ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
----🕮----

ĐỒ ÁN MÔN HỌC

Môn học: Đồ Hoạ Máy Tính

Học kỳ II (2020 - 2021)

XÂY DỰNG TRÒ CHƠI STACK SỬ DỤNG THƯ VIỆN THREE.JS

Giảng viên: **TS. Mai Tiến Dũng**

Sinh viên thực hiện			
STT	Họ tên	MSSV	Lớp
1	Nguyễn Phúc Đạt	18520573	CS105.K21.KHTN
2	Đồng Quốc Tuấn	18520185	CS105.K21.KHTN
3	Huỳnh Minh Trí	18520176	CS105.K21.KHTN

Thành phố Hồ Chí Minh, tháng 6 năm 2021

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
----🕮----

ĐỒ ÁN MÔN HỌC

Môn học: Đồ Hoạ Máy Tính

Học kỳ II (2020 - 2021)

XÂY DỰNG TRÒ CHƠI STACK SỬ DỤNG THƯ VIỆN THREE.JS

Giảng viên: **TS. Mai Tiến Dũng**

Sinh viên thực hiện			
STT	Họ tên	MSSV	Lớp
1	Nguyễn Phúc Đạt	18520573	CS105.K21.KHTN
2	Đồng Quốc Tuấn	18520185	CS105.K21.KHTN
3	Huỳnh Minh Trí	18520176	CS105.K21.KHTN

Thành phố Hồ Chí Minh, tháng 6 năm 2021

MỤC LỤC

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

----🕮----

Đồ Án Cuối Kỳ

Tên đề tài: Xây Dựng Trò Chơi Stack Sử Dụng Thư Viện THREE.JS

……

……

……

……

……

……

……

Điểm (Thang điểm 10) ……..

GIẢNG VIÊN HƯỚNG DẪN
(Ký tên, ghi rõ họ tên)

MỞ ĐẦU

Đồ họa máy tính là một trong các lĩnh vực của khoa học máy tính nghiên cứu về các cơ sở toán học, thuật toán và kỹ thuật dùng để tạo lập, hiển thị và điều khiển hình ảnh trên màn hình máy tính. Các hình ảnh này gồm cả ảnh 2D và 3D. Đồ họa máy tính sử dụng nhiều kiến thức từ nhiều nhánh của toán học và vật lý như đại số, hình học, quang học,... Ngày nay, lĩnh vực này được ứng dụng trong nhiều ngành nghề khác nhau như: sản xuất các thiết bị nhập xuất, thiết kế vi mạch, xây dựng, chỉnh sửa ảnh/ video, trò chơi điện tử,...

Trong những năm gần đây, trò chơi ngày càng phổ biến và phát triển. Điều này phần lớn nhờ vào sự phát triển của Đồ họa máy tính, cụ thể là các thư viện và các công cụ hỗ trợ đang dần trở nên mạnh mẽ. Các nhà phát triển trò chơi có thể dễ dàng tạo lập các nhân vật một cách dễ dàng, nhanh chóng và đẹp mắt, thậm chí ngoại hình của các nhân vật còn rất giống người thật. Hành động của nhân vật hay tương tác trong trò chơi cũng trở nên chân thật và mượt mà. Điều này giúp thu hút một lượng rất lớn người chơi.

Ví dụ về đồ hoạ hình ảnh xây dựng từ phim AVATAR

Trong đồ án môn học này, dựa vào kiến thức từ môn học Đồ họa máy tính, chúng tôi đã tìm hiểu và xây dựng một trò chơi dựa trên ý tưởng của *Stack game* bằng thư viện Three.js.

PHẦN 1: TỔNG QUAN

Stack game là trò chơi giải trí thu hút nhiều người chơi hiện đang có mặt ở AppStore và Google Play. Nhiệm vụ của người chơi là phải chồng tháp cao nhất có thể. Các khối hình 3D xuất hiện từ bên trái và bên phải, người chơi phải chạm vào màn hình nếu nghĩ chúng phù hợp với nhau. Sau mỗi lần xếp không thẳng hàng thì hình khối nào thừa sẽ được cắt đi nên nó càng ngày càng thu nhỏ. Các khối phân cấp từ màu này sang màu khác, điều này tạo thêm một chút đa dạng cho trò chơi. Nếu như khối 3D chuyển động vượt khỏi màn hình hoặc không chồng lên khối tháp thì trò chơi sẽ kết thúc.

Với mục tiêu xây dựng trò chơi dựa trên ý tưởng của Stack game, chúng tôi đã tìm hiểu và xây dựng các nội dung sau bằng thư viện Three.js:

- Thiết lập môi trường game
- Định nghĩa Stack và khối hộp
- Xử lý các sự kiện của người chơi và cài đặt chuyển động
- Thực hiện một số phép biến đổi trên khối hộp.

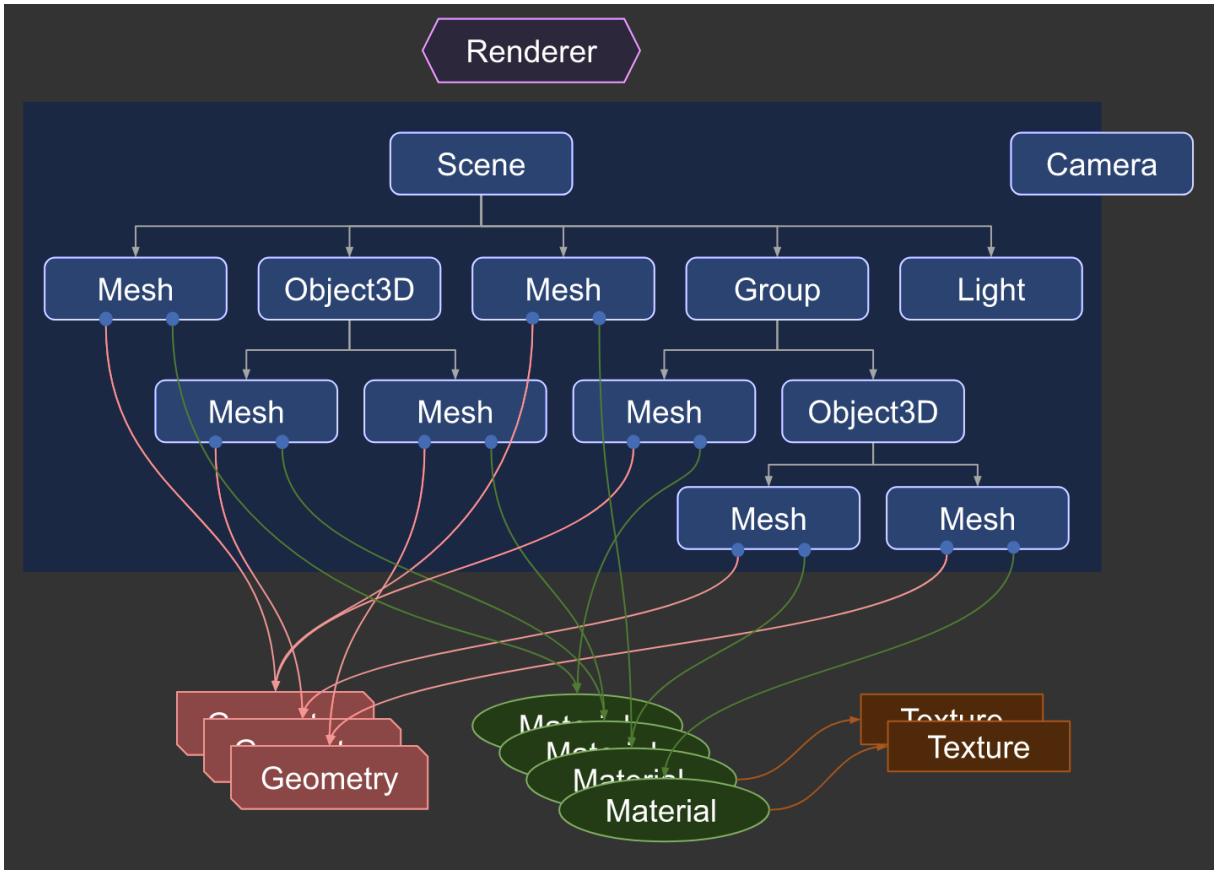
PHẦN 2: NỘI DUNG

1. Three.js

Three.js là một thư viện 3D cố gắng làm cho việc tải nội dung 3D trên trang web trở nên dễ dàng nhất có thể.

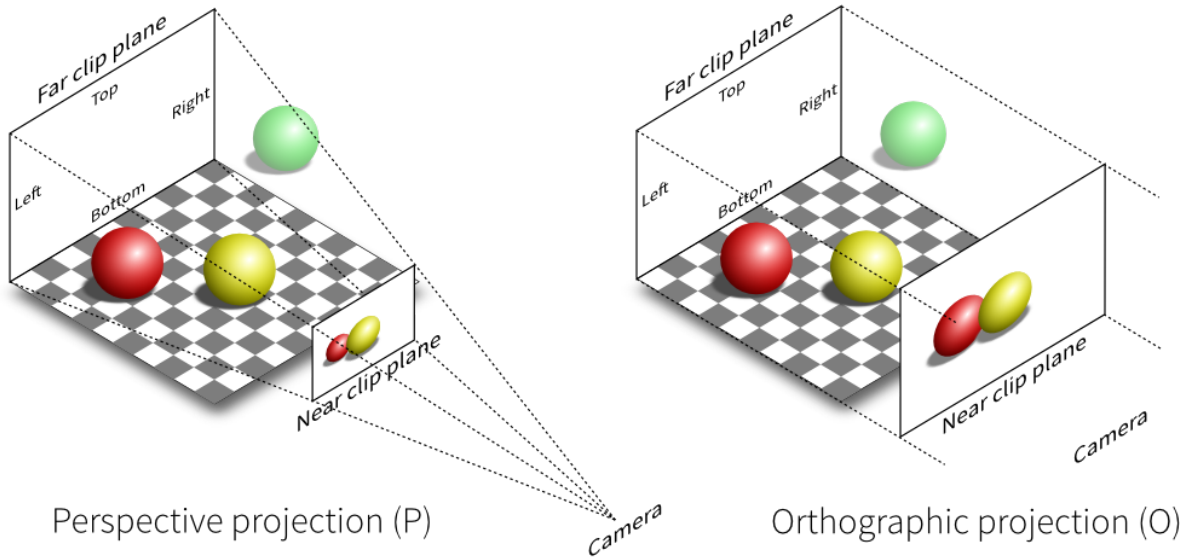
Three.js sử dụng WebGL để vẽ 3D. WebGL (Web Graphics Library) cũng là một thư viện hỗ trợ vẽ đồ họa trên máy tính tuy nhiên thư viện này chỉ hỗ trợ vẽ các thành phần cơ bản như điểm, đường thẳng và hình tam giác. Để có thể xây dựng một chương trình sử dụng WebGL sẽ yêu cầu rất nhiều đoạn mã, vì thế three.js được tạo ra để hỗ trợ việc xây dựng nội dung 3D một cách dễ dàng nhất. Cụ thể, three.js sẽ hỗ trợ một số thứ như khung cảnh (scene), ánh sáng (light), bóng (shadow), chất liệu (material), kết cấu (texture) và tất cả những khác mà WebGL không hỗ trợ.

Một ứng dụng three.js yêu cầu hàng loạt đối tượng và kết nối chúng với nhau. Hình (.) là sơ đồ biểu diễn một ứng dụng three.js.

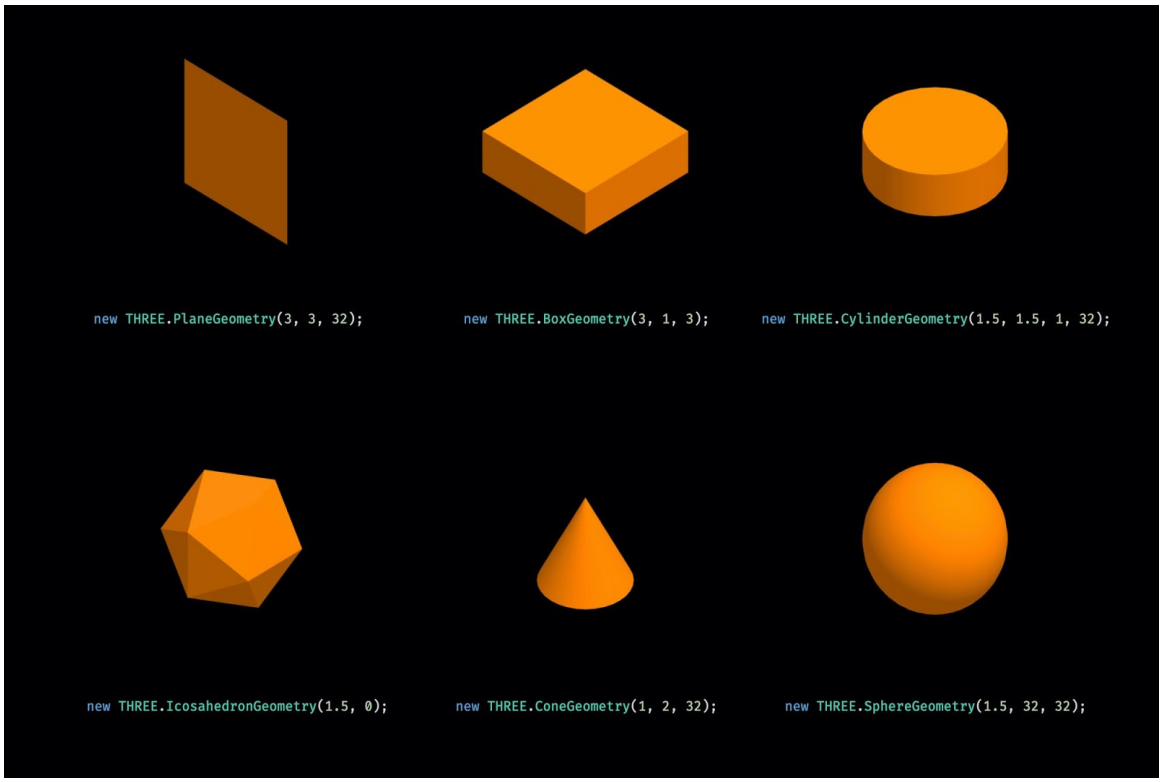


Trong đó:

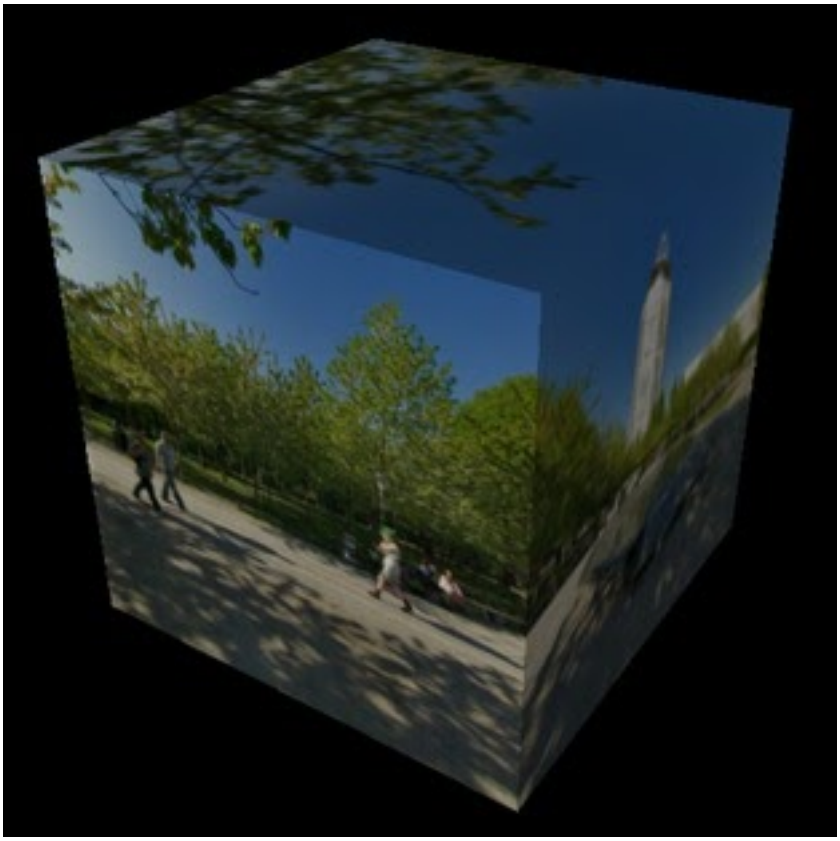
- **Renderer:** là thành phần không thể thiếu của three.js. Nó dựa vào đối tượng Scene và một phần khung cảnh 3D qua góc nhìn của Camera để render thành ảnh 2D.
- Phần còn lại được biểu diễn dưới dạng cấu trúc cây, gọi là **Scene Graph.** Nó bao quát hầu hết đối tượng trong three.js như Scene, Material, Geometry, Mesh, Light, Camera. Các đối tượng này xác định một cấu trúc cây cha/con có phân cấp và đại diện cho nơi các đối tượng xuất hiện và cách chúng được định hướng. Nút con được định vị tương đối với nút cha. Ví dụ, đối tượng các bánh xe có thể là con của đối tượng ô tô vì thế việc di chuyển và định hướng của ô tô sẽ dẫn đến các bánh xe tự động chuyển động.
- **Camera:** là đối tượng dùng để quan sát khung cảnh 3D và cung cấp thông tin cho Renderer. Có rất nhiều loại Camera có sẵn trong three.js trong đó PerspectiveCamera và OrthographicCamera được sử dụng phổ biến nhất.



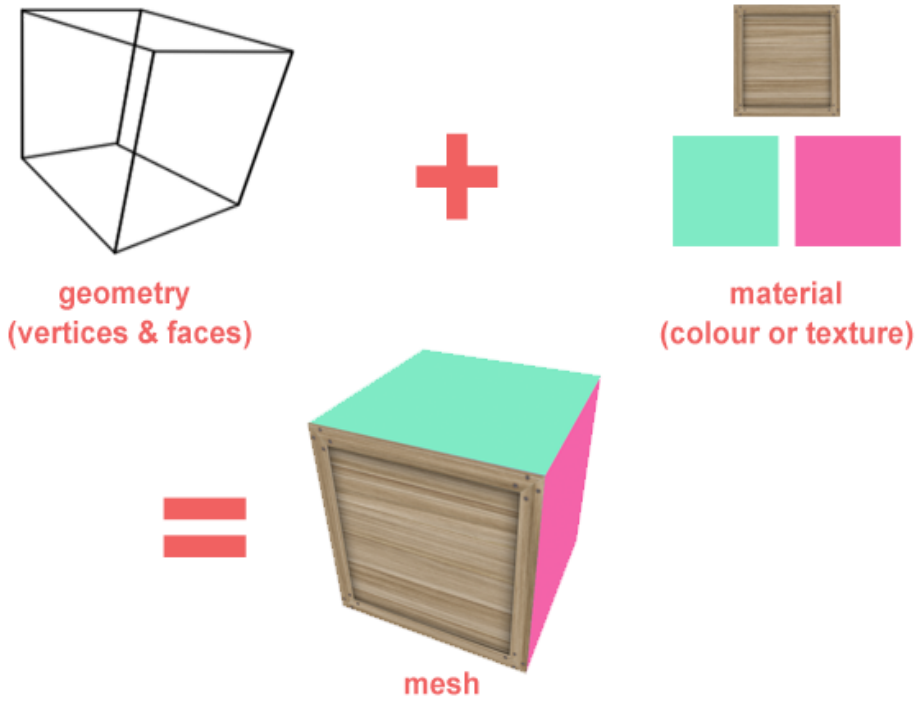
- **Geometry:** biểu diễn thông tin hình học của một đối tượng như là khối cầu, khối hộp, mặt phẳng, con mèo, con người, cây cối, tòa nhà,... Three.js cung cấp nhiều đối tượng hình học cốt lõi đáp ứng nhu cầu của người dùng.



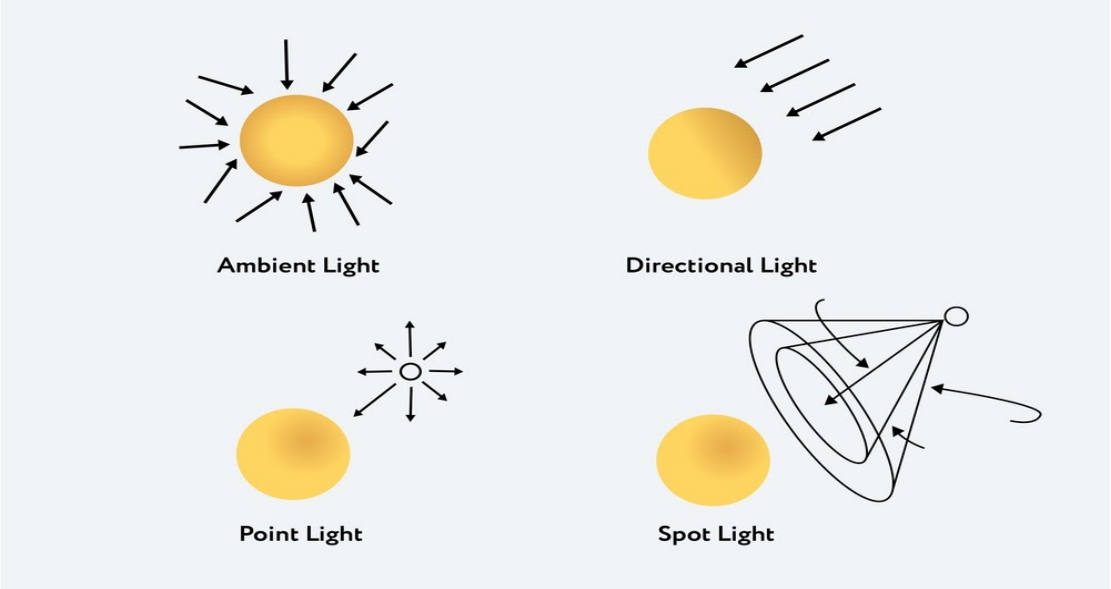
- **Texture:** đại diện cho những hình ảnh được tải lên hoặc được render từ Scene khác. Texture cũng là một cách để render đối tượng chân thực mà không phải sử dụng nhiều khối đa giác.



- **Material:** mang thông tin về thuộc tính bề mặt đối tượng như là màu sắc, độ phản chiếu. Ngoài ra, có thể áp dụng Texture cho bề mặt của đối tượng.
- **Mesh:** là một đối tượng kết hợp bởi một đối tượng Geometry và một đối tượng Material cụ thể. Cả đối tượng Geometry và đối tượng Material có thể được sử dụng bởi nhiều đối tượng Mesh.



- **Light:** là đối tượng mô phỏng lại ánh sáng có ở thực tế. Có 4 loại ánh sáng chính: Directional Light, Ambient Light, Point Light và Spot Light.



2. Thiết lập môi trường

Đầu tiên, ta cần khởi tạo các đối tượng Scene, Camera.

```
// Scene
scene = new THREE.Scene();

// Camera
const aspect = window.innerWidth / window.innerHeight;
const width = 10;
const height = width / aspect;
camera = new THREE.OrthographicCamera(
width / -2, // left
width / 2, // right
height / 2, // top
height / -2, // bottom
0, // near plane
100 // far plane
);
```

Vì mặc định môi trường có background màu đen, nên chúng tôi thêm vào 2 loại ánh sáng là Ambient Light và Directional Light để cho các đối tượng nổi bật hơn. Ambient Light sẽ tỏa sáng từ mọi phía với cường độ như nhau. Trong khi đó, Directional Light sẽ phát ra từ một nơi chỉ định.

```
// Set up lights
const ambientLight = new THREE.AmbientLight(0xffffff,
0.6);
scene.add(ambientLight);
const dirLight = new THREE.DirectionalLight(0xffffff,
0.6);
dirLight.position.set(10, 20, 0);
scene.add(dirLight);
```

Cuối cùng là khởi tạo đối tượng WebGLRenderer để render bối cảnh trò chơi vào trình duyệt.

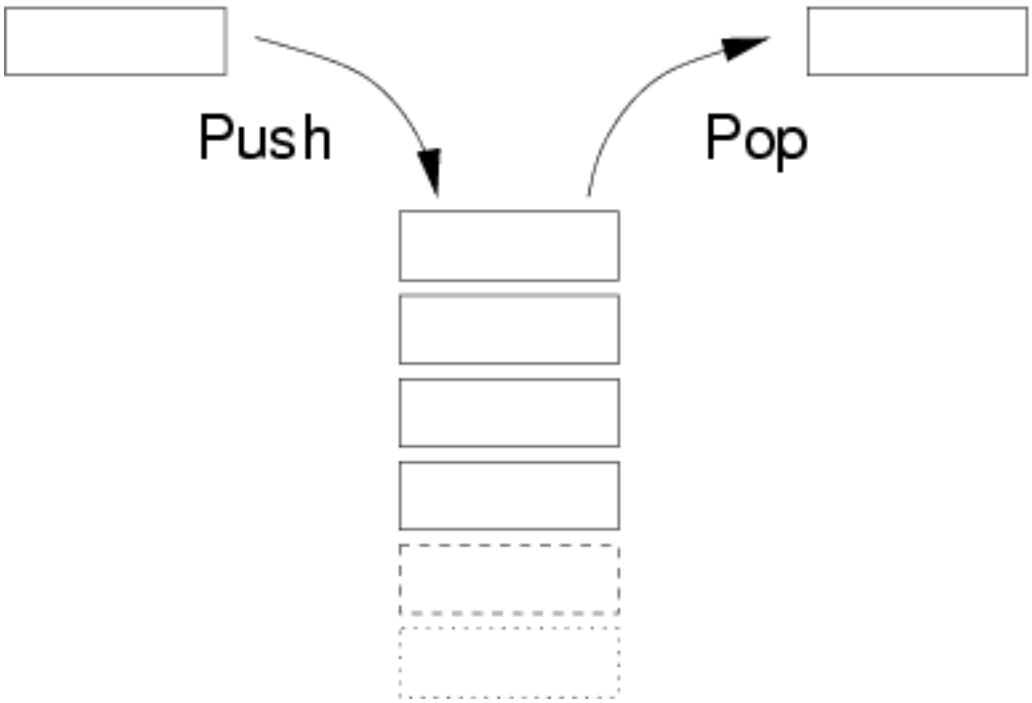
```
// Set up renderer
renderer = new THREE.WebGLRenderer({ antialias: true });
renderer.setSize(window.innerWidth, window.innerHeight);
```

```
renderer.render(scene, camera);
document.body.appendChild(renderer.domElement);
```

3. Định nghĩa Stack và khối hộp

Stack (ngăn xếp) trong khoa học máy tính là một cấu trúc dữ liệu dạng thùng chứa (container) của các phần tử (thường gọi là các lớp (layer)) hoạt động theo nguyên lý "vào sau ra trước" (Last In First Out / LIFO) và có hai phép toán cơ bản: push and pop.

Push bổ sung một phần tử vào đỉnh (top) của ngăn xếp, nghĩa là sau các phần tử đã có trong ngăn xếp. Pop giải phóng và trả về phần tử đang đứng ở đỉnh của ngăn xếp. Trong stack, các đối tượng có thể được thêm vào stack bất kỳ lúc nào nhưng chỉ có đối tượng thêm vào sau cùng mới được phép lấy ra khỏi stack.

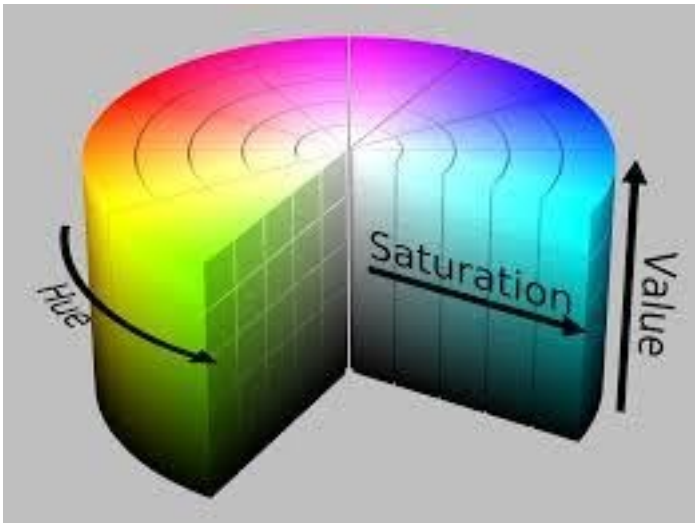


Chúng tôi sử dụng cấu trúc dữ liệu Stack là cấu trúc chính của trò chơi và các phần tử được chứa trong Stack chính là các khối hộp. Mỗi khối hộp được xem là một layer của Stack bao gồm các thuộc tính sau: chiều dài (width), chiều sâu (depth), chiều cao (height), màu sắc (color), vị trí (position), hướng dịch chuyển (direction).

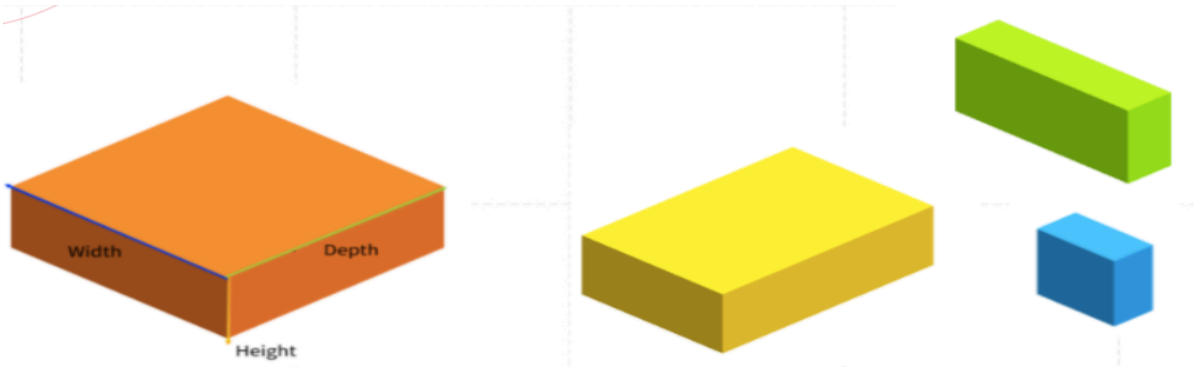
Trong quá trình khối hộp chuyển động, khi nhận một sự kiện (click hoặc nhấn phím hoặc tap) từ người chơi, kích thước của khối hộp sẽ bị thay đổi nếu như có phần thừa với layer đầu tiên của Stack. Cụ thể là chiều dài (width) và chiều sâu (depth) của khối hộp sẽ thay đổi trong suốt quá trình trò chơi diễn ra, tuy nhiên chiều cao (height) của khối hộp sẽ luôn được giữ nguyên.

Ngoài ra, để tạo sự đa dạng cho trò chơi, chúng tôi sử dụng hệ màu HSL để biểu diễn màu sắc cho từng layer. HSL viết tắt của Hue, Saturation và Lightness là hệ màu được sử dụng phổ biến ngày nay trong tất cả các hệ thống mã hóa truyền hình kỹ thuật số, bộ chọn màu, phần mềm chỉnh sửa hình ảnh. Hue là giá trị góc trên vòng tròn màu sắc nằm trong khoảng từ 0 độ đến 360 độ. Saturation là phần trăm giá trị của màu. Lightness là phần trăm độ sáng của màu. Chúng tôi sử dụng công thức sau để tính giá trị màu, mỗi layer trên sẽ tăng thêm 4 độ của giá trị Hue:

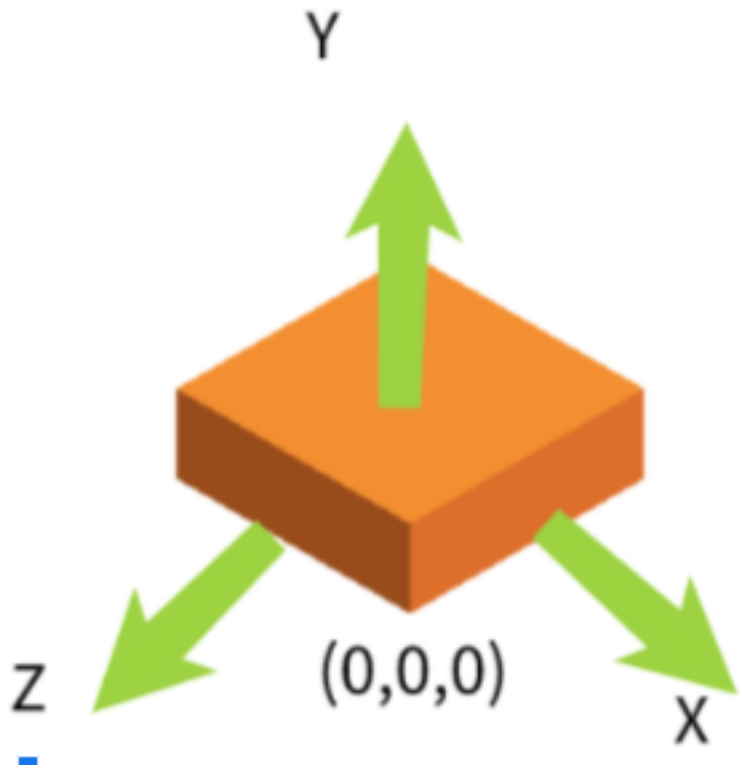
Color = HSL(30 + stack.length * 4, 100%, 50%)



Three.js đã cung cấp lớp BoxGeometry để hỗ trợ việc biểu diễn khối hộp và lớp MeshPhongMaterial biểu diễn màu sắc, mô hình phản chiếu ánh sáng. Kết hợp hai loại thực thể này, chúng tôi đã mô phỏng được khối hộp hoàn chỉnh.

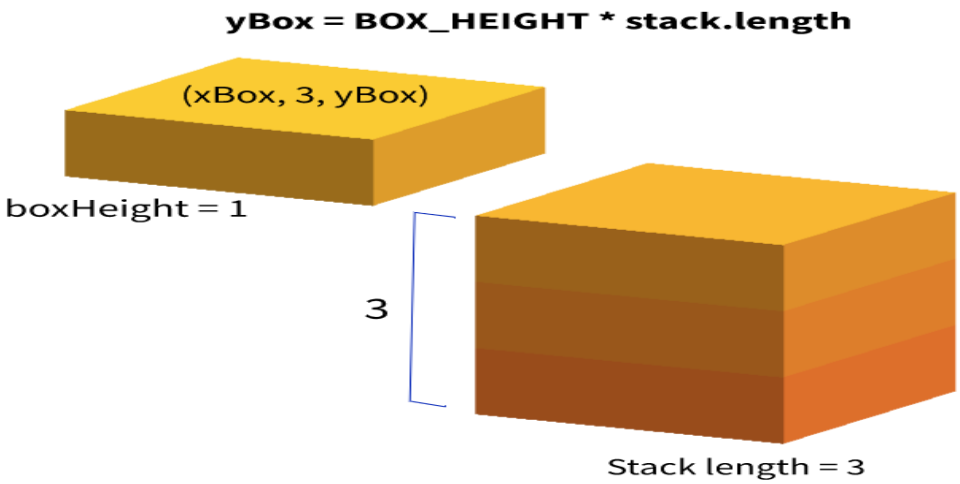


Tuy nhiên, để biểu diễn khối hộp vào không gian 3D, chúng ta cần phải biết vị trí tọa độ x, y, z của chúng. Đầu tiên, chúng tôi định nghĩa Stack nằm ở hệ trục tọa độ (0, 0, 0) như hình bên dưới.



Có thể thấy khi Stack chứa càng nhiều phần tử, thì tọa độ y của các layer sau sẽ tăng lên. Vì độ cao (height) của khối hộp luôn không đổi, nên chúng tôi định nghĩa tọa độ y của layer bằng công thức

yBox = BOX_HEIGHT * stack.length



Khi một khối hộp chuyển động sẽ có 2 hướng chuyển động là trái và phải tương ứng với chuyển động trên hệ trục tọa độ x hoặc z. Một khối hộp đã chuyển động từ hướng x thì khối hộp tiếp theo sẽ chuyển động theo hướng y và ngược lại. Chính vì thế chúng tôi đã định nghĩa thuộc tính hướng chuyển động (direction) của khối hộp và tọa độ x hoặc y sao cho khuất khỏi góc nhìn của Camera. Giả sử hướng chuyển động của khối hộp là x thì tọa độ x tương ứng của nó là -5. Điều này tương ứng nếu khối hộp chuyển động theo trục tọa độ y, lý do chúng tôi chọn tọa độ có giá trị âm vì mong muốn khối hộp sẽ trượt theo chiều dương của hệ trục tọa độ đó.

Chúng tôi đã định nghĩa phép push cho Stack khi có được tất cả thông tin về thuộc tính của khối hộp.

```
let stack=[];
const boxHeight = 1; //Height of each layer
function addLayer(x, z, width, depth, direction) {
  const y = boxHeight * stack.length; // Add the new box
one layer higher
  // ThreeJS
  const geometry = new THREE.BoxGeometry(width, boxHeight,
depth);
  const color = new THREE.Color(`hsl(${30 + stack.length *
4}, 100%, 50%)`);
  const material = new THREE.MeshPhongMaterial({ color });
  const mesh = new THREE.Mesh(geometry, material);
  mesh.position.set(x, y, z);
  scene.add(mesh);
  const layer = {
    threejs: mesh,
```

```
    width,
    depth,
    direction
  };
  stack.push(layer);
}
```

4. Xử lý sự kiện và cài đặt chuyển động

Để có thể tương tác với trò chơi, chúng tôi đã dùng hàm `window.addEventListener` của Javascript để xử lý sự kiện của người dùng. Có 3 sự kiện chính cần được xử lý chính là ‘click’, ‘touch start’ và ‘keydown’.

Khi người dùng click, tap hoặc nhấn phím Space, nếu như trò chơi chưa bắt đầu thì sẽ bắt đầu trò chơi, nếu không thì sẽ thực hiện các phép tính toán để xem xét thêm 1 layer vào Stack.

Nếu như khối hộp chuyển động vượt khỏi màn hình hoặc không chồng lên khối tháp thì trò chơi sẽ kết thúc. Khi trò chơi kết thúc sẽ hiển thị thông báo nhấn phím R để chơi lại.

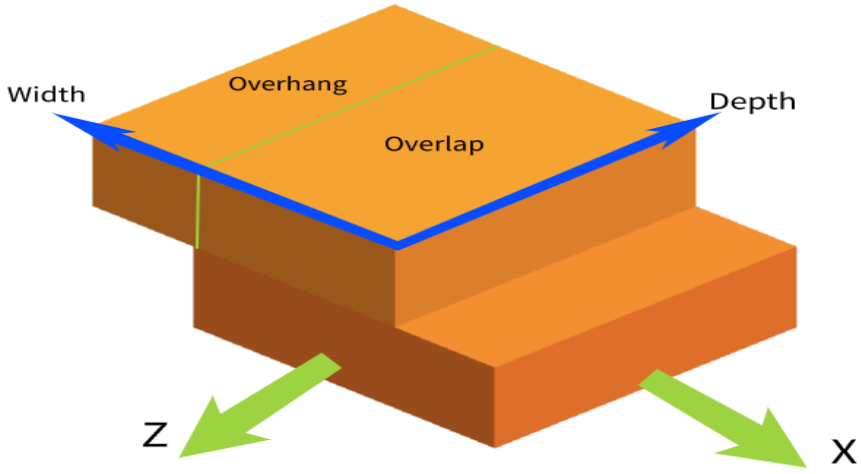
Để thực hiện được chuyển động của các khối hộp trong trò chơi, chúng tôi dùng hàm `requestAnimationFrame` của JavaScript để cho renderer cập nhật sau mỗi frame đồng thời thay đổi vị trí tọa độ của các khối hộp. Chúng tôi cập nhật vị trí tọa độ của khối hộp bằng công thức sau:

```
box.position[box.direction] += speed
```

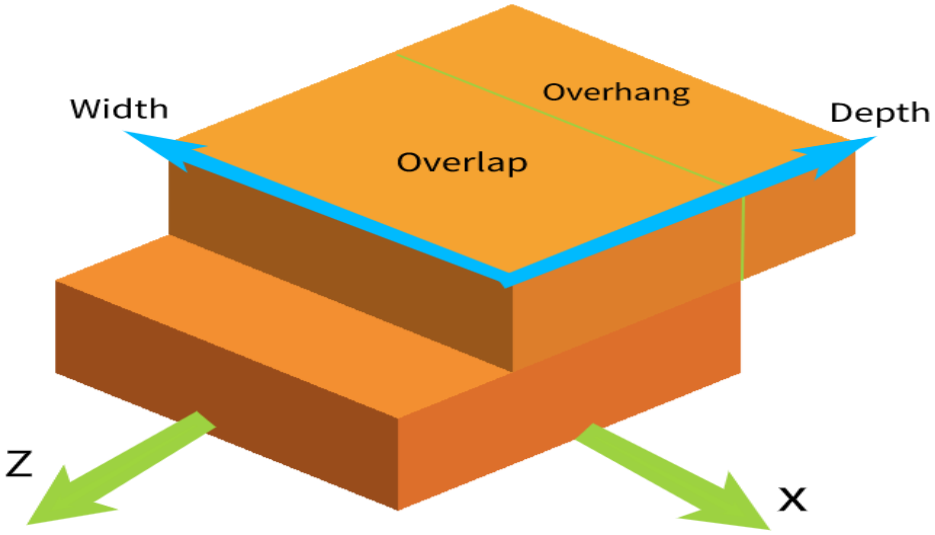
Khi một khối hộp đang di chuyển, nếu có hành động click, tap hoặc nhấn Space thì ta sẽ tính toán xem liệu khối hộp đó có khớp với layer đầu tiên của Stack hay không. Nếu không, ta sẽ loại bỏ khối hộp đó và trò chơi kết thúc. Ngược lại, khối hộp sẽ dừng chuyển động, thực hiện một số phép biến đổi để cắt bỏ phần dư thừa ra khối hộp và push thêm một khối hộp mới. Phần sau, chúng tôi sẽ trình bày một số phép tính toán trên khối hộp.

5. Xử lý một số phép biến đổi trên khối hộp

Khi khối hộp xếp chồng lên layer đầu tiên của Stack, chúng sẽ chia thành 2 phần: overlap (chồng lấp) và overhang (phần thừa).



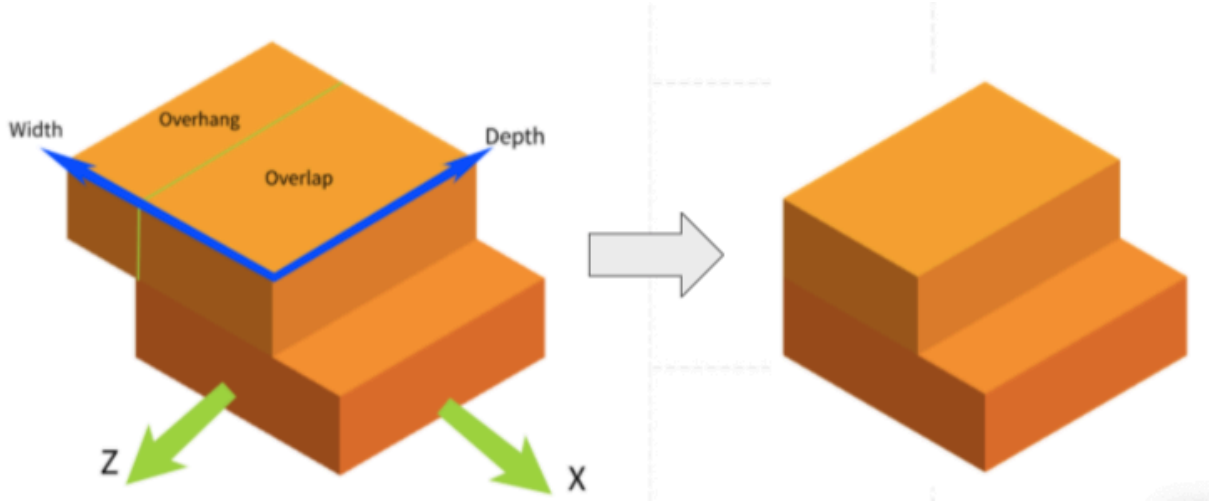
Ban đầu, chúng tôi định nghĩa chiều dài (width) sẽ tương ứng với hệ trục tọa độ x còn chiều sâu (depth) tương ứng với hệ trục tọa độ y. Vì thế nếu khối hộp dịch chuyển theo trục tọa độ x, khi cắt bỏ phần thừa (overhang) sẽ làm giảm chiều dài (width) của khối hộp. Tương tự với khối hộp dịch chuyển theo trục tọa độ y sẽ có khả năng làm giảm chiều sâu (depth).



Gọi delta là sự chênh lệch tọa độ của layer trên và layer dưới của stack. Lúc này, giá trị của phần thừa (overhang) chính là trị tuyệt đối của delta. Từ đó suy ra phần chồng lấp (overlap) bằng hiệu kích thước ban đầu của khối hộp và phần thừa (overhang). Phần chồng lấp (overlap) cũng chính là kích thước của khối hộp sau

khi cắt. Tuy nhiên, chúng ta cần thay đổi vị trí của khối hộp sau khi bị cắt để khớp với Stack. Vị trí được thay đổi theo công thức:

```
topLayer.position[topLayer.direction] -= delta / 2
```



Sau khi khối hộp đã được biến đổi, chúng tôi sử dụng hàm `addLayer` để thêm phần tử mới vào Stack với hướng chuyển động ngược lại với khối vừa được biến đổi và tiếp tục thực hiện chuyển động cho khối hộp mới.

Mỗi lần thêm vào Stack một phần tử, số điểm sẽ tăng lên 1, cho đến khi khối hộp chuyển động không trùng khớp với Stack hoặc dịch chuyển vượt ra khỏi màn hình trò chơi sẽ kết thúc.

KẾT LUẬN

Ngày nay, mô hình hóa 3D là lĩnh vực tiềm năng được nhiều nhà nghiên cứu quan tâm. Việc mô hình hóa vật thể đã hỗ trợ con người trong rất nhiều lĩnh vực như bất động sản, nội thất, y học, vũ trụ, giải trí,... Nhiều thư viện, engine được ra đời để phục vụ việc xây dựng và biểu diễn khối hình học 3D một cách dễ dàng.

Trong đồ án môn học này, chúng tôi đã đưa ra các khái niệm cơ bản của Three.js, đồng thời nêu ra các phép tính cơ bản cốt lõi của trò chơi Stack game. Để trò chơi có thể tiếp cận với nhiều người dùng, chúng tôi cần phải nâng cấp thêm một số chuyển động đẹp và thêm một số logic mới khiến trò chơi hấp dẫn hơn. Cuối cùng, việc học tập và nghiên cứu đồ họa máy tính là điều không thể thiếu trong công cuộc cách mạng 4.0 hiện nay.

TÀI LIỆU THAM KHẢO

1. Three.js - Javascript 3D Library (threejs.org)
2. WebGL: 2D and 3D graphics for the web
3. Stack game
4. Stack data structure
5. Ambient Light - Three.js
6. Directional Light - Three.js
7. Scene - Three.js
8. Camera - Three.js
9. Mesh - Three.js
10. Material - Three.js
11. Texture - Three.js
12. WebGLRenderer - Three.js

BẢNG PHÂN CÔNG CÔNG VIỆC

PHÂN CÔNG		
Thành viên	Công việc	Đánh giá
Đồng Quốc Tuấn	• Tìm hiểu cách sử dụng thư viện Three.js • Cài đặt trò chơi và deploy lên Heroku • Hỗ trợ, đóng góp ý kiến slide và report.	Hoàn thành tốt
Huỳnh Minh Trí	• Tìm hiểu cách sử dụng thư viện Three.js • Tìm hiểu và lên ý tưởng đồ án • Thiết kế slide và viết report	Hoàn thành tốt
Nguyễn Phúc Đạt	• Tìm hiểu cách sử dụng thư viện Three.js • Tìm hiểu và lên ý tưởng đồ án • Thiết kế slide và viết report	Hoàn thành tốt