ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

PHÁT TRIỂN ỨNG DỤNG ĐỒ HOẠ HỖ TRỢ HỌC TẬP

ĐỒ ÁN II

Chuyên ngành: TOÁN TIN

Giảng viên hướng dẫn:	**TS. Vương Mai Phương**
Sinh viên thực hiện:	**Phạm Thu Trang**
Mã số sinh viên:	**20195931**
Lớp:	**Toán tin 01 - K64**

HÀ NỘI – 2023

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đồ án

(a)

(b)

(c)

2. Kết quả đạt được

(a)

(b)

(c)

3. Ý thức làm việc của sinh viên:

(a)

(b)

(c)

Hà Nội, ngày 27 tháng 7 năm 2023

Giảng viên hướng dẫn

Lời cảm ơn

Trong kỳ học vừa qua, tuy được học tập và làm việc với giảng viên hướng dẫn trong một thời gian ngắn nhưng bản thân em tự nhận thấy đã học hỏi được rất nhiều kiến thức và kỹ năng cần thiết và bổ ích. Báo cáo là tổng hợp những nội dung cơ bản nhất của chủ đề nghiên cứu dưới sự hướng dẫn tận tình của TS. Vương Mai Phương cùng những kết quả có được từ quá trình làm việc nghiêm túc. Tuy nhiên, do hạn chế về thời gian và kiến thức bản thân nên không thể tránh khỏi những thiếu sót. Em rất mong nhận được những đánh giá và góp ý từ thầy cô và các bạn. Em xin chân thành cảm ơn.

Hà Nội, ngày 27 tháng 7 năm 2023

Tác giả đồ án

Phạm Thu Trang

Mục lục

Bảng ký hiệu và chữ viết tắt

WebGL	Web Graphics Library
OpenGL	Open Graphics Library
OpenGL ES	Open Graphics Library for Embedded System
HTML	HyperText Markup Language
GLSL	OpenGL Shading Language

Danh sách bảng

Danh sách hình vẽ

Mở đầu

Tên đề tài: **"Phát triển ứng dụng đồ họa hỗ trợ học tập"**.

Ứng dụng được phát triển dựa trên :

- Ngôn ngữ lập trình JavaScript, HTML5, CSS.
- Thư viện lập trình 3D dành cho WebGL là Three.js.

Báo cáo gồm bốn chương:

1. **Chương 1: Giới thiệu đề tài và giải pháp.** Chương này đưa ra vấn đề sau đó trình bày định hướng giải pháp và các công việc cần thực hiện cho đề tài. Ngoài ra, chương cũng giới thiệu cơ bản về thư viện đồ họa web - WebGL.
2. **Chương 2: Thư viện Three.js.** Chương sẽ giới thiệu tổng quan về thư viện Three.js, là thư viện 3D JavaScript dành cho WebGL
3. **Chương 3: Mô hình 3D.** Chương sẽ trình bày về các cách tạo ra mô hình phức tạp và giới thiệu về định dạng tệp OBJ/MTL cho mô hình 3D.
4. **Chương 4: Xây dựng chương trình.** Chương cuối sẽ mô tả về ứng dụng được tạo ra, trình bày các thành phần xây dựng chương trình và đánh giá kết quả thu được.

Chương 1

TỔNG QUAN VỀ ĐỊNH HƯỚNG ĐỀ TÀI

1.1 Giới thiệu đề tài

1.1.1 Đặt vấn đề

Cùng với sự phát triển không ngừng của công nghệ, việc sử dụng các công nghệ mới nhất để nâng cao chất lượng giáo dục đang trở thành một xu hướng phổ biến. Trong đó, công nghệ đồ họa 3D được sử dụng rộng rãi để hỗ trợ việc học tập. Nó không chỉ truyền đạt kiến thức một cách trực quan và sinh động mà còn có thể cho phép tương tác với các nội dung học tập một cách dễ dàng và hiệu quả.

Để tận dụng tối đa tiềm năng của công nghệ đồ họa 3D, đề tài sẽ tập trung vào việc thiết kế và triển khai một ứng dụng đồ họa 3D hỗ trợ học tập, sử dụng WebGL để đem lại trải nghiệm học tập tốt hơn cho học sinh và sinh viên.

1.1.2 Mục tiêu đề tài

Ứng dụng này sẽ được thiết kế với các tính năng tương tác, điều hướng và hiển thị dữ liệu một cách trực quan, giúp cho học sinh và sinh viên dễ dàng tương tác với các nội dung học tập một cách hiệu quả. Đồng thời, xây dựng cơ

sở dữ liệu và nội dung cho ứng dụng này, đảm bảo tính chính xác và hiệu quả trong việc truyền tải kiến thức.

Với đề tài này, mong muốn xây dựng ứng dụng có thể đóng góp vào việc thúc đẩy sự phát triển của công nghệ đồ họa 3D trong giáo dục, từ đó nâng cao chất lượng giáo dục và giúp cho học sinh và sinh viên có trải nghiệm học tập tốt hơn trong thời đại công nghệ số.

1.1.3 Nội dung công việc

Để tạo nội dung truyền thông tương tác cho web truyền thống, yêu cầu các ứng dụng hoặc tích hợp của bên thứ ba được cài đặt trong trình duyệt. Với sự gia tăng của các tiêu chuẩn web mở và nhu cầu hỗ trợ các khả năng đồ họa gốc trong trình duyệt web, thư viện đồ họa web đã ra đời để mang lại khả năng tương thích giữa các trình duyệt và được tích hợp hoàn toàn với các tiêu chuẩn web. Thư viện đồ họa web hay Web Graphics Library, thường được biết đến với tên viết tắt là WebGL, là một thư viện cho đồ họa dành cho web, được thiết kế để kết xuất ra các đồ họa 2D và 3D có thể tương tác trong trình duyệt web.

Dưới đây là các nội dung các công việc để thực hiện đề tài:

- Tìm hiểu về thư viện đồ họa web WebGL.
- Tìm hiểu về thư viện Three.js hỗ trợ lập trình 3D bằng JavaScript và cách tạo ra các mô hình 3D.
- Xây dựng được chương trình ứng dụng đồ họa với các tính năng cho phép tương tác, điều hướng và hiển thị các mô hình 3D.

1.2 Thư viện đồ họa web - WebGL

WebGL được phát triển và duy trì bởi Khronos Group, nơi quản lý một số tiêu chuẩn mở quan trọng thúc đẩy sự đổi mới về máy tính, đồ họa và phương

tiện truyền thông ngày nay như OpenGL, COLLADA,...Trên trang web chính thức của Khronos [2], WebGL được mô tả như sau:

"WebGL là tiêu chuẩn web mở, đa nền tảng, miễn phí bản quyền dành cho API[1] đồ họa 3D cấp thấp dựa trên OpenGL ES, tiếp xúc với ECMAScript[2] thông qua phần tử HTML5 Canvas. Các nhà phát triển quen thuộc với OpenGL ES 2.0 sẽ nhận ra WebGL là một API dựa trên đổ bóng sử dụng GLSL, với các cấu trúc tương tự về mặt ngữ nghĩa với các cấu trúc của API OpenGL ES cơ bản. Nó rất gần với đặc điểm kỹ thuật của OpenGL ES, với một số nhượng bộ được thực hiện đối với những gì các nhà phát triển mong đợi từ các ngôn ngữ được quản lý bộ nhớ như JavaScript. WebGL 1.0 hiển thị bộ tính năng OpenGL ES 2.0; WebGL 2.0 hiển thị API OpenGL ES 3.0"

WebGL là một API, được truy cập độc quyền thông qua một bộ giao diện lập trình JavaScript, không có các thẻ đi kèm giống như với HTML. Kết xuất 3D trong WebGL tương tự như vẽ 2D bằng phần tử canvas, ở chỗ tất cả được thực hiện thông qualệnh gọi API JavaScript. Trên thực tế, quyền truy cập vào WebGL được cung cấp bằng cách sử dụng phần tử canvas hiện có và có được ngữ cảnh bản vẽ đặc biệt dành riêng cho WebGL.

WebGL dựa trên OpenGL ES 2.0, nó là một bản chuyển thể của OpenGL tiêu chuẩn kết xuất 3D lâu đời. “ES” là viết tắt của “hệ thống nhúng”, nghĩa là nó đã được điều chỉnh để sử dụng trong các thiết bị điện toán nhỏ, đặc biệt là điện thoại và máy tính bảng. OpenGL ES là API hỗ trợ đồ họa 3D cho iPhone, iPad, điện thoại và máy tính bảng Android.

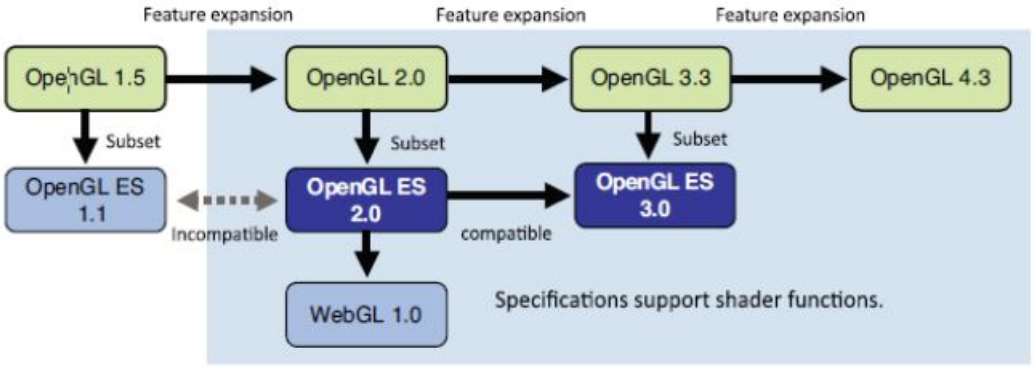
WebGL sử dụng khả năng tăng tốc phần cứng cho phép các nhà phát triển có khả năng xây dựng đồ họa tương tác hiển thị thời gian thực hiệu suất cao trong trình duyệt web, chẳng hạn như trò chơi, mô phỏng, trực quan hóa dữ liệu, video hoạt hình, mô hình 3D, hoạt ảnh phân tử,...

Hiệu quả của WebGL có lẽ là một trong những ưu điểm chính vì nó cho phép

[1]API là cơ chế cho phép 2 thành phần phần mềm giao tiếp với nhau bằng một tập hợp các định nghĩa và giao thức.
[2]ECMAScript được tạo ra để tiêu chuẩn hóa JavaScript, để thúc đẩy nhiều hiện thực độc lập

các ứng dụng web có yêu cầu cao khai thác toàn bộ khả năng của card đồ họa. Hơn nữa, WebGL không có tích hợp và tương thích với tất cả các trình duyệt chính, vì vậy người dùng không cần lo lắng về việc tải xuống hoặc cài đặt bất kỳ ứng dụng bên ngoài nào để chạy ứng dụng WebGL.

Đi cùng với sự phát triển của các phiên bản OpenGL, OpenGL ES thì các phiên bản mới của WebGL cũng ra đời. Mối quan hệ này được thể hiện qua sơ đồ sau:



Hình 1.1: Mối quan hệ giữa OpenGL, OpenGL 1.1/2.0/3.0 và WebGL. [3]

1.2.1 Cách hoạt động của WebGL

Một chương trình WebGL bao gồm các mã được viết bằng GLSL, JavaScript và HTML5. Cụ thể như sau:

- GLSL được sử dụng để viết các trình đổ bóng và các chương trình đặc biệt được thực thi trên phần cứng đồ họa tính toán các thuộc tính như đỉnh, kết cấu, màu sắc, tia chớp,... trong quá trình hiển thị đối tượng hoặc cảnh. GLSL cung cấp cho các nhà phát triển quyền kiểm soát quan trọng đối với quy trình đồ họa.
- JavaScript và HTML được sử dụng chủ yếu làm ngôn ngữ ràng buộc và để cung cấp ngữ cảnh hiển thị.

Một chương trình WebGL thường được tạo bởi nhiều lần vẽ trên phần tử

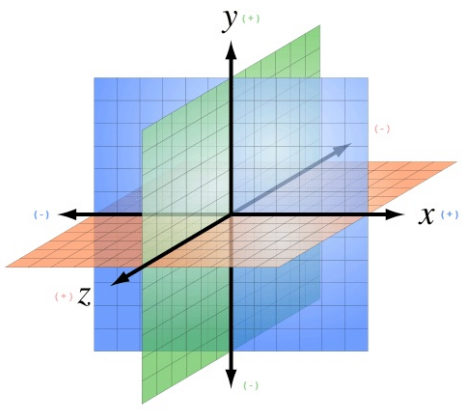
HTML canvas được thực hiện bởi đơn vị xử lý đồ họa (GPU) thông qua một quy trình được gọi là quy trình kết xuất.

Để tạo đồ họa có độ trung thực cao, sử dụng WebGL có thể khá phức tạp vì nó gần như là chương trình cấp thấp. Tuy nhiên, có nhiều thư viện sẵn có giúp trừu tượng hóa một số khó khăn của WebGL khiến chúng ít dài dòng hơn, chẳng hạn như twgl, Three.js, PhiloGL và J3D.

1.2.2 Một số thành phần cơ bản của WebGL

(a) Hệ tọa độ

Quá trình vẽ 3D diễn ra trong một hệ tọa độ 3D, trong đó có một tọa độ bổ sung z mô tả độ sâu, được hiểu là khoảng cách vào hoặc ra khỏi màn hình mà một đối tượng được vẽ. Cũng giống như hệ 3D, hệ tọa độ trong WebGL có 3 chiều x, y, z với x chạy theo chiều ngang từ trái sang phải, y chạy dọc từ dưới lên trên và z dương đi ra khỏi màn hình.



Hình 1.2: Hệ tọa độ 3D

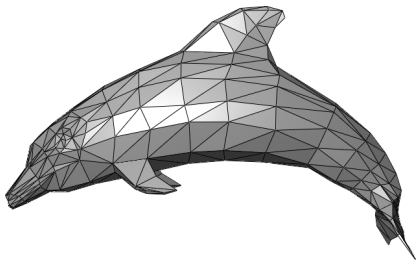
(b) Các nền tảng của WebGL

- Đỉnh (Vertices): là một điểm được giao bởi các cạnh của một đối tượng 3D, được biểu diễn bởi 3 giá trị số thực tương ứng với mỗi trục x, y, z. Trong WebGL các đỉnh được lưu trữ bằng mảng javascript.

- Các chỉ số (Indices): là các số nguyên dương dùng để định danh các đỉnh. Các chỉ số được sử dụng để vẽ các lưới trong WebGL.
- Các bộ đệm (Bufferes): là các vùng bộ nhớ của GPU cấp cho WebGL dùng để giữ dữ liệu, có các loại như
  - Vertex buffer object: lưu trữ dữ liệu tương ứng với đỉnh.
  - Index buffer object: lưu trữ dữ liệu tương ứng với chỉ số.
  - Frame buffer: là một phần của bộ nhớ đồ họa được sử dụng đẻ lưu trữ dữ liệu về các cảnh.

(c) Lưới - Mesh

Cho đến ngày nay, tuy có nhiều cách để vẽ đồ họa 3D nhưng cách phổ biến nhất vẫn là sử dụng một lưới . Lưới là một đối tượng bao gồm một hoặc nhiều hình đa giác, được xây dựng từ đỉnh (x , y , z) xác định vị trí tọa độ trong không gian 3D. Các đa giác được sử dụng phổ biến nhất trong mắt lưới là hình tam giác (nhóm ba đỉnh) và hình tứ giác (nhóm bốn đỉnh). Lưới 3D thường được gọi là mô hình 3D.



Hình 1.3: Lưới đa giác

(d) HTML5 canvas

Canvas là 1 phần tử của HTML5, cho phép thực hiện lập trình kết xuất đồ họa các đối tượng hai chiều trên trang web. Canvas chiếm một khu vực trong trang web với chiều rộng và chiều cao định trước. Sau đó sử dụng Javascript có thể truy cập vào khu vực này để vẽ thông qua một tập các hàm đồ họa tương tự như các API 2D khác.

Để viết một ứng dụng WebGL thì sẽ cần sử dụng thành phần canvas của HTML5. HTML5 canvas cung cấp cách dễ dàng để vẽ đồ họa, tạo các ảnh, làm các chuyển động đơn giản sử dụng Javascript với Cú pháp như sau:

<canvas id = "mycanvas" width = "100" height = "100"></canvas>

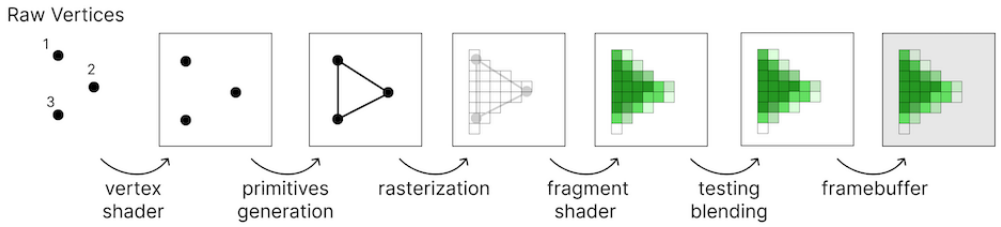
Trong đó:

- id : định danh cho canvas
- width, height : độ rộng, độ cao

HTML5 canvas có một hàm *getContext()* được dùng để vẽ lên canvas ví dụ như với tham số "2D" thì sẽ thu được 2D context dùng để vẽ các đồ họa 2D.

1.2.3 Quá trình kết xuất - Rendering Pipeline

Quá trình kết xuất bao gồm một chuỗi các thao tác được yêu cầu mỗi khi một đối tượng (cảnh 2D/3D) được hiển thị trên màn hình.



Hình 1.4: Quá trình kết xuất WebGL[3]

Trong đó:

1. **Vertices**: là thuộc tính mô tả hình học của đối tượng được biểu diễn bằng tọa độ không gian 2D hoặc 3D của nó và được lưu trữ trong một cấu trúc dữ liệu như mảng. Họ cũng có thể chỉ định các thuộc tính khác như màu sắc, kết cấu, độ phản xạ, vectơ thông thường (được sử dụng để chiếu sáng),...

[3]Nguồn: https://www.geeksforgeeks.org/webgl-introduction/

2. **Vertex shader**: là một chức năng tạo tọa độ không gian clip bằng cách áp dụng phép biến đổi cho đỉnh và giúp chuẩn bị dữ liệu để sử dụng trong trình đổ bóng phân đoạn.

3. **Primitive generation**: kết nối các đỉnh để tập hợp các nguyên hàm tam giác.

4. **Rasterization**: là quá trình chiếu tam giác được mô tả trước đây dưới dạng tọa độ vectơ thành các mảnh nhỏ gọi là mảnh (dữ liệu cần thiết để tạo pixel màn hình) và nó cũng nội suy các giá trị trên các giá trị nguyên thủy cho mỗi pixel.

5. **Fragment shader**: là hàm tính toán và gán màu cho pixel.

6. **Testing blending**: lấy đầu ra của trình đổ bóng phân đoạn và kết hợp nó với các màu trong bộ đệm khung hiện tại để hiển thị các hiệu ứng trong suốt và mờ.

7. **Framebuffer**: lưu trữ khung dữ liệu trong bộ nhớ GPU đại diện cho nội dung hiển thị của màn hình

Chương 2

THƯ VIỆN THREE.JS

Tất cả các trình duyệt hiện đại trở nên mạnh mẽ hơn và dễ truy cập hơn trực tiếp bằng JavaScript. Đồng thời, áp dụng WebGL , API JavaScript, cho phép hiển thị đồ họa 3D và 2D tương tác hiệu suất cao trong bất kỳ trình duyệt web tương thích nào bằng cách sử dụng khả năng của bộ xử lý đồ họa (GPU). Tuy nhiên, lập trình WebGL rất phức tạp, cần viết nhiều code, dễ gặp lỗi. Mà vấn đề này đã được giải quyết một cách rất hiệu quả bằng việc sử dụng thư viện Three.js. Đây là một thư viện JavaScript để tạo và hiển thị đồ họa 3D trên trình duyệt sử dụng WebGL, giúp việc lập trình 3D trên trình duyệt dễ dàng hơn.

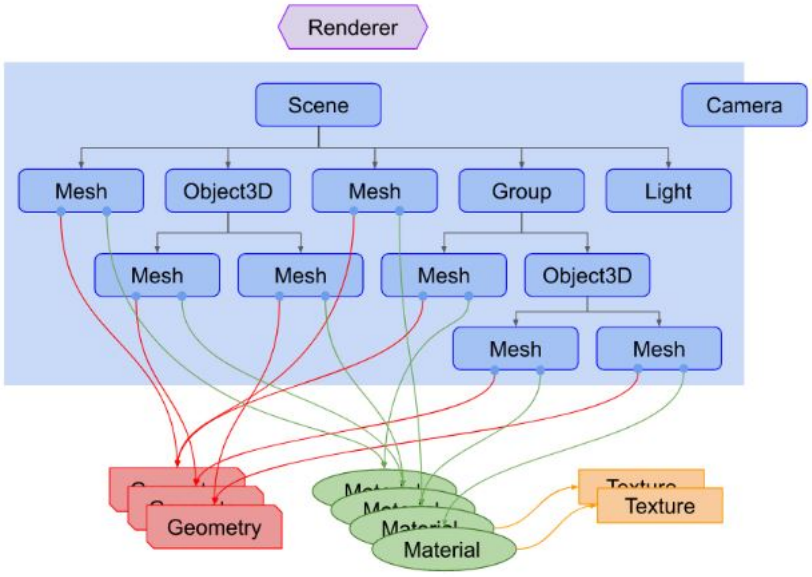
2.1 Cấu trúc ứng dụng Three.js

Three.js là một thư viện JavaScript đa năng, mã nguồn mở, nhẹ, đa trình duyệt được phát hành trên GitHub vào năm 2010 bởi Ricardo Cabello (mr.doob). Bằng cách sử dụng WebGL, ta có thể trực tiếp sử dụng các tài nguyên xử lý của card đồ họa (GPU) và tạo các cảnh 3D với hiệu năng cao. Thư viện Three.js chịu trách nhiệm tạo hiệu ứng ánh sáng, đổ bóng, vật liệu, kết cấu và mô hình hình học 3D, những thứ rất khó tạo trong WebGL. Vì Three.js sử dụng JavaScript nên bạn có thể tương tác với các thành phần trang web khác, thêm hình ảnh động và tương tác, thậm chí tạo trò chơi với một số logic. Và với việc sử dụng nó, ta có thể làm hầu hết mọi thứ mà ta có thể tưởng tượng.

2.1.1 Các thành phần cần thiết

- Một đối tượng Scene: Là một khung cảnh mà chứa mọi đối tượng 3D tồn tại, nó giống như là 'vũ trụ thu nhỏ' vậy.
- Một camera: Cũng tương tự như là camera trong thế giới thực, được sử dụng để nhìn khung cảnh (scene).
- Một đối tượng WebGLRenderer: 1 thứ 'máy móc' với input là camera, scene và output là những hình ảnh được vẽ.
- Canvas: Phần tử HTML canvas, cái này giống như 1 bức tranh (hay cái màn chiếu) trống, và three.js sẽ vẽ lên đây.

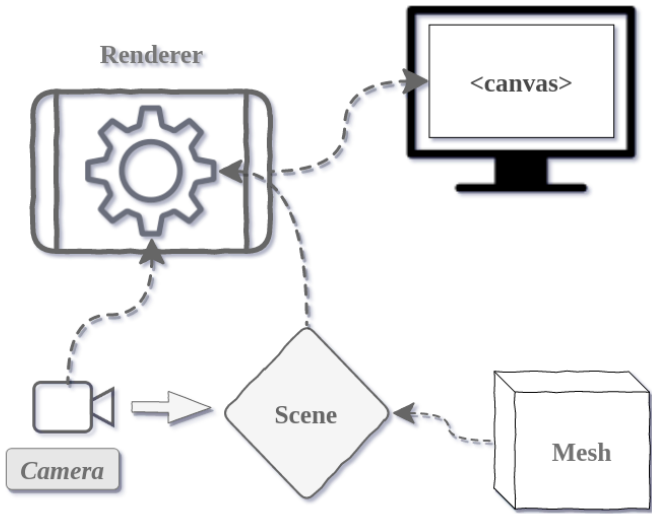
2.1.2 Mô hình cấu trúc



Hình 2.1: Cấu trúc ứng dụng Three.js

Hiểu đơn giản như này: camera của điện thoại chính là camera, bạn chụp ảnh thì chỉ có thể chụp được trong tầm nhìn của camera (xa thì bị mờ), vì vậy camera chính là thứ quyết định bạn có thể xem được scene gì. Lúc này điện thoại đóng vai trò là renderer lấy ra các scene được camera ghi lại.

Ngoài ra, có thể thêm thành phần canvas để có thể chiếu các scene đã được lấy ra từ renderer.



Hình 2.2: Cấu trúc của Three.js có thêm thành phần canvas[1]

2.2 Một số thành phần cơ bản của Three.js

2.2.1 Máy ảnh

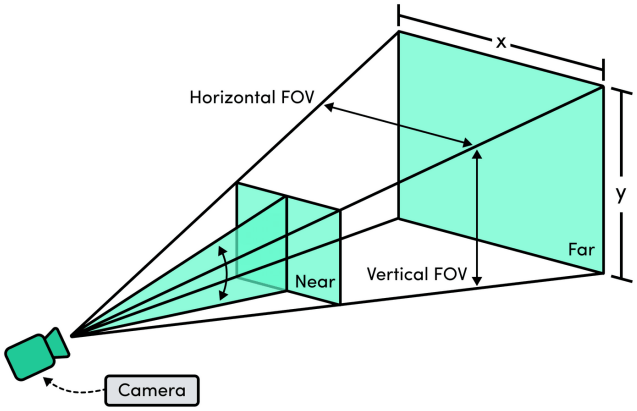
Trong Three.js, có hai loại máy ảnh là OrthographicCamera và PerspectiveCamera. Tuy rằng, PerspectiveCamera phức tạp hơn nhưng nó sẽ cho ra hình ảnh giống thế giới thực hơn so với OrthographicCamera.

(a) PerspectiveCamera

Một PerspectiveCamera sẽ mô phỏng hành động của một máy ảnh quay phim trong đời thực. Đối tượng càng xa máy ảnh thì trông càng bé. Vị trí của camera và hướng của nó sẽ quyết định phần nào của khung cảnh được render trên màn hình.

Khi khởi tạo một máy ảnh mới, cần truyền vào các tham số xác định vùng quan sát là một hình chóp cụt:

[1]Nguồn: https://viblo.asia/p/threejs-bai-1-lam-quen-voi-moi-truong-3d-cua-threejs-vyDZO7ROZwj



Hình 2.3: PerspectiveCamera

Thuộc tính	**Giá trị**	**Ý nghĩa**
fov	0 - 360	Góc quan sát, mô phỏng độ rộng của góc nhìn, tính bằng độ. Giá trị phù hợp với mắt người là dưới 180, thường dùng 45-50.
aspect	số thực dương	Tỉ lệ của vùng quan sát: chiều rộng/chiều cao
near, far	Số thực dương, $0 < \text{near} < \text{far}$	Khoảng cách gần nhất và xa nhất máy ảnh có thể quan sát được.

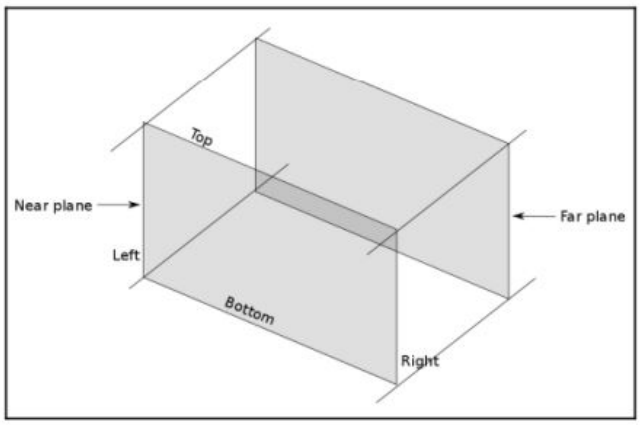
Bảng 2.1: Bảng các thuộc tính của PerspectiveCamera

Mỗi khi thay đổi các tham số này cần cập nhật ma trận chiếu bằng lời gọi hàm *updateProjectMatrix()*.

Để chỉ định điểm máy ảnh hướng tới, dùng *lookAt(p)*.

Vector hướng nhìn được tính sau khi biết điểm máy ảnh hướng tới: v = điểm máy ảnh nhìn tới – vị trí của máy ảnh.

(b) OrthographicCamera



Hình 2.4: OrthographicCamera

Khi khởi tạo một máy ảnh OrthographicCamera, cần truyền vào các tham số xác định vùng quan sát là một hình hộp:

Thuộc tính	**Giá trị**	**Ý nghĩa**
left, right	Số thực, left < right	Tọa độ mặt trái và mặt phải (vuông góc với trục x) của vùng quan sát.
top, bottom	bottom < top	Tọa độ mặt trên, dưới (vuông góc với trục y) của vùng quan sát.
near, far	0 <= near < far	Khoảng cách gần nhất và xa nhất máy ảnh có thể quan sát được.

Bảng 2.2: Bảng các thuộc tính của OrthographicCamerra

2.2.2 Trình kết xuất

Trình kết xuất có nhiệm vụ phân tích mọi thứ trên khung cảnh như vị trí của các đối tượng trong mối quan hệ với các đối tượng khác, với tọa độ thế giới,... và hiển thị nó trên canvas.

Trong Three.js, có các trình kết xuất sau: WebGLRenderer, WebGL1Renderer, CSS2DRenderer, CSS3DRenderer, SVGRenderer. Tuy nhiên, chỉ cần sử dụng

WebGLRenderer là đã có thể tận dụng sức mạnh của WebGL. Để khởi tạo một đối tượng Renderer mới, sử dụng câu lệnh như sau:

const renderer = new WebGLRenderer(parameters);

Trong đó, tham số parameters là một đối tượng với các thuộc tính định nghĩa các hành vi của trình kết xuất ví dụ như:

- canvas: chỉ định phần tử DOM canvas trong trang để vẽ đầu ra, tương ứng với thuộc tính domElement của đối tượng trình kết xuất. Nếu không truyền ở đây, một phần tử canvas mới sẽ được tạo bằng phương thức *appendChild()*:

  document.body.appendChild(renderer.domElement);

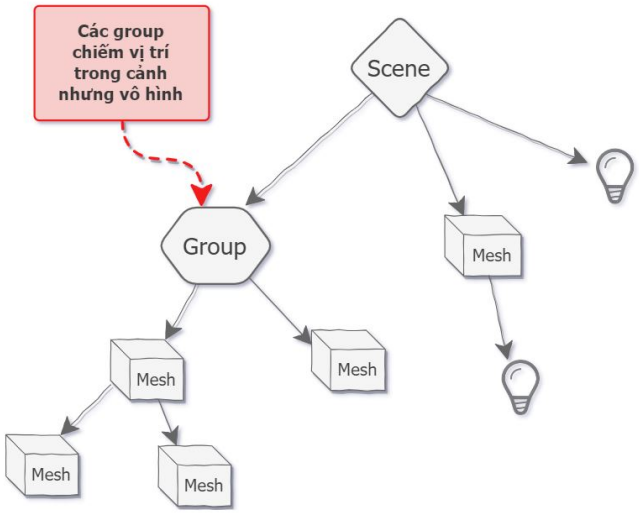
- antialias: chỉ định có thực hiện xử lý antialiasing hay không, mặc định là false. Nếu để antialias bằng true thì các đường thẳng sẽ sắc nét hơn, không bị trông giống bậc thang hoặc răng cưa.

Một số phương thức thương được sử dụng của trình kết xuất là *setClearColor()*, *setSize()*, *render()* với thông số cụ thể:

- setClearColor(color: Color, alpha: Float)
- setSize(width: Integer, height: Integer, updateStyle: Boolean)
- render(scene: Object3D, camera: Camera)

2.2.3 Khung cảnh

Khung cảnh là một đối tượng mà ở đó tất cả các yếu tố khác đều tồn tại. Mọi thứ mà chúng ta thấy trong kết xuất cuối cùng thường được chứa trong một khung cảnh, vì vậy nó giống như một thùng chứa gốc cho những thứ khác. Trong Three.js, khung cảnh chứa tất cả các đối tượng, nguồn sáng, và các đối tượng khác cần thiết để kết xuất.



Hình 2.5: Thành phần của một khung cảnh dưới dạng cây

2.2.4 Đối tượng

Bất cứ thứ gì bên trong khung cảnh thì đều được coi là một đối tượng. Một số đối tượng cơ bản được sử dụng như lưới, hình học, kết cấu, vật liệu, đèn,...

(a) Lưới

Thông thường, các hình dạng được hiển thị bên trong khung cảnh bằng cách sử dụng các đối tượng lưới. Lưới là đối tượng dựa trên sự kết hợp giữa hình học và vật liệu, tương tự như cấu tạo bên ngoài của con người sẽ cần có khung xương và lớp da. Để tạo ra một đối tượng trong Three.js, câu lệnh thực hiện như sau:

const mesh = new THREE.Mesh(geometry, material);

(b) Hình học

Hình học giống như một khung xương, một hình dạng của vật thể được tạo ra từ các đỉnh, các cạnh và các mặt. Three.js có một tập nhiều các hình học sẵn mà không cần phải tự định nghĩa tất cả các đỉnh cũng như các mặt, việc cần làm nhập vào các thông số cho hình học mà muốn sử dụng. Một số hình học cơ bản là:

- BoxGeometry: hình hộp, ví dụ như bức tường

- PlaneGeometry: mặt phẳng
- SphereGeometry: hình cầu, ví dụ như quả bóng, trái đất
- TorusGeometry: hình vòng, ví dụ như bánh xe, bánh donut
- TorusKnotGeometry: hình vòng có nút thắt
- CylinderGeometry: hình trụ

(c) Vật liệu

Vật liệu là một yếu tố để tạo ra lưới, giống như lớp vỏ của một khối hình học và nó xác định đối tượng sẽ trông như thế nào trong khung cảnh, nó sẽ tương tác với ánh sáng như thế nào thông qua các tham số như trong suốt, khung dây, sáng bóng, thô ráp, giống kim loại,...

Tên	**Mô tả**
MeshBasicMaterial	Màu sắc của vật liệu không phụ thuộc vào ánh sáng.
MeshLambertMaterial	Màu sắc vật liệu được tính theo công thức tô bóng Gouraud.
MeshPhongMaterial	Có ánh sáng phản chiếu, phụ thuộc 2 tham số: màu phản chiếu và hệ số phản chiếu.
MeshStandardMaterial	Vật liệu này sử dụng physically based rendering. Một mẫu vật ký được sử dụng để quyết định cách ánh sáng tương tác với các bề mặt. Điều này cho phép bạn tạo các đối tượng chính xác và chân thật hơn.
MeshPhysicalMaterial	Một mở rộng của MeshStandardMaterial cho phép nhiều điểu chỉnh hơn về sự phản xạ.
MeshNormalMaterial	Màu sắc vật được tính theo vector pháp tuyến của bề mặt, không phụ thuộc vào các nguồn sáng.
MeshDepthMaterial	Sử dụng khoảng cách từ máy ảnh đến đối tượng để quyết định màu sắc. Càng gần thì màu trắng, càng xa thì màu đen. Sự thay đổi giữa màu trắng và màu đen dựa vào các giá trị khoảng cách near và far của máy ảnh.

Bảng 2.3: Bảng mô tả các vật liệu được sử dụng

Mỗi vật liệu sẽ có các thuộc tính khác nhau có thể tùy chỉnh để phù hợp với đối tượng.

Thuộc tính	Ý nghĩa	Chú thích
side	vật liệu có 1 hay 2 mặt	
shadowSide	mặt trước, sau, hay cả 2 mặt tạo ra bóng	
colorWrite	false: vật liệu vô hình nhưng vẫn chắn sáng (dùng thuộc tính renderOrder để gán thứ tự vẽ phù hợp).	
transparent	Vật liệu có cho ánh sáng đi qua hay không.	
opacity	Hệ số cản sáng.	
color	Màu của vật liệu. Mặc định là màu trắng.	
vertexColors	Màu của vertex	
needsUpdate	Một số thuộc tính vật liệu (như fog, texture...) nếu cần thay đổi giá trị khi chạy thì phải chỉ định needsUpdate = true.	
id, uuid, name	Mã và tên vật liệu	
visible	Ẩn/hiện	
wireframe	Vẽ khung dây hay vẽ bề mặt	
flatShading	Tính màu sắc bề mặt theo nguyên lý tô bóng phẳng (dựa vào vector pháp tuyến bề mặt)	
emissive	Màu bức xạ từ vật liệu ra môi trường, không bị ảnh hưởng bởi các nguồn sáng khác. Mặc định là màu đen.	Lambert Phong
shininess	Hệ số này càng cao, vật càng bóng	Phong
specular	Màu phản chiếu (hiệu ứng gương).	Phong
metalness	Thể hiện tính chất kim loại của vật liệu. Hệ số càng cao vật liệu càng phản ứng với ánh sáng giống kim loại. Hệ số này nhận giá trị trong miền: [0, 1]. Giá trị mặc định là 0.5.	Standard Physical
roughness	Thuộc tính này dùng để xác định thành phần ánh sáng khuếch tán. Hệ số này nhận giá trị trong miền:[0, 1]. Giá trị mặc định là 0.5	Standard Physical
clearCoat	độ dày của lớp mạ bên ngoài vật liệu. Giá trị càng lớn càng làm rõ hiệu ứng của thuộc tính clearCoatRoughness	Physical
clearCoatRoughness	Độ nhám của lớp mạ.	Physical
reflectivity	thuộc tính này dành cho vật liệu phi kim loại hoặc tính kim loại thấp (hệ số metalness gần hoặc bằng 0). Thể hiện khả năng phản xạ ánh sáng của vật liệu.	Physical

Bảng 2.4: Bảng các thuộc tính của vật liệu

(d) Kết cấu

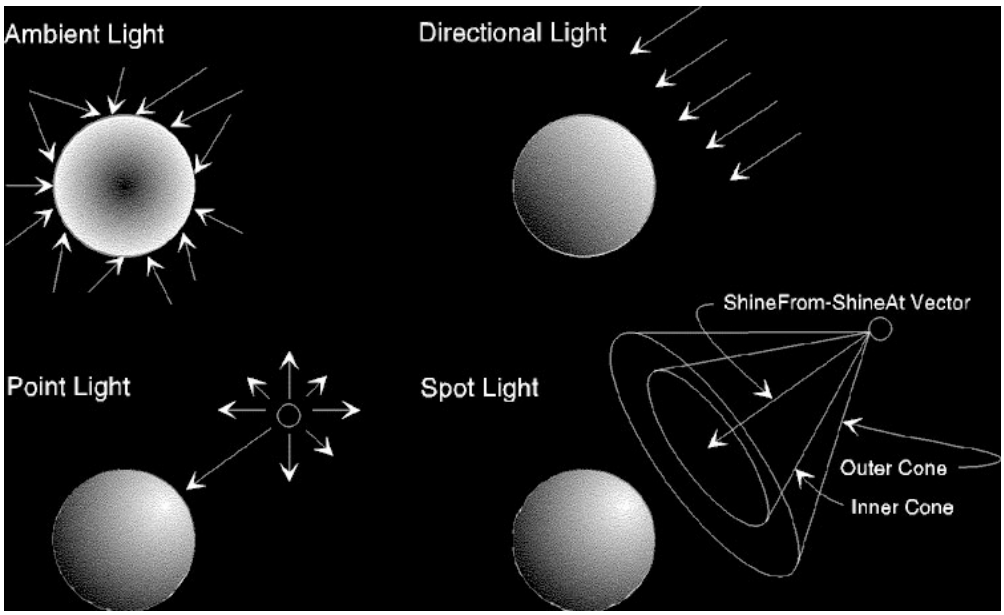
Để thay đổi đối tượng 3D, ngoài việc thay đổi các thuộc tính của vật liệu thì ta có thể sử dụng một hình ảnh và kéo nó bao phủ bề mặt của đối tượng đó, có thể gọi là quá trình khảm. Lúc này, màu sắc của đối tượng sẽ giống màu của ảnh. Ảnh trong trường hợp này được gọi là kết cấu (texture).

- Ảnh sử dụng cho kết cấu : Có thể sử dụng các định dạng ảnh thông dụng như PNG, JPG, GIF, BMP. Để cho kết quả tốt nhất, kích thước ảnh nên là số mũ của 2 (ví dụ 256x256, 512x512, 1024x1024). Nếu kích thước của ảnh không phải số mũ của 2, Three.js sẽ scale ảnh về giá trị số mũ của 2 gần nhất.
- Các bước chung để khảm một hình
  - Tạo một đối tượng đọc ảnh, ví dụ TextureLoader
  - Tạo một texture và đọc dữ liệu ảnh đã chọn vào texture
  - Gán texture cho thuộc tính map của vật liệu
- Các loại ánh xạ sử dụng kết cấu
  - Color Map: định nghĩa màu sắc của đối tượng từ texture bằng cách gán texture cho thuộc tính **map** của vật liệu.
  - Bump map: được sử dụng để mô phỏng sự gồ ghề, lồi lõm của một bề mặt, thường được lưu ở một ảnh đen trắng, màu đen là điểm có độ cao nhỏ nhất, màu trắng là điểm có độ cao lớn nhất. Thuộc tính của vật liệu để gán Texture là **bumpMap**. Ngoài ra, còn có thuộc tính **bumpScale** để chỉ định tỷ lệ độ cao là lớn hay nhỏ.
  - Normal map: là biến thể của bump map, sử dụng vector pháp tuyến ở từng điểm, các thông số màu RGB tương ứng với các tọa độ XYZ, chi tiết hơn so với Bump map. Thuộc tính của vật liệu để gán Texture là **normalMap**. Ngoài ra, còn có thuộc tính **normalScale** để có thể thiết lập tỷ lệ theo trục X và Y.
  - Displacement map: hình dạng của đối tượng bị chỉnh sửa như nó

bị thay thế so với Bump map, chỉ có kết quả tốt khi đối tượng của chúng ta chứa nhiều đỉnh. Thuộc tính của vật liệu để gán Texture là **displacementMap**. Ngoài ra, còn có thuộc tính **displacementScale** là tỷ lệ thay thế.

- Alpha map: điều chỉnh độ trong suốt của bề mặt. Nếu giá trị của map là màu đen, phần đó của đối tượng sẽ trong suốt hoàn toàn, và nếu giá trị là màu trắng, phần đó sẽ đục hoàn toàn. Thuộc tính của vật liệu để gán Texture là **alphaMap**. Ngoài ra, còn phải thiết lập thuộc tính **transparent = true** và thuộc tính **side:DoubleSide** để có thể nhìn được mặt trong của hình.
- Emissive map: có thể được sử dụng để làm các phần nào đó của đối tượng phát sáng. Thuộc tính của vật liệu để gán Texture là **emissiveMap**. Đồng thời, thiết lập thuộc tính **emissive** là màu gì đó khác màu đen để nó kết hợp với emissive map. Hai giá trị màu này sẽ được nhân với nhau để ra kết quả hiển thị cuối cùng.

(e) Nguồn sáng



Hình 2.6: Các loại nguồn sáng

Light	Helper	Tạo bóng	Hướng tia sáng
AmbientLight			Vô hướng, tác động đều, toàn cục
HemisphereLight	HemisphereLightHelper		Theo hai hướng: từ trên xuống và từ dưới lên Mặt hướng lên trên chịu tác động của màu bầu trời Mặt hướng xuống dưới chịu tác động của màu mặt đất
DirectionalLight	DirectionalLightHelper	Có	Hướng từ position đến target.position
SpotLight	SpotLightHelper	Có	Hướng từ position đến target.position
PointLight	PointLightHelper	Có	Tỏa ra tất cả các hướng
RectAreaLight	RectAreaLightHelper		Dùng góc xoay

Bảng 2.5: Đặc điểm của các nguồn sáng

Chi tiết thuộc tính của một số loại nguồn sáng hay được sử dụng:

- Hai thuộc tính cơ bản
  - color: màu sắc
  - intensity: cường độ
- **PointLight**: ngoài 2 thuộc tính cơ bản, còn có
  - castShadow: có tạo bóng không.
  - position: vị trí đặt nguồn sáng.
  - distance: khoảng cách giới hạn chiếu sáng. Nếu bằng 0 thì các tia sáng sẽ chiếu vô hạn. Nếu distance lớn hơn 0 thì các tia sáng sẽ chiếu với cường độ mạnh nhất ở nguồn và giảm dần đến khoảng cách distance.
- **SpotLight**: chứa thuộc tính của PointLight và có thêm các thuộc tính khác
  - angle: Tính theo radian, cho biết độ rộng của góc rọi.
  - penumbra: Làm trơn biên vùng được chiếu sáng. Nhận giá trị nằm trong [0, 1], mặc định là 0.
  - target: Đối tượng được chiếu sáng.

Chương 3

MÔ HÌNH 3D

Ở chương 2, báo cáo đã trình bày một số hình học và vật liệu cơ bản mà thư viện Three.js cung cấp để giúp tạo ra các lưới hay đối tượng 3D đơn giản. Thực chất, hình học của các đối tượng trên có thành phần chính là các đa giác, được sử dụng phổ biến nhất là "tam giác". Ngoài ra, còn có các "hình tứ giác" và "n-cạnh" có nhiều đỉnh. Mỗi đa giác được kết nối với các đa giác khác và chúng cùng nhau tạo ra một lưới đa giác, về cơ bản đó là một mô hình 3D [4]. Nhưng trong thực tế, ý nghĩa của việc xây dựng các đối tượng 3D này là tạo ra mô hình 3D phức tạp với các chi tiết giống đời thực, có thể mang yếu tố hư cấu trong đó. Có rất nhiều cách khác nhau để có thể tạo các hình học và lưới nâng cao, phức tạp, được coi là yếu tố chính tạo ra các sản phẩm 3D hiện nay. Trong chương này, báo cáo sẽ trình bày về hai cách tiếp cận để xây dựng chúng, đó là:

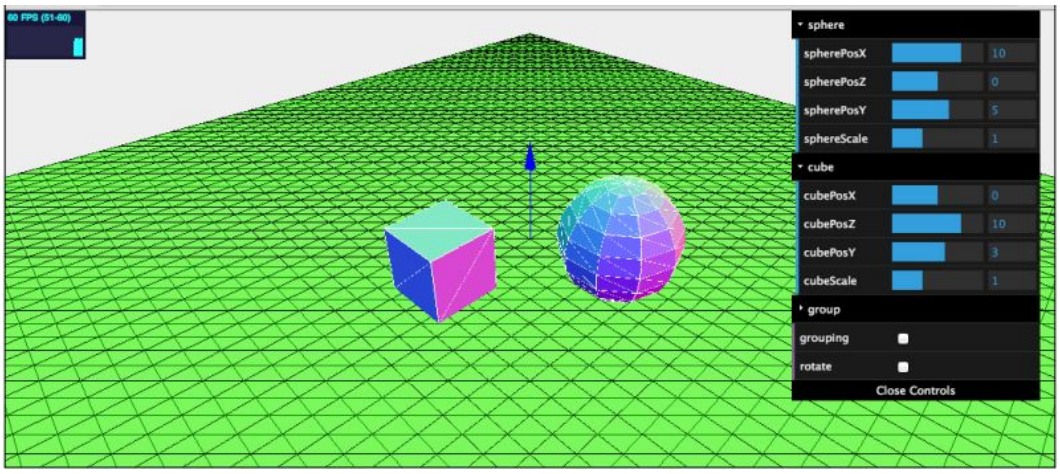
- Nhóm các đối tượng và hợp nhất các lưới
- Tải các mô hình có sẵn

3.1 Nhóm các đối tượng và hợp nhất các lưới

3.1.1 Gộp các đối tượng khác nhau thành một nhóm

Three.js có chức năng tạo ra nhóm (Group) khi ta muốn tạo ra một lưới từ một hình học bằng nhiều vật liệu khác nhau. Khi đó, nhiều bản sao hình học sẽ

được thêm vào, mỗi bản sao này sẽ có vật liệu cụ thể của riêng nó. Kết quả trả về sẽ trông giống như một lưới sử dụng nhiều vật liệu, nhưng thực tế thì nó là một nhóm chứa một số lượng lưới nhất định.



Hình 3.1: Gộp các đối tượng khác nhau thành một nhóm [1].

Việc tạo nhóm được thực hiện dễ dàng với 4 bước sau:

1. Tạo một đối tượng Group
2. Tạo các lưới cần thiết
3. Thêm các lưới vào Group
4. Thêm Group vào scene

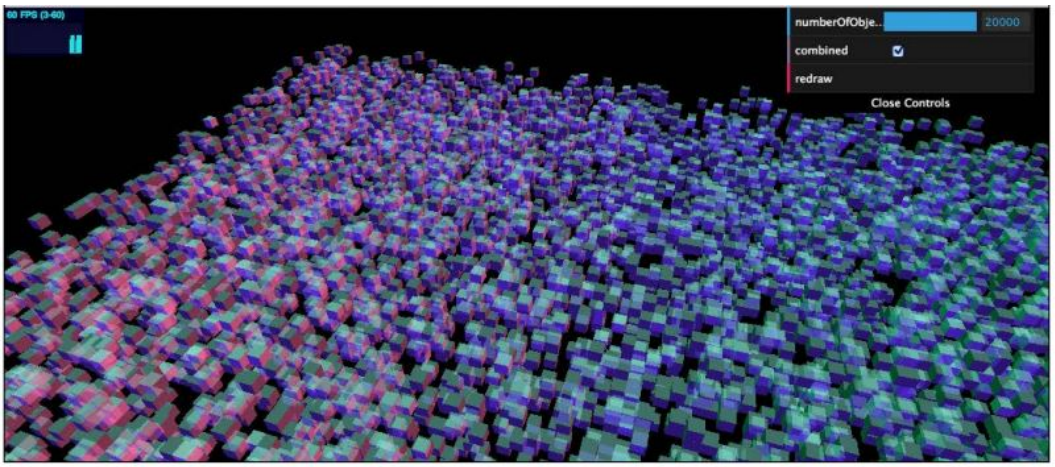
Tác dụng của việc gộp nhiều đối tượng thành 1 nhóm như vậy là để thuận lợi cho các biến đổi hình học như di chuyển, chia tỷ lệ, xoay và dịch các đối tượng cùng nhau. Khi sử dụng nhóm, ta vẫn có thể tham khảo, sửa đổi và định vị trí của các hình học riêng lẻ. Điều duy nhất cần nhớ là tất cả các vị trí, góc quay, và bản dịch được thực hiện liên quan đến đối tượng gốc.

3.1.2 Hợp nhất các lưới hình học khác nhau thành một lưới

Trong đa số các trường hợp, sử dụng nhóm cho phép dễ dàng thao tác và quản lý số lượng lớn lưới. Tuy nhiên, trong trường hợp các lưới trên trở thành các đối tượng, với số lượng xử lý lớn như vậy thì hiệu suất sẽ trở thành một vấn đề. Giải

pháp là thay vì gộp các đối tượng ta chỉ gộp các lưới hình học của chúng, sau đó tạo ra một đối tượng duy nhất từ lưới thu được. Các bước thực hiện như sau:

1. Khai báo 1 biến kiểu Geometry để chứa lưới gộp
2. Với mỗi object:
    - Cập nhật ma trận biến đổi của object.
    - Bổ sung lưới hình học của object vào geometry bằng lệnh merge()
3. Tạo một object với lưới gộp thu được.
4. Thêm object vừa tạo vào scene

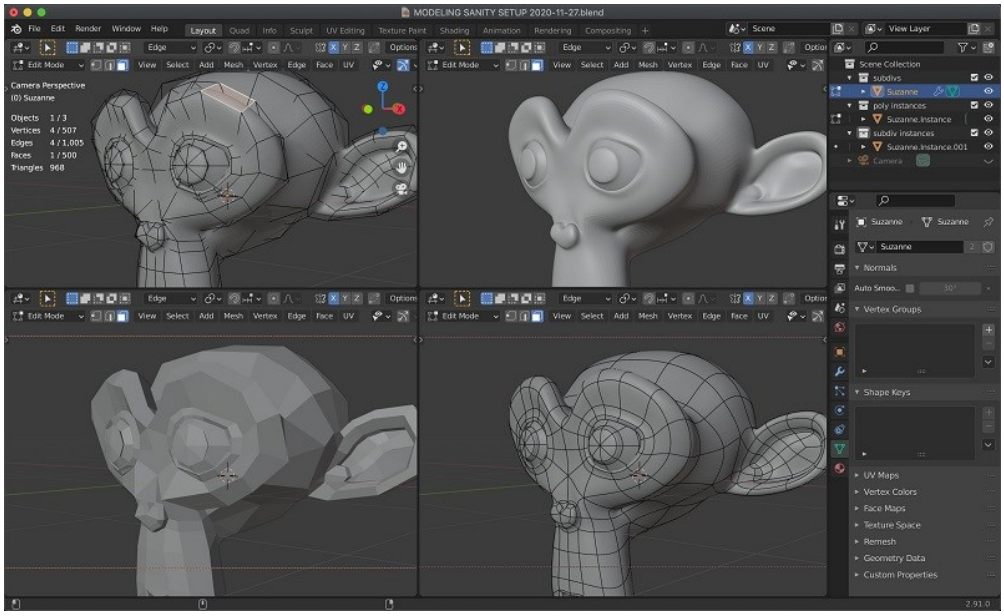


Hình 3.2: Hợp nhất các lưới hình học khác nhau thành một lưới [1].

Với hình ảnh trên, có thể thấy với 20.000 khối lập phương được thêm vào thì việc kết xuất vẫn dễ dàng mà không làm giảm hiệu suất [1]. Việc hợp nhất lưới hình học của từng đối tượng vào một hình học duy nhất rồi mới được thêm vào khung cảnh cũng được coi là một nhược điểm của phương pháp này. Bởi khi đó, ta sẽ không thể kiểm soát đối như muốn di chuyển, xoay hoặc chia tỷ lệ với các hình khối riêng lẻ (trừ khi bạn tìm kiếm chính xác mặt, đỉnh và định vị của chúng).

3.2 Tải các mô hình có sẵn

Trong thực tế, việc tạo ra các mô hình 3D phức tạp sẽ được coi là rất khó khăn nếu làm thủ công bằng cách kết hợp các hình học cơ bản mà Three.js cung cấp. Do đó, để việc này trở nên dễ dàng hơn, một số phần mềm chuyên dụng đã ra đời như Blender, Maya,... để hỗ trợ tạo ra các mô hình 3D. Lúc này, việc cần làm chỉ là tải mô hình vào trong khung cảnh và hiển thị. Ở phần này, báo cáo giới thiệu qua một số định dạng được hỗ trợ từ Three.js và sẽ tập trung trình bày kỹ hơn về mô hình có định dạng OBJ đi kèm với các tham số chứa trong tệp có định dạng MTL.



Hình 3.3: Mô hình 3D được tạo trên Blender

3.2.1 Một số định dạng mô hình 3D

Three.js cho phép đọc một số các định dạng của mô hình 3D và tải lên hình học cũng như lưới được xác định trong các tệp đó. Mục đích cơ bản của định dạng tệp 3D là lưu trữ thông tin về các mô hình 3D dưới dạng văn bản thuần hoặc dữ liệu nhị phân. Dưới đây là bảng mô tả một số định dạng do Three.js cung cấp:

Định dạng	Mô tả
JSON	Three.js có định dạng JSON riêng mà bạn có thể sử dụng để xác định một cách khai báo một hình học hoặc một cảnh. Mặc dù đây không phải là định dạng chính thức nhưng nó rất dễ sử dụng và rất hữu ích khi bạn muốn sử dụng lại các hình học hoặc cảnh phức tạp.
OBJ & MTL	OBJ là một định dạng 3D đơn giản được phát triển đầu tiên bởi Wavefront Technologies. Đây là một trong những định dạng tệp 3D được sử dụng rộng rãi nhất và được sử dụng để xác định hình dạng của một đối tượng. MTL là định dạng đồng hành với OBJ. Trong tệp MTL, tài liệu của các đối tượng trong tệp OBJ được chỉ định.
Collada	Collada là một định dạng để xác định nội dung kỹ thuật số ở định dạng dựa trên XML. Đây cũng là một định dạng được sử dụng rộng rãi, được hỗ trợ bởi khá nhiều ứng dụng 3D và công cụ kết xuất.
STL	STL là viết tắt của STereoLithography và được sử dụng rộng rãi để tạo mẫu nhanh. Các mô hình cho máy in 3D thường được định nghĩa là tệp STL. Three.js cũng có một trình xuất STL tùy chỉnh, có tên là STLExporter.js, nếu bạn muốn xuất các mô hình của mình sang STL từ Three.js.
CTM	CTM là một định dạng tệp được tạo bởi openCTM. Nó được sử dụng làm định dạng để lưu trữ các mắt lưới dựa trên hình tam giác 3D ở định dạng nhỏ gọn.
VTK	VTK là định dạng tệp được xác định bởi Bộ công cụ trực quan hóa và được sử dụng để chỉ định các đỉnh và mặt. Có sẵn hai định dạng và Three.js hỗ trợ định dạng ASCII cũ.
PDB	Đây là một định dạng rất chuyên biệt, do Ngân hàng dữ liệu Protein tạo ra, được sử dụng để xác định hình dạng của protein. Three.js có thể tải và trực quan hóa các protein được chỉ định ở định dạng này.
PLY	Định dạng này được gọi là định dạng tệp đa giác. Điều này thường được sử dụng để lưu trữ thông tin từ máy quét 3D.

Bảng 3.1: Bảng mô tả một số định dạng của mô hình 3D do Three.js cung cấp [1].

Ngoài ra, trong những năm gần đây, một định dạng nổi lên như là chuẩn, đó là glTF (GL Transmission Format) với ưu điểm là mã nguồn mở, được thiết kế tối ưu để hiển thị chứ không phải để chỉnh sửa, dung lượng nhẹ, tải nhanh, có đầy đủ các tính năng như Material, Animation. Định dạng glTF này có thể ở hai dạng như sau:

- Dạng tệp JSON chuẩn .gltf không nén và có thể đi kèm thêm với các tệp

.bin

- Dạng tệp nhị phân .glb chứa tất cả dữ liệu trong chỉ một tệp.

Với mỗi định dạng sẽ có hàm gọi đến tương ứng, ví dụ như với định dạng glTF hay OBJ sẽ có hàm gọi đến tương ứng là GLTFLoader, OBJLoader.

3.2.2 Định dạng tệp OBJ và MTL của mô hình 3D

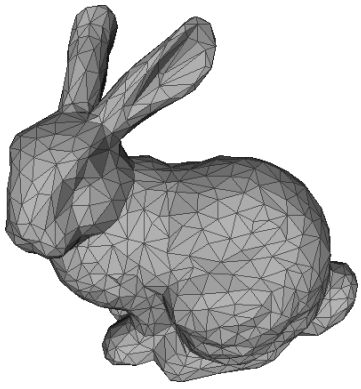
(a) Định dạng tệp OBJ

Định dạng tệp OBJ được coi là một trong những định dạng quan trọng được sử dụng rộng rãi trong các ứng dụng đồ họa 3D. Nó có thể mã hóa và lưu trữ thông tin của các đối tượng hình học. Cả hình học đa giác như điểm, đường thẳng, đỉnh kết cấu, mặt và hình học dạng tự do (đường cong và bề mặt) đều được định dạng OBJ hỗ trợ. Định dạng này không hỗ trợ chuyển động hoặc thông tin liên quan đến ánh sáng và vị trí của cảnh [5].

Định dang tệp OBJ cho thấy sự linh hoạt trong việc thực hiện mã hóa hình dạng bề mặt của các mô hình 3D với nhiều phương pháp khác nhau.

- **Các mặt đa giác**

  Ở dạng cơ bản nhất, định dạng tệp OBJ cho phép người dùng tạo ra bề mặt của mô hình 3D bằng các hình dạng hình học đơn giản như hình tam giác, hình tứ giác hoặc đa giác phức tạp hơn.



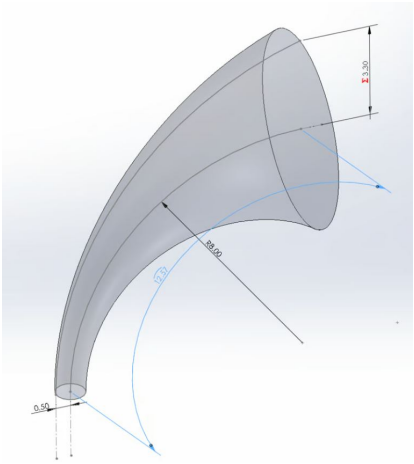
Hình 3.4: Mô hình 3D của con thỏ có bề mặt được tạo từ các hình tam giác.

Nhược điểm của phương pháp này là bề mặt mô hình 3D sẽ có một độ

thô nhất định do được ghép từ các đa giác với nhau. Để làm giảm độ thô thì ta có thể làm giảm kích thước của các đa giác. Tuy nhiên, việc đó lại làm tăng số lượng đa giác cần thiết để che phủ hết bề mặt mô hình, dẫn đến kích thước tệp chứa nó trở lên khổng lồ và khó làm việc. Do đó, điều quan trọng trong phương pháp này chính là tìm ra sự cân bằng giữa 2 yếu tố trên để thu được mô hình 3D có độ thô tốt nhất.

- **Đường cong tự do**

  Với phương pháp thứ hai, định dạng tệp OBJ cho phép mã hóa hình dạng bề mặt của mô hình 3D bằng một tập hợp các đường cong tự do (Cardinal Splines, Bezier curves,...) chạy dọc theo bề mặt mô hình.



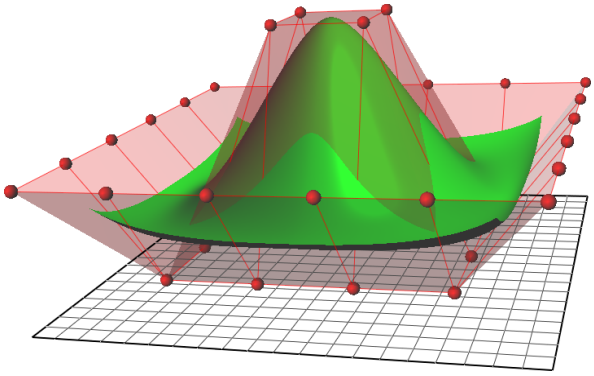
Hình 3.5: Đường cong tự do trên bề mặt mô hình 3D

Với việc sử dụng ít tham số toán học hơn thì các đường cong dạng tự do phức tạp hơn so với các mặt đa giác. Mặt khác, nó cũng là một ưu điểm về mặt lưu trữ dữ liệu. Do yêu cầu ít dữ liệu nên các đường cong dạng tự do được sử dụng để tạo mã hóa chất lượng cao cho bất kỳ mô hình 3D nào mà không cần mở rộng kích thước tệp.

- **Bề mặt tự do**

  Với định dạng tệp OBJ, ta cũng có thể chỉ định việc ốp lát hình học bề mặt với các mảng bề mặt dạng tự do thay vì đa giác đơn giản. Loại miếng vá bề mặt dạng tự do (NURBS - Non-uniform rational basis spline) này rất hữu ích với các bề mặt không có kích thước xuyên tâm

cứng như thân xe tải, cánh máy bay trực thăng hoặc thân tàu.



Hình 3.6: Bề mặt NURBS

Phương pháp này có ưu điểm là kích thước tệp nhỏ hơn với độ chính xác cao hơn so với các phương pháp khác.

Dưới đây là ví dụ một phần của tệp OBJ như sau:

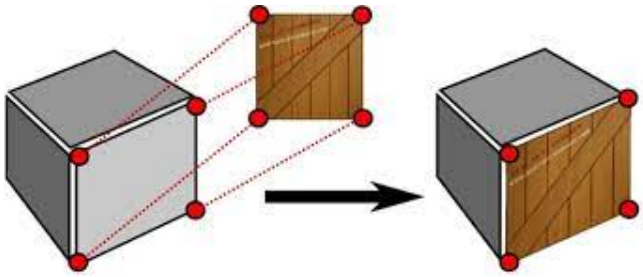
```
v -0.032442 0.010796 0.025935
v -0.028519 0.013697 0.026201
v -0.029086 0.014533 0.021409
usemtl Material
f 2731 2735 2736 2732
f 2732 2736 3043 3044
```

Trong đó:

- v : Đỉnh hình học
- usemtl : Tên vật liệu
- f : Mặt

(b) Định dạng tệp MTL

Để mô hình 3D trở lên sinh động, định dạng tệp MTL có nhiệm vụ lưu trữ các thông tin về màu sắc và kết cấu của mô hình để hỗ trợ cho tệp OBJ. Hai định dạng tệp này thường được sử dụng cùng nhau để có thể hiện thị một mô hình kết cấu nhiều màu.



Hình 3.7: Minh họa cách hoạt động của OBJ và MTL

Tệp MTL dựa trên mã ASCII mô tả các thuộc tính hình dạng bề mặt sẽ được áp dụng cho các mặt đa giác hoặc các mảng cong dạng tự do được xác định trong tệp OBJ. Tệp MTL được coi là một "thư viện" có thể chứa một hoặc nhiều định nghĩa vật liệu được đặt tên, mỗi định nghĩa có thể chỉ định các đặc điểm màu sắc, kết cấu và phản xạ.

Tệp MTL chứa một chuỗi các định nghĩa vật liệu, mỗi định nghĩa bắt đầu bằng từ khóa **newmtl** và tên cho vật liệu. Các câu lệnh trong định nghĩa vật liệu bao gồm một từ khóa, theo sau là các tùy chọn, giá trị hoặc tham chiếu cụ thể theo từ khóa đến các tệp bổ sung để sử dụng làm bản đồ kết cấu. Định nghĩa vật liệu kết thúc ở cuối tệp hoặc ở câu lệnh **newmtl** tiếp theo . Các câu xác định các đặc tính của vật liệu có thể theo bất kỳ thứ tự nào. Một ví dụ về định nghĩa vật liệu là:

```
newmtl Material
Ka 0.000000 0.000000 0.000000
Kd 0.640000 0.640000 0.640000
Ks 0.010000 0.010000 0.010000
Ns 56.862745
Ni 1.000000
d 1.000000
illum 2
map_Kd texture.jpg
```

Trong đó:

- Ka : chỉ định màu xung quanh , để giải thích cho ánh sáng phân tán trong toàn bộ cảnh bằng cách sử dụng các giá trị từ 0 đến 1 cho các thành phần RGB.
- Kd : chỉ định màu khuếch tán , thường đóng góp phần lớn màu cho một đối tượng. Trong ví dụ này, Kd đại diện cho màu xám, màu này sẽ được sửa đổi bởi bản đồ kết cấu màu được chỉ định trong câu lệnh map_Kd.
- Ks : chỉ định màu phản chiếu , màu được nhìn thấy khi bề mặt sáng bóng và giống như gương.
- Ns : xác định trọng tâm của các điểm nổi bật trong vật liệu. Giá trị Ns thường nằm trong khoảng từ 0 đến 1000, với giá trị cao dẫn đến vùng sáng tập trung, chặt chẽ.
- Ni : xác định mật độ quang học (còn gọi là chỉ số khúc xạ) trong vật liệu hiện tại. Các giá trị có thể nằm trong khoảng từ 0,001 đến 10. Giá trị 1,0 có nghĩa là ánh sáng không bị bẻ cong khi đi qua một vật thể.
- d : chỉ định một yếu tố để hòa tan , vật liệu này hòa tan vào nền bao nhiêu. Hệ số 1,0 hoàn toàn không rõ ràng. Hệ số 0,0 là hoàn toàn minh bạch.
- illum : chỉ định mô hình chiếu sáng , sử dụng giá trị số.
  - illum 0 : mô hình chiếu sáng màu không đổi, sử dụng Kd cho vật liệu
  - illum 1 : mô hình chiếu sáng khuếch tán sử dụng đến Ka, Kd, cường độ và vị trí của từng nguồn sáng và góc mà nó chiếu vào bề mặt.
  - illum 2 : mô hình chiếu sáng gương và khuếch tán sử dụng đến Ka, Kd, Ks, cường độ và vị trí của từng nguồn sáng và góc mà nó chiếu vào bề mặt.
- map_Kd : chỉ định tệp kết cấu màu sẽ được áp dụng cho hệ số phản xạ khuếch tán của vật liệu. Trong quá trình kết xuất, các giá trị map_Kd được nhân với các giá trị Kd để lấy các thành phần RGB.

Các định dạng OBJ và MTL được phần mềm chuyên dụng Blender hỗ trợ

nên có thể dễ dàng chọn xuất các tệp lưu thông tin mô hình dưới định dạng OBJ/MTL. Còn với việc đọc mô hình, Three.js hỗ trợ hai thư viện OBJLoader.js, MTLLoader.js, cụ thể như sau:

- Nếu chỉ muốn đọc hình học của mô hình thì chỉ cần sử dụng OBJLoader.
- Nếu muốn đọc mô hình có đầy đủ màu sắc và kết cấu thì sử dụng kết hợp cả hai là OBJLoader và MTLLoader.

Chương 4

XÂY DỰNG CHƯƠNG TRÌNH

Trong các chương trước, báo cáo đã giới thiệu về chi tiết các thư viện cũng như các framework để hỗ trợ trong việc thực hiện tạo ra các mô hình 3D. Từ đó, ta cũng có thể tự xây dựng được các mô hình 3D từ đơn giản đến phức tạp. Do đó, ở trong chương này, báo cáo sẽ trình bày các thành phần được xây dựng trong chương trình. Đồng thời, nêu lên công dụng, chức năng của ứng dụng đồ họa này và các đánh giá về nó.

4.1 Mô tả ứng dụng

4.1.1 Tổng quan

Ứng dụng này thuộc loại ứng dụng đồ họa 3D hỗ trợ học tập, được xây dựng trên nền tảng WebGL. Ứng dụng sử dụng công nghệ WebGL để hiển thị các đối tượng và mô hình 3D một cách trực quan và sinh động. Giao diện sử dụng các kỹ thuật đồ họa 3D để tạo ra các hiệu ứng ánh sáng, bóng đổ, texture và chuyển động đầy đủ. Người dùng có thể di chuyển, xoay và phóng to,thu nhỏ các đối tượng 3D để quan sát chi tiết.

Ứng dụng sẽ cung cấp các mô hình giải phẫu cơ thể người 3D, bao gồm các cơ quan, cơ, xương và các cấu trúc khác. Người dùng có thể di chuyển và xoay các mô hình này để hiểu rõ hơn về cấu trúc và tính chất của chúng. Ngoài ra,

ứng dụng còn có các mô hình về hình học cơ bản có thể điều chỉnh các thông số cơ bản như chiều dài, chiều rộng, bán kính,...

4.1.2 Công dụng

Công dụng của ứng dụng là giúp người học có thể tiếp cận và học tập về các khái niệm, cấu trúc và tính chất của cơ thể người một cách sinh động và trực quan. Các mô hình giải phẩu cơ thể người 3D và các tính năng tương tác, điều hướng và hiển thị dữ liệu của ứng dụng giúp người học hiểu rõ hơn về các cơ quan, cấu trúc và tính chất của cơ thể người.

Ứng dụng này có thể được sử dụng trong nhiều lĩnh vực khác nhau, bao gồm giáo dục đại học, y khoa, giáo dục phổ thông và các lĩnh vực liên quan đến giải phẩu cơ thể người. Người học có thể sử dụng ứng dụng này để quan sát chi tiết các mô hình 3D thực tế liên quan đến giải phẩu cơ thể người.

Một số ứng dụng cụ thể của ứng dụng này có thể bao gồm:

- Giúp hiểu rõ hơn về các cơ quan, hệ thống và cấu trúc của cơ thể người.
- Hỗ trợ việc học tập và giảng dạy trong các lĩnh vực y khoa, sinh học và dược học.
- Giúp người học chuẩn bị cho các kỳ thi và chứng chỉ trong các lĩnh vực liên quan đến giải phẩu cơ thể người.
- Giúp người học có thể tương tác với các nội dung học tập một cách sinh động và trực quan hơn.

Ngoài ra, các mô hình về hình học cơ bản có thể được sử dụng để quan sát trực quan hỗ trợ trong việc học các hình học không gian trong lĩnh vực toán học.

4.1.3 Một số tính năng

Tính năng	**Mô tả**
Lựa chọn mô hình	Cho phép người dùng lựa chọn mô hình phù hợp để sử dụng.
Thay đổi màu sắc	Cho phép người dùng thay đổi màu sắc của khung cảnh cũng như từng bộ phận của mô hình.
Hiển thị thông tin	Sử dụng chuột phải ấn chọn vào ví trí bộ phận cần xem thông tin để có thể hiển thị thông tin của chúng.
Ẩn hoặc hiện	Cho phép hiển thị hoặc ẩn đi áp dụng cho từng bộ phận của mô hình.
wireframe	Cho phép hiển thị mô hình ở dạng khung dây.
Xoay, di chuyển	Cho phép người dùng di chuyển mô hình đến vị trí mong muốn và có thể thực hiện xoay quanh nó.
Bề mặt	Cho phép người dùng lựa chọn hiển thị loại bề mặt là mặt ngoài, mặt trong hoặc cả hai mặt để có thể quan sát chi tiết từ ngoài vào trong của mô hình.
Thay đổi kích thước	Áp dụng cho các mô hình hình học cơ bản để có thể thay đổi các kích thước như chiều dài, chiều rộng, chiều cao hay bán kính hình cầu.
Thay đổi phân đoạn	Áp dụng cho các mô hình hình học cơ bản để thực hiện thay đổi phân đoạn của mô hình, có thể quan sát khi hiển thị dưới dạng khung dây.
Thay đổi góc	Áp dụng cho các mô hình có dạng hình tròn, hình cầu để cắt hoặc quan sát dưới góc nhìn khác nhau.

Bảng 4.1: Bảng về một số tính năng của ứng dụng.

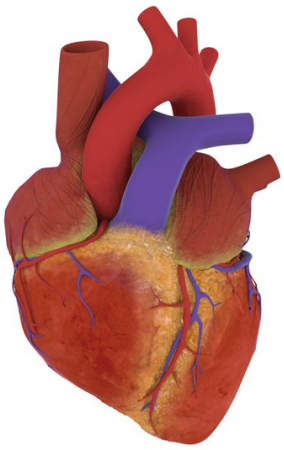
4.2 Các mô hình giải phẫu cơ thể người

Đây là các mô hình 3D phức tập với nhiều chi tiết mà rất khó để tự tạo ra với three.js nên thay vì tự tạo thì ta có thể thực hiện tìm các nguồn có sẵn của các mô hình 3D về lĩnh vực này và thực hiện đọc vào, chỉnh sửa cho phù hợp.

4.2.1 Giới thiệu về các mô hình

(a) Mô hình 3D về tim

Mô hình được lấy trên trang web **BLEND SWAP** [6], nó được đăng lên trang web này vào ngày 16/5/2014 do tài khoản có tên là daylanKifky.



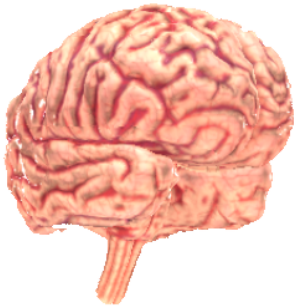
Hình 4.1: Mô hình 3D về trái tim con người

Đây là một mô hình bán thực tế của một trái tim con người. Nó được tạo ra cho mục đích mô phỏng để hiển thị rõ ràng các bộ phận khác nhau, đó là tâm thất, tâm nhĩ, động mạch chủ, tĩnh mạch, động mạch phổi và mạch vành.

Mô hình được tải về dưới định dạng tệp BLEND (.blend) nên nó sẽ được thực hiện chuyển về định dạng tệp OBJ/MTL thông qua phần mềm chuyên dụng Blender.

(b) Mô hình 3D về não

Mô hình được lấy trên trang web **Sketchfab** [7] với tiêu đề "Human brain, Cerebrum & Brainstem" do tài khoản có tên FrankJohansson đăng tài vào khoảng năm 2020.

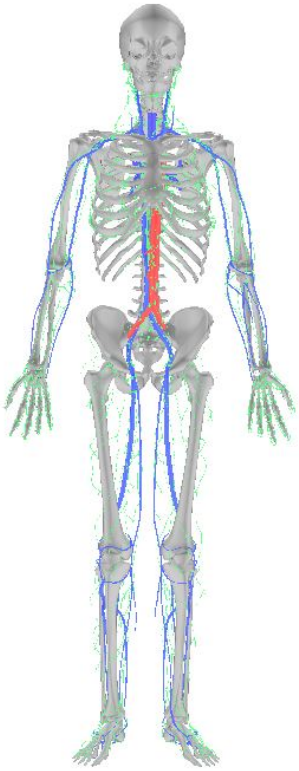


Hình 4.2: Mô hình 3D về bộ não con người

Mô hình được tải về dưới định dạng tệp DAE (.dae) nên nó sẽ được thực hiện chuyển về định dạng tệp OBJ/MTL thông qua phần mềm chuyên dụng Blender.

(c) Mô hình 3D về hệ thống bạch huyết

Mô hình được lấy trên trang web **Sketchfab** [7] với tiêu đề "Lymphatic System: an overview" do tài khoản có tên E-learning UMCG đăng tài vào khoảng năm 2018.



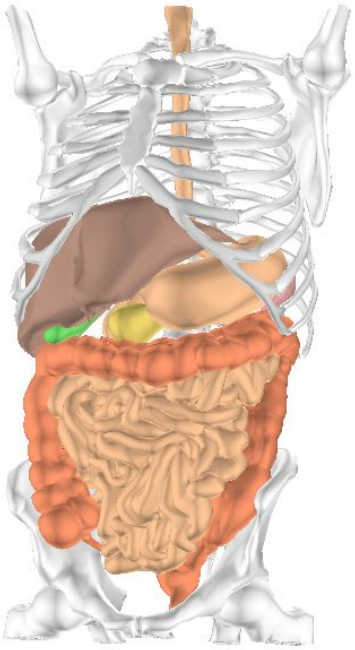
Hình 4.3: Mô hình 3D về phổi con người

Mô hình này cho thấy một cái nhìn tổng quan về hệ thống bạch huyết ở người. Ngoài các phần chính của hệ bạch huyết, nó cho thấy khí quản (màu trắng) và một phần của động mạch chủ (màu đỏ), các phần của hệ thống tĩnh mạch (màu xanh nước biển) và bộ xương (màu xám). Các bộ phận của động mạch chủ và hệ thống tĩnh mạch đã được thêm vào vì các bạch huyết có xu hướng đi theo các tĩnh mạch ở chi và các động mạch trong thân.

Mô hình được tải về dưới định dạng tệp FBX (.fbx) nên nó sẽ được thực hiện chuyển về định dạng tệp OBJ/MTL thông qua phần mềm chuyên dụng Blender.

(d) Mô hình 3D về cơ quan của đường tiêu hóa

Mô hình được lấy trên trang web **Sketchfab** [7] với tiêu đề "Gastrointestinal tract" do tài khoản có tên E-learning UMCG đăng tài vào khoảng năm 2018.



Hình 4.4: Mô hình 3D về cơ quan của đường tiêu hóa

Mô hình được tải về dưới định dạng tệp FBX (.fbx) nên nó sẽ được thực hiện chuyển về định dạng tệp OBJ/MTL thông qua phần mềm chuyên dụng Blender.

(e) Mô hình 3D về phổi

Mô hình được lấy trên trang web **Sketchfab** [7] với tiêu đề "Anatomy of the airways (opaque lungs)" do tài khoản có tên E-learning UMCG đăng tài vào khoảng năm 2018.

Hình 4.5: Mô hình 3D về phổi con người

Mô hình được tải về dưới định dạng tệp FBX (.fbx) nên nó sẽ được thực hiện chuyển về định dạng tệp OBJ/MTL thông qua phần mềm chuyên dụng Blender.

4.2.2 Thực hiện đọc và tinh chỉnh mô hình 3D

Sau khi thực hiện tìm kiếm và tải các mô hình về, chúng đã được chuyển về định dạng tệp OBJ/MTL. Khi đó, ta có thể thực hiện đọc các mô hình dưới sự trợ giúp của thư viện Three.js và trực tiếp chỉnh sửa trên đó.

Đầu tiên, sử dụng kết hợp OBJLoader và MTLLoader do Three.js cung cấp để có thể đọc hai định dạng tệp mà ta có với lệnh load().

Với mỗi lệnh load(), ta sẽ tạo ra một hàm có chức năng đọc và chỉnh sửa mô hình. Thực chất, các mô hình được sử dụng ở đây đều một nhóm chứa nhiều lưới nên có thể chỉnh sửa từng lưới ngoài việc chỉnh sửa chung cả mô hình. Khi chỉnh sửa từng lưới, cần gọi đúng tên lưới cần sửa với getObjectByName("Tên lưới") và đưa ra thứ muốn chỉnh sửa ví dụ như màu sắc vật liệu.

Trong Three.js, mỗi lưới trên được hiểu là một children trong nhóm. Do đó, ta có thể tạo ra các lưới rỗng và thực hiện sao chép các children vào các lưới rỗng này. Bước sao chép này sẽ được thực hiện trong hàm ta tạo ra ở lệnh load(). Mục đích của việc này là các lưới rỗng sau khi được sao chép sẽ có thể chỉnh sửa được mô hình ở bất cứ đâu mà không cần phải thực hiện trong lệnh load().

4.2.3 Tạo các chú thích trên mô hình 3D

Để tạo ra các chú thích trên mô hình có thể ẩn hoặc hiện dựa vào việc sử dụng chuột, đầu tiên ta sẽ khởi tạo một số đối tượng như sau:

- Một đối tượng THREE.Raycaster. Lớp Raycaster này được thiết kế để hỗ trợ raycasting. Raycasting được sử dụng để chọn chuột (tìm ra đối tượng nào trong không gian 3D mà chuột đi qua) trong số những thứ khác.
- Một vector 2 chiều để xác định tọa độ của chuột trong tọa độ thiết bị được chuẩn hóa (NDC). Vector này sẽ kết hợp với tọa độ máy ảnh để thực hiện lệnh setFromCamera() để cập nhật tia với điểm gốc và hướng mới.
- Tạo ra bảng(popup) để hiển thị văn bản với các cài đặt về phông chữ, màu nền được khai báo trong thẻ <style>...</style> ở đầu chương trình. Ngoài ra, ta sẽ phải khai báo và xác định vị trí của bảng bằng các vector 3 chiều.

Sau khi tạo ra các bảng điển hiển thị văn bản, thực hiện viết các hàm để tính toán vị trí đặt bảng và cập nhật vị trí đó khi sử dụng các điều khiển màn hình.

Cuối cùng, từ các đối tượng và hàm được tạo trên ta sẽ xác định được một hàm để thực hiện việc hiển thị văn bản trên các thành phần của mô hình với chức năng là khi ấn chuột phải vào thành phần nào thì sẽ hiển thị bảng tên của nó và có thể thay đổi vị trí khi xoay hoặc di chuyển mô hình.

4.3 Cài đặt các thành phần khác

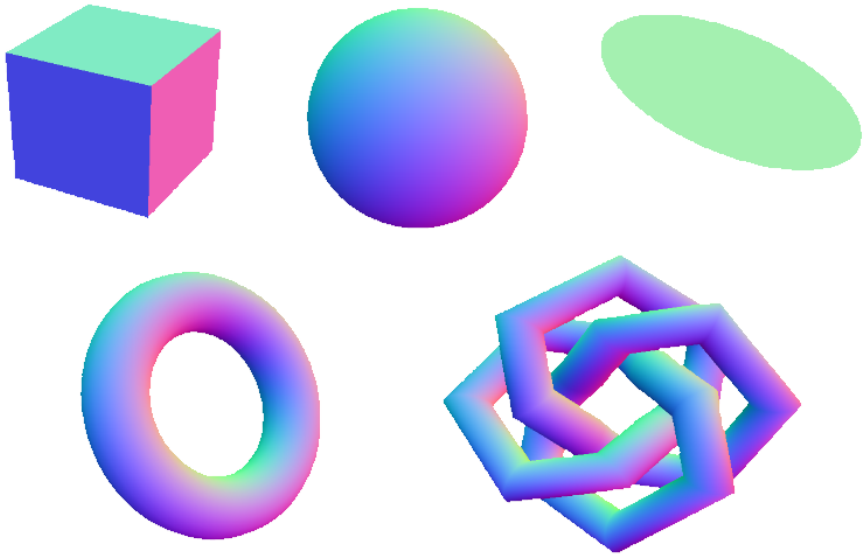
4.3.1 Một số thành phần cần thiết

- Tạo một khung cảnh bằng câu lệnh THREE.Scene()
- Sử dụng một máy ảnh loại PerspectiveCamera với các thông số như fov, aspect, near, far...
- Tạo một trình kết xuất cơ bản WebGLRenderer.

- Sử dụng một số đèn chiếu sáng PointLight đặt ở các ví trí khác nhau xung quanh khung cảnh với các thông số như color, intensity, distance,...

4.3.2 Các mô hình hình học cơ bản

Với các mô hình hình học cơ bản, ta có thể dễ dàng thực hiện xây dựng bởi thư viện Three.js cung cấp các thư viện con để hỗ trợ tạo ra chúng. Dưới đây là một số mô hình hình học được tạo ra trong chương trình.



Hình 4.6: Một số mô hình hình học cơ bản

Yếu tố hình học của các hình trên được biểu diễn qua Three.js tương ứng với mỗi hình đó, cụ thể:

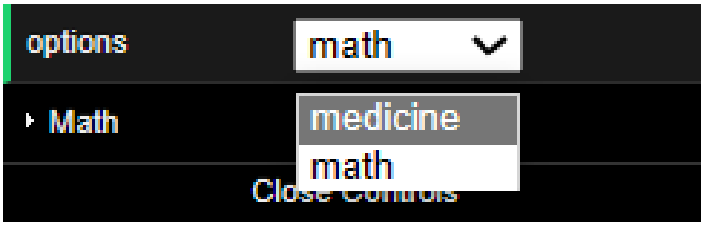
- Hình lập phương: BoxGeometry
- Hình cầu: SphereGeometry
- Hình tròn: CircleGeometry
- Hình xuyến: TorusGeometry
- Nút hình xuyến: TorusKnotGeometry

Ở hình trên, các mô hình hình học này đều đang được sử dụng vật liệu

MeshNormalMaterial. Ta cũng có thể dễ dàng xây dựng một chức năng điều khiển thay đổi vật liệu của chúng. Do đó, trong chương trình được xây dựng ở báo cáo này, trước hết là sẽ thực hiện tạo ra một chức năng thay đổi vật liệu về dạng MeshLambertMaterial. Và dễ dàng thực hiện tương tự với các dạng vật liệu còn lại.

4.3.3 Xây dựng bảng điều khiển

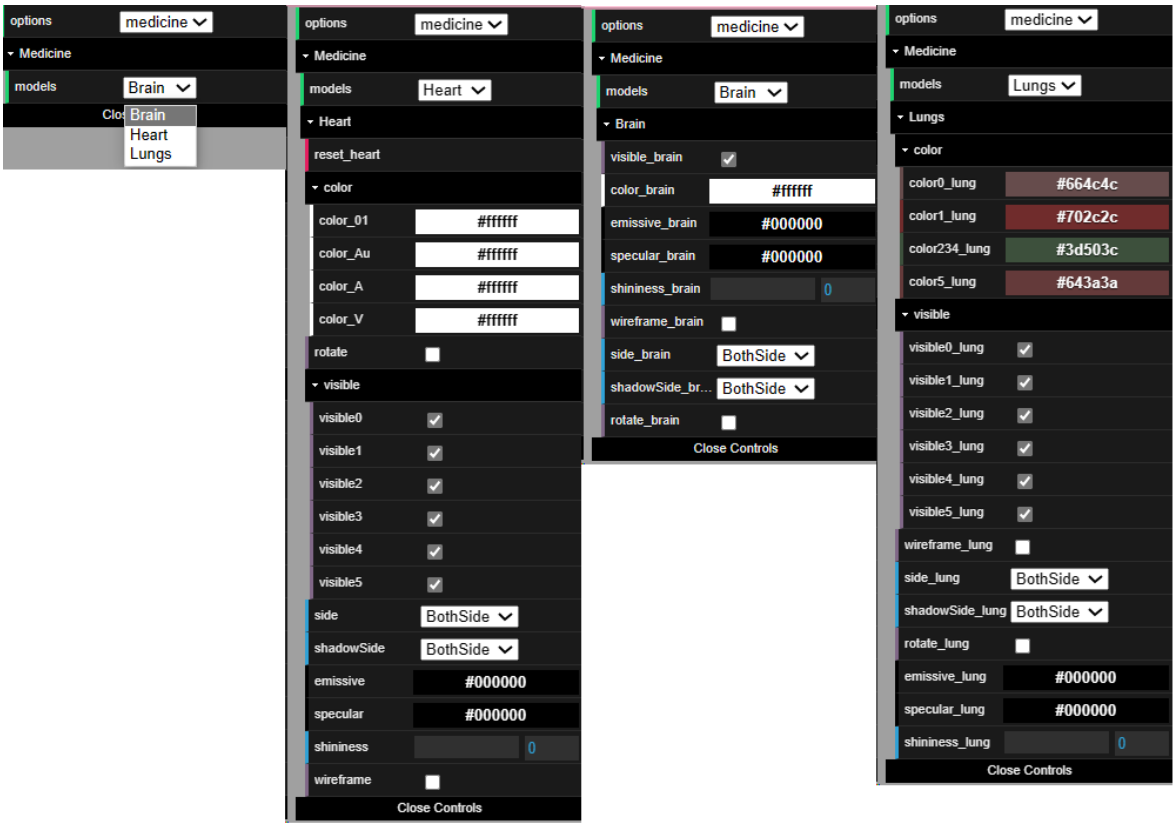
Sau khi có được các mô hình trên, ta thực hiện xây dựng một bảng điều khiển để có thể chọn ra mô hình muốn hiển thị. Tại mỗi một mô hình sẽ có các chức năng riêng để thay đổi thông số của mô hình đó. Do các mô hình được phân làm hai loại là các mô hình về mặt y học và các mô hình hình học cơ bản nên đầu tiên sẽ cần tạo ra một sự lựa chọn giữa hai loại này.



Hình 4.7: Chức năng lựa chọn loại mô hình trên bảng điều khiển.

Với lựa chọn là các mô hình về y học thì ta lại tiếp tục lựa chọn tương ứng với ba mô hình tim, não, phổi đã giới thiệu ở trên. Trong mỗi sự lựa chọn này thì sẽ có một gói chức năng riêng được hiển thị, ở đó nó cho phép người dùng thao tác chỉnh sửa trực tiếp các mô hình. Có một vài chức năng thường được sử dụng như màu sắc, ẩn hoặc hiện, quay, wireframe,...

Cũng tương tự như sự lựa chọn trên, khi chọn các mô hình hình học (thuộc về lĩnh vực toán học) thì cũng có tiếp sự lựa chọn trong năm mô hình đã tạo. Và với mỗi mô hình cũng có các chức năng thay đổi riêng thuộc vào từng gói, được hiển thị khi mô hình tương ứng được lựa chọn. Các mô hình hình học này, ta có thể thay đổi kích thước của chúng như chiều dài, chiều rộng, chiều cao hay màu sắc, wireframe,...



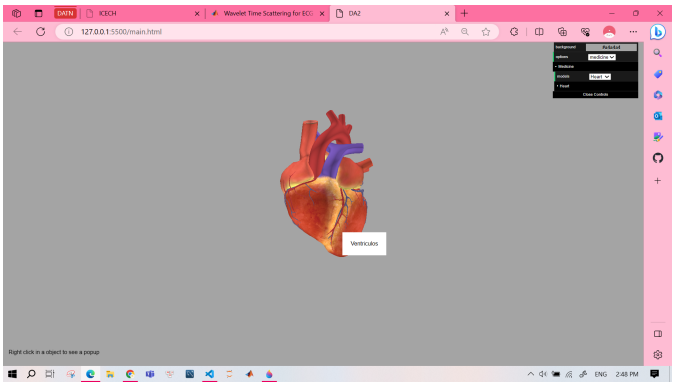
Hình 4.8: Bảng điều khiển của các mô hình về y học

Ngoài ra, còn có thể thay đổi cả vật liệu, như Hình 4.6 các mô hình đang được sử dụng MeshNormalMaterial. Vật liệu này có thể làm cho một số chức năng không thể hoạt động như thay đổi màu sắc nên cần xây dựng một chức năng có thể thay đổi vật liệu.

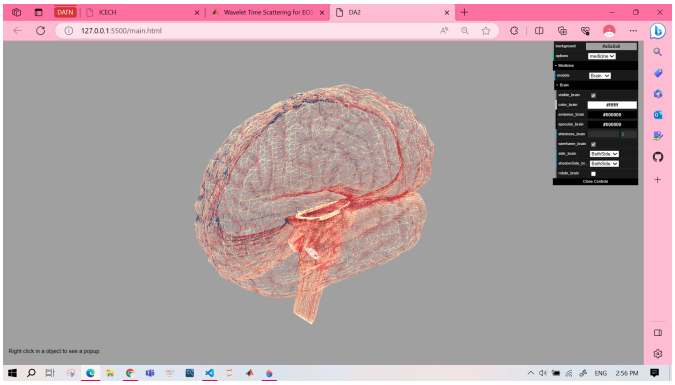
Hình 4.9: Bảng điều khiển của các mô hình hình học cơ bản

4.4 Đánh giá

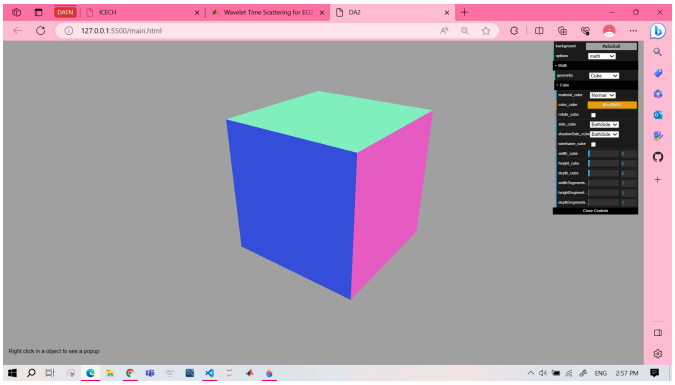
Đây là một chương trình cho phép hiển thị các mô hình 3D để có thể quan sát các chi tiết của nó. Ngoài ra, ứng dụng còn cho phép chỉnh sửa trực tiếp trên các mô hình này tùy vào mong muốn của người dùng. Dưới đây là một số hình minh họa.



Hình 4.10: Hình minh họa chương trình về mô hình tim



Hình 4.11: Hình minh họa chương trình về mô hình não



Hình 4.12: Hình minh họa chương trình về mô hình hình lập phương

Ứng dụng đã đáp ứng được mục tiêu đề ra của đề tài, tuy nhiên vẫn còn nhiều hạn chế. Ở đây, nó mới chỉ cung cấp cho người dùng một số tính năng hữu ích để học tập về giải phẫu cơ thể và hình học không gian thông qua mô hình 3D. Điều này, đòi hỏi người dùng cần có kiến thức chuyên môn hoặc đang tìm hiểu về lĩnh vực này. Do đó, việc mở rộng ra các lĩnh vực khác cũng là một việc cần thiết để phát triển đề tài này.

Kết luận

Báo cáo đề tài "**Phát triển ứng dụng đồ họa hỗ trợ học tập**" đã giới thiệu sơ lược về các thư viện đồ họa và đi vào xây dựng chương trình ứng dụng đồ họa cho phép hiển thị và tương tác với các mô hình 3D thông qua thư viện 3D JavaScript Three.js. Qua quá trình nghiên cứu, tìm hiểu và thực hiện báo cáo, em đã thấy mình đạt được những kết quả:

- Nắm được kiến thức sơ lược về thư viện đồ họa lập trình 3D.
- Nắm được cách làm việc với các mô hình 3D.
- Xây dựng chương trình ứng dụng đồ họa để hỗ trợ trong việc học tập.

Tuy nhiên, do hạn chế về thời gian và kiến thức, báo cáo vẫn tồn tại những thiếu sót và có thể phát triển thêm:

- Thiết kế giao diện đẹp mắt và linh động hơn.
- Thêm tính năng đọc các mô hình trực tiếp trên trang web.
- Thêm mô hình của các lĩnh vực khác.
- Thêm thông tin và tạo ra các câu hỏi hoặc trò chơi tương tác giữa các bộ phận của mô hình.

Qua đây, em xin gửi lời cảm ơn sâu sắc tới cô Vương Mai Phương, giảng viên Trường Đại học Bách Khoa Hà nội, đã tận tình hướng dẫn cho em để em có thể hoàn thành báo cáo này.

Em xin chân thành cảm ơn!

Tài liệu tham khảo

[1] Jos Dirksen, *Learning Three.js: The JavaScript 3D Library for WebGL*, Packt Publishing, 2013.

[2] Trang web của Khronos Group https://www.khronos.org/webgl/, truy cập lần cuối vào hồi 21 giờ ngày 20/6/2023.

[3] WEBGL - Chuẩn Mới Cho Đồ Họa 3D Trên Web. URL: https://codelearn.io/sharing/tim-hieu-webgl-cong-nghe-do-hoa-3d, truy cập lần cuối vào hồi 15 giờ ngày 19/7/2023.

[4] What Are Polygons in 3D Modeling. URL: https://cgifurniture.com/what-are-polygons-in-3d-modeling/, truy cập lần cuối vào hồi 23 giờ ngày 5/7/2023.

[5] Trang web FILEFORMAT. URL: https://docs.fileformat.com/3d/obj/, truy cập lần cuối vào hồi 22 giờ ngày 8/7/2023.

[6] Trang web BLEND SWAP https://www.blendswap.com/blend/12579, truy cập lần cuối vào hồi 9 giờ ngày 10/7/2023.

[7] Trang web Sketchfab https://sketchfab.com/3d-models, truy cập vào 21 giờ ngày 10/7/2023.