | 2.1-4.4.4 | 12-12.1 | | | | 4-11.2 | | | |
| 11 | 85 | 80 | 85 | 14 | 70 | 14.0 | all | 81 | 59 | 85 | 79 | 12.12 | 12.0 | 10.4 | 7.12 | 2.5 |
| | | 81-82 | 86-88 | TP | | | | | | | | | | | | |

- WebGL 2.0 yêu cầu phần cứng có hỗ trợ thư viện đồ họa OpenGL ES 3.0
 - OpenGL ES là phiên bản đơn giản hơn của OpenGL dành cho các hệ thống nhúng.
 - Từ phiên bản OpenGL ES 2.0 trở đi, thư viện đồ họa chạy trên **nền tảng Shader**

GLSL



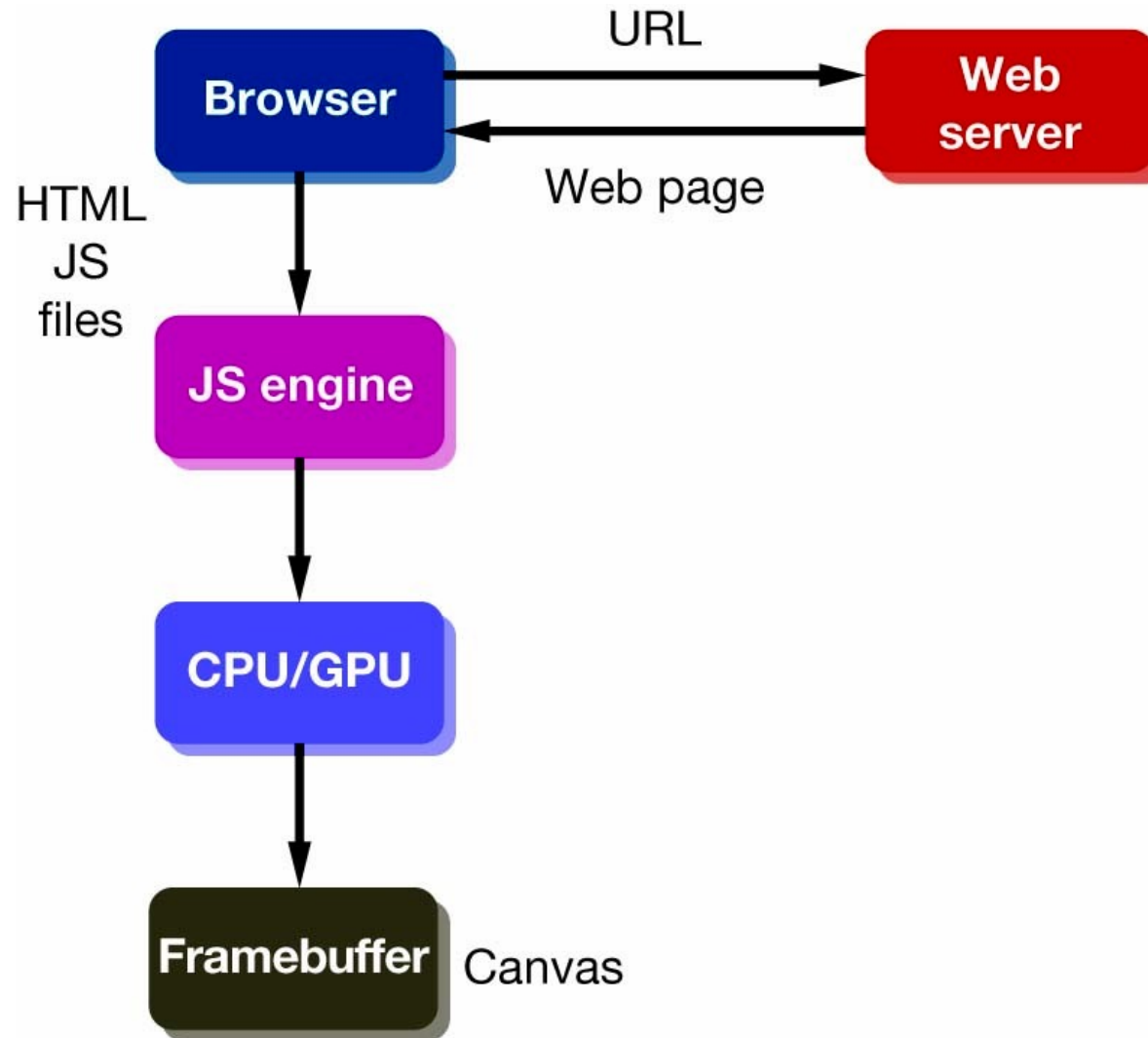
- GLSL (OpenGL Shader Language) là một ngôn ngữ tương tự ngôn ngữ C và được dùng để tạo ra các shader.

```
<script id="vertex-shader" type="x-shader/x-vertex">
  #version 300 es

  in vec2 vPosition;

  void main()
  {
    gl_Position = vec4(vPosition, 0.0, 1.0);
  }
</script>
```

Thực thi chương trình đồ họa trên WebBrowser



Thư viện Javascript hỗ trợ



- File **InitShaders.js**: Dùng để khởi tạo các shaders.
- File **MV.js**: Dùng để hỗ trợ tính toán ma trận, véc-tơ, phép chiếu,



Ví dụ 1



- Vẽ một hình tam giác màu đỏ
 - File HTML
 - Chứa mô tả của trang Web.
 - Chứa các tiện ích, thư viện của chương trình.
 - Chứa các shaders.
 - File JavaScript
 - Chứa việc khởi tạo đối tượng đồ họa trong bộ nhớ.
 - Yêu cầu vẽ hình dựa trên đối tượng đồ họa trong bộ nhớ.



Ví dụ vẽ hình tam giác

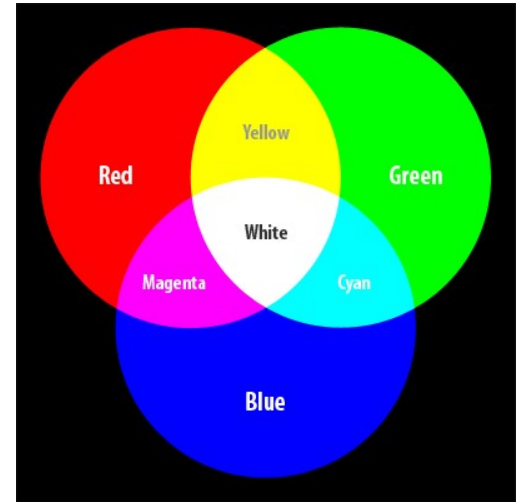


- Các lưu ý:
 - **Đối tượng `<canvas>`** trong file **`index.html`** là vùng chứa các đối tượng đồ họa.
 - **Thay đổi kích thước canvas** bằng cách thiết lập các thuộc tính *width* và *height*.
 - **Đường dẫn** tới các file `*.js` hỗ trợ lập trình cần được xác định chính xác.
 - **Đường dẫn tương đối**: Thiết lập đường dẫn tính từ thư mục chứa file `index.html`
 - Ví dụ: `.././Common/MV.js`
 - **Đường dẫn tuyệt đối**: Thiết lập đường dẫn tính từ thư mục gốc.
 - Ví dụ: `/home/libraries/Common/MV.js`

Mở rộng Ví dụ



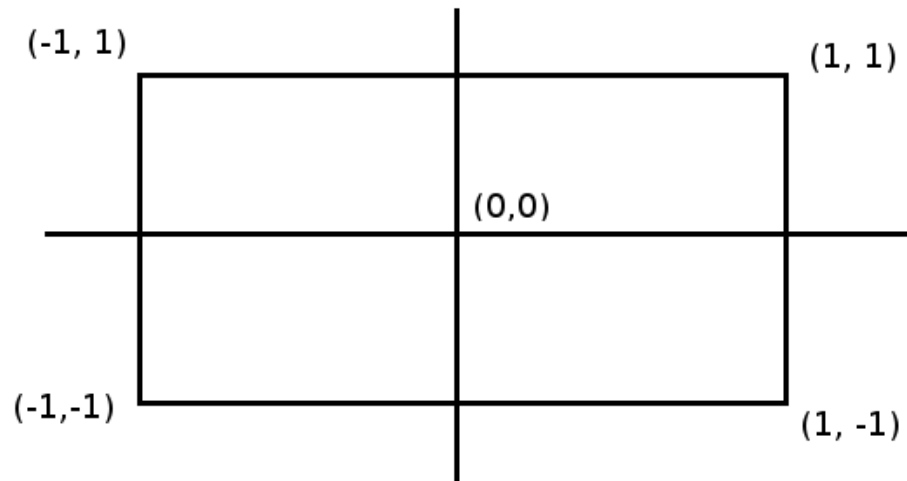
- Đổi màu cơ bản cho hình tam giác
 - Màu xanh lá cây
 - Màu xanh da trời
 - Màu vàng, magenta, lục lam
- Đổi màu tùy ý
 - Lấy màu trong các cửa sổ tạo màu



Mở rộng Ví dụ



- Vẽ 2 hình tam giác dựa trên trục tọa độ WebGL dựa trên các đỉnh tam giác cho trước.
 - Tam giác thứ nhất: $[-1, 0]$, $[-0.5, 1]$, $[0, 0]$
 - Tam giác thứ hai: $[0, 0]$, $[-0.5, -1]$, $[0.5, -1]$



Hết Tuần 1



Cảm ơn các bạn đã chú ý lắng nghe !!!