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

PHÁT TRIỂN ỨNG DỤNG ĐỒ HOẠ MÁY TÍNH TRONG LĨNH VỰC Y TẾ

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

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 23 tháng 2 năm 2023

Giảng viên hướng dẫn

Lời cảm ơn

Em xin trân trọng cảm ơn TS. Vương Mai Phương đã giúp em hoàn thiện đồ án này.

Hà Nội, tháng 2 năm 2023

Tác giả đồ án

Phạm Thu Trang

Tóm tắt nội dung Đồ án

1. .

2. .

3. .

Hà Nội, tháng 2 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 máy tính trong lĩnh vực y tế"**. (tạm thời)

Đề tài được xây dựng 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 về thư viện đồ họa web - WebGL.** Chương này giới thiệu cơ bản về thư viện đồ họa web - WebGL.
2. **Chương 2: Thư viện 3D JavaScript dành cho WebGL - Three.js.** Chương này giới thiệu tổng quan về thư viện Three.js, là thư viện 3D JavaScript dành cho WebGL
3. **Chương 3: Làm việc với mô hình 3D** ... Cập nhật
4. **Chương 4: Xây dựng chương trình và kết quả.** Chương này trình bày các bước xây dựng chương trình và kết quả thu được.

Chương 1

Giới thiệu về thư viện đồ họa web - WebGL

Để tạo nội dung truyền thông tương tác cho web theo 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. Trong chương này, báo cáo sẽ trình bày một cách tổng quan về thư viện đồ họa web này.

1.1 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.

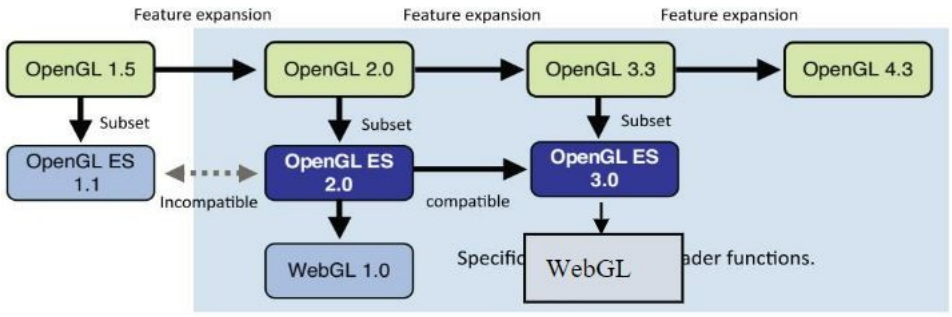
WebGL sử dụng khả năng tăng tốc phần cứng cho phép các nhà phát triển có khả năng xây dựng đồ họa tương tác hiển thị thời gian thực hiệu suất cao trong trình duyệt web, chẳng hạn như trò chơi, mô phỏng, trực quan hóa dữ liệu, video hoạt hình, mô hình 3D, hoạt ảnh phân tử,...

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

[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

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

1.2 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.

1.3 OpenGL Shader Language - GLSL

GLSL là một ngôn ngữ đổ bóng cấp cao với cú pháp dựa trên ngôn ngữ lập trình C. Nó được tạo ra bởi OpenGL ARB để cung cấp cho các nhà phát triển quyền kiểm soát trực tiếp hơn đối với quá trình đồ họa mà không cần phải sử dụng hợp ngữ ARB hoặc các ngôn ngữ dành riêng cho phần cứng.

1.3.1 Trình đổ bóng đỉnh - Vertex Shader

Trình đổ bóng đỉnh thao tác tọa độ trong không gian 3D và được gọi một lần trên mỗi đỉnh. Mục đích là để biến đổi hình học từ một nơi này sang nơi khác. Nó xử lý dữ liệu của mỗi đỉnh như tọa độ, màu sắc, cấu trúc bề mặt (texture) với các nhiệm vụ là :

- Biến đổi các đỉnh.
- Áp dụng màu sắc.
- Tạo ánh sáng.
- Sinh ra cấu trúc bề mặt mới.
- Biến đổi cấu trúc bề mặt.

1.3.2 Trình đổ bóng phân đoạn - Fragment Shader

Một lưới đồ họa được ghép bởi nhiều hình tam giác, và bề mặt của mỗi một tam giác được hiểu là một mảnh (fragment). Trình đổ bóng phân đoạn xác định các màu RGBA (đỏ, lục, lam, alpha) cho mỗi pixel đang được xử lý — một trình

đổ bóng phân đoạn duy nhất được gọi một lần cho mỗi pixel. Mục đích là để tính toán và tạo màu cho các điểm của tất cả các mảnh. Nhiệm vụ của nó là:

- Truy cập cấu trúc bề mặt.
- Áp dụng lên cấu trúc bề mặt.
- Tạo độ mờ.
- Tính toán màu sắc.

1.4 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:

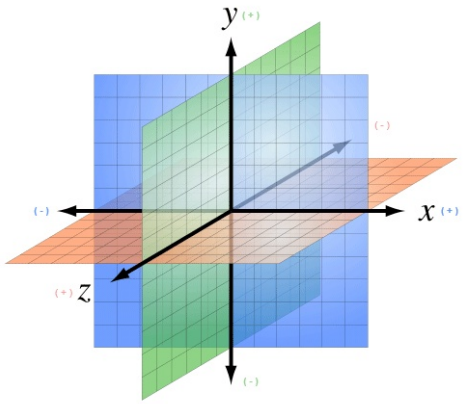
- id : định danh cho canvas
- width, height : độ rộng, độ cao

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.

1.5 Một số thành phần cơ bản của WebGL

1.5.1 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

1.5.2 Các nền tảng của WebGL

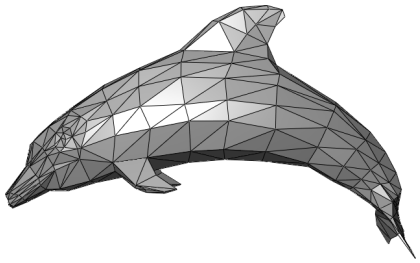
- Đỉnh (Vertices): là một điểm được giao bởi các cạnh của một đối tượng 3D, được biểu diễn bởi 3 giá trị số thực tương ứng với mỗi trục x, y, z. Trong WebGL các đỉnh được lưu trữ bằng mảng javascript.
- Các chỉ số (Indices): là các số nguyên dương dùng để định danh các đỉnh. Các chỉ số được sử dụng để 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.

1.5.3 Lưới - Mesh

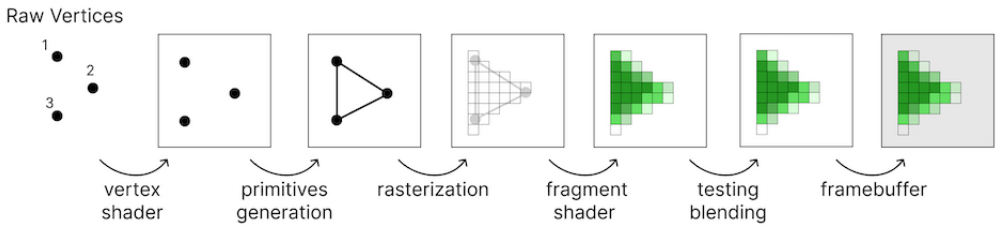
Cho đến ngày nay, tuy có nhiều cách để vẽ đồ họa 3D nhưng cách phổ biến nhất vẫn là sử dụng một lưới . Lưới là một đối tượng bao gồm một hoặc nhiều hình đa giác, được xây dựng từ đỉnh (x , y , z) xác định vị trí tọa độ trong không gian 3D. Các đa giác được sử dụng phổ biến nhất trong mắt lưới là hình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

1.6 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

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), v.v.
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 3D JavaScript dành cho WebGL - 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 Thư viện 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

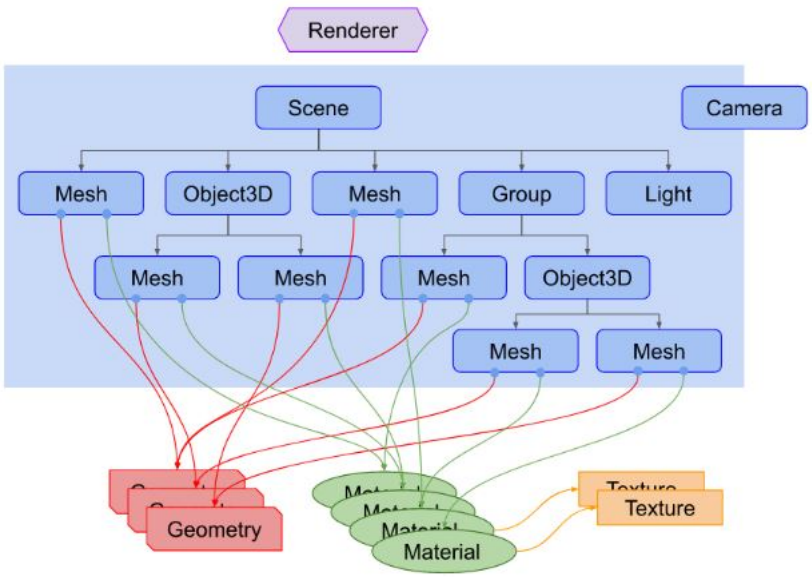
động và tương tác, thậm chí tạo trò chơi với một số logic. Và với việc sử dụng nó, ta có thể làm hầu hết mọi thứ mà ta có thể tưởng tượng.

2.2 Cấu trúc ứng dụng Three.js

2.2.1 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.

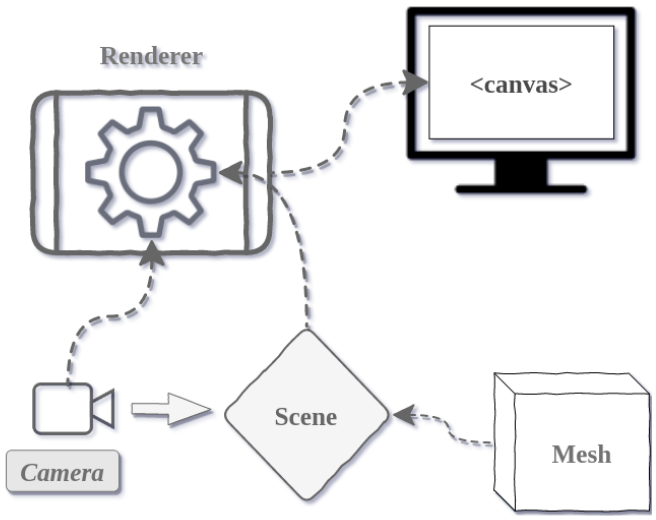
2.2.2 Mô hình cấu trúc



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

2.3 Một số thành phần cơ bản của Three.js

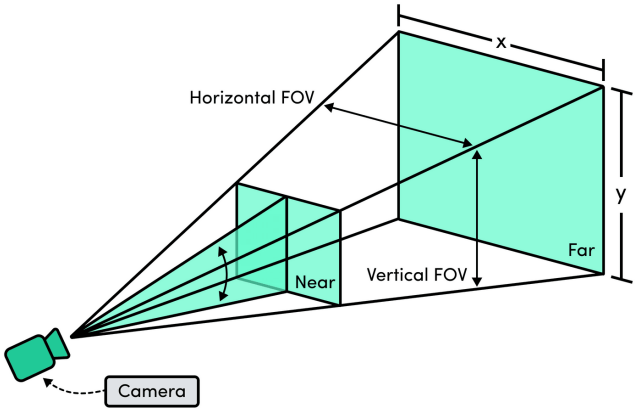
2.3.1 Máy ảnh

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.



Hình 2.3: PerspectiveCamera

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

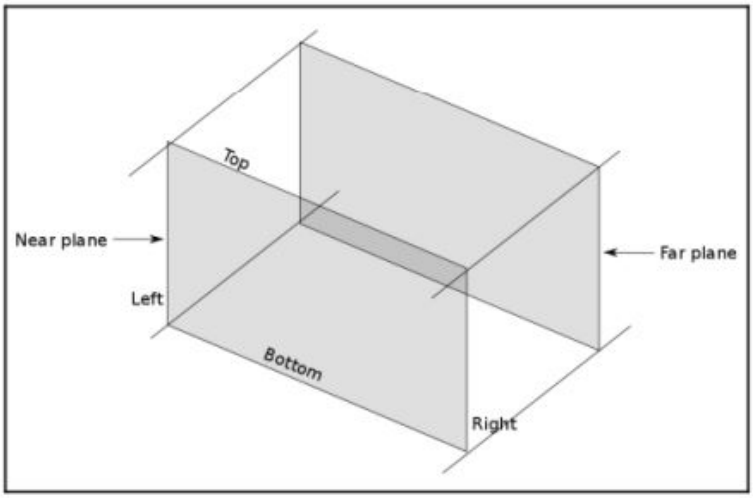
Thuộc tính	**Giá trị**	**Ý nghĩa**
fov	0 - 360	Góc quan sát, mô phỏng độ rộng của góc nhìn, tính bằng độ. Giá trị phù hợp với mắt người là dưới 180, thường dùng 45-50.
aspect	số thực dương	Tỉ lệ của vùng quan sát: chiều rộng/chiều cao
near, far	Số thực dương, $0 < \text{near} < \text{far}$	Khoảng cách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

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.

2.3.2 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);

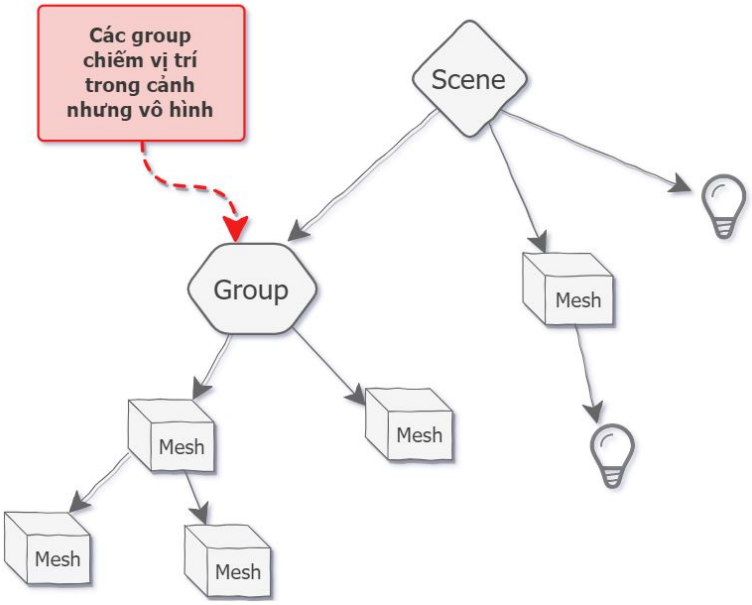
- antialias: chỉ định có thực hiện xử lý antialiasing hay không, mặc định là false. Nếu để antialias bằng true thì các đường thẳng sẽ sắc nét hơn, không bị trông giống bậc thang hoặc răng cưa.

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)

2.3.3 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

2.3.4 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 - Mesh

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 kết cấu, 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 Mesh(geometry, material);

(b) Hình học - geometry

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) Kết cấu - material

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 vàhệ số phản chiếu.
MeshStandardMaterial	Material này sử dụng physically based rendering. Một model 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 reflection.

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.

(d) Vật liệu - texture

(e) Đèn - Light

2.4 Thư viện dat.GUI

Chương 3

Làm việc với mô hình 3D

3.1 Mô hình dạng .obj và .mtl

3.2 Đọc mô hình 3D

3.3 Thay đổi các định dạng

Chương 4

Xây dựng chương trình và kết quả

4.1 Giới thiệu và thực hiện tải mô hình 3D

4.2 Sử dụng texture cho mô hình 3D

4.3 Thực hiện xây dựng controls

4.4 Xây dựng các mô hình hình học cơ bản

4.5 Tạo các chú thích trên mô hình 3D

Kết luận

Đồ án đã đạt được mục tiêu đề ra

1. ...
2. ...

Kết quả của đồ án

1. ...
2. ...

Hướng phát triển của đồ án trong tương lai

1. Xây dựng thêm chức năng đánh giá đồ án.
2. Hỗ trợ giám sát tiến độ đồ án của sinh viên.
3. Cải thiện giao diện đẹp mắt và thân thiện với người dùng hơn.

Tài liệu tham khảo

Giáo trình

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

Trang web

[2] Trang web của Khronos Group https://www.khronos.org/webgl/, truy cập lần cuối vào hồi ...