|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  BÁO CÁO BÀI TẬP 1  **ĐA PHƯƠNG TIỆN NÂNG CAO**  **Chủ đề: Xây dựng đối tượng đồ họa với công cụ Filament**     |  |  |  | | --- | --- | --- | | Giảng viên hướng dẫn: | TS. Phạm Văn Tiến | | | Nhóm sinh viên: | Nguyễn Quốc An | 20160015 | |  | Nguyễn Đặng Sơn Lam | 20164839 | |  | Nguyễn Thị Lan Anh | 20160162 |     Hà Nội, 10-2020 |

# LỜI NÓI ĐẦU

Một trong những yêu cầu tất yếu với các thiết bị công nghệ hiện đại là vấn đề đồ hoạ. Một ứng dụng được ưa chuộng phải đi kèm với đồ hoạ hấp dẫn. Bài tập “Xây dựng đối tượng đồ hoạ với công cụ Filament” đem đến cho nhóm trải nghiệm thực hành để tạo ra đồ hoạ cơ bản. Các yêu cầu cụ thể từ thầy giúp nhóm định hình được các bước cần triển khai hay từng mục tiêu nhỏ cần đạt được. Qua đó, việc thực hành bài bản và dễ dàng tiếp cận hơn. Nhóm đã thực hiện được các yêu cầu đặt ra, tuy nhiên, để đạt đến mức hoàn hảo thì rất khó. Nguyên nhân chính là còn nhiều thiếu sót trong kiến thức, hiểu biết của các thành viên. Chúng em mong muốn được tham khảo, học hỏi thêm từ các nhóm khác trong lớp và đánh giá của thầy để có thể làm tốt hơn trong tương lai.

Chúng em xin cảm ơn thầy vì đã vô cùng nhiệt huyết, tận tâm trong những giờ trên giảng đường cũng như qua công cụ học online Microsoft Teams.

**MỤC LỤC**

[LỜI NÓI ĐẦU 2](#_Toc54813672)

[DANH MỤC HÌNH VẼ 4](#_Toc54813673)

[DANH MỤC BẢNG BIỂU 5](#_Toc54813674)

[CHƯƠNG 1. TỔNG QUAN VỀ FILAMENT 6](#_Toc54813675)

[1. Khái niệm 6](#_Toc54813676)

[1.1 Nguyên tắc 6](#_Toc54813677)

[1.2 Kết xuất thực thể 6](#_Toc54813678)

[2. Material 7](#_Toc54813679)

[3. Lighting 8](#_Toc54813680)

[CHƯƠNG 2. THỰC HÀNH 10](#_Toc54813681)

[1. Môi trường 10](#_Toc54813682)

[2. Giới thiệu chung 10](#_Toc54813683)

[2.1. Các thành phần chính của project 10](#_Toc54813684)

[2.2 Chạy project 11](#_Toc54813685)

[3. Thiết lập các thành phần cần thiết 11](#_Toc54813686)

[3.1. Tạo lâp tệp HTML 11](#_Toc54813687)

[3.2 Tạo lập tệp Javascript điều khiển vật thể 12](#_Toc54813688)

[3.3 Tạo lập tệp đối tượng filamesh. 13](#_Toc54813689)

[3.4 Vật liệu 14](#_Toc54813690)

[3.5 Ánh sáng 16](#_Toc54813691)

[3.6 Camera 17](#_Toc54813692)

[CHƯƠNG 3: KẾT QUẢ 18](#_Toc54813693)

[DANH MỤC THAM KHẢO 19](#_Toc54813694)

# DANH MỤC HÌNH VẼ

[Hình 1. Đối tượng đồ họa với vật liệu texturedlit.filamat 18](#_Toc54814823)

[Hình 2. Đối tượng đồ họa với vật liệu aidefault.mat 18](#_Toc54814824)

# DANH MỤC BẢNG BIỂU

[Bảng 1. Một số đặc tính của material 7](#_Toc54815382)

[Bảng 2. Các loại ánh sáng và đơn vị 9](#_Toc54815383)

# CHƯƠNG 1. TỔNG QUAN VỀ FILAMENT

## 1. Khái niệm

Filament là một công cụ hiển thị thực thể (Physically based rendering- PBR) cho Android. Mục tiêu của Filament là cung cấp một bộ công cụ và APIs cho các nhà phát triển Android, cho phép họ tạo ra kết xuất 2D và 3D chất lượng cao một cách dễ dàng.

Đề cập đến các mô hình vật liệu và ánh sáng được sử dụng trong Filament.

### 1.1 Nguyên tắc

Kết xuất thời gian thực là một lĩnh vực nghiên cứu tích cực và có số lượng lớn các phương trình, thuật toán và cách triển khai để lựa chọn cho mọi tính năng đơn lẻ. Trước hết phải xác định mục tiêu hoặc các nguyên tắc để tuân theo và quyết định thuật toán.

**Hiệu suất thời gian thực**

Mục tiêu chính là thiết kế và triển khai trên nền tảng di động, cụ thể là GPU lớp OpenGL ES 3.x.

**Chất lượng**

Hệ thống kết xuất nhấn mạnh chất lượng hình ảnh tổng thể. Tuy nhiên có thể thoả hiệp để đáp ứng, hỗ trợ GPU hiệu suất thấp và trung bình.

**Dễ sử dụng**

Cần có khả năng lặp lại thường xuyên và nhanh chóng, trực quan.

**Tính linh hoạt**

Phương pháp tiếp cận thực thể loại bỏ các kết xuất không thực tế. Ví dụ giao diện người dùng sẽ cần vật liệu không được chiếu sáng.

**Kích thước**

Mong muốn thư viện file có kích thước càng nhỏ càng tốt để bất kì ứng dụng nào cũng có thể sử dụng nó.

### 1.2 Kết xuất thực thể

Filament áp dụng PBR vì những lợi ích của nó trên quan điểm hiệu quả về nghệ thuật và sản xuất. Phương pháp kết xuất dựa trên thực thể cung cấp sự thể hiện chính xác hơn của vật liệu và cách chúng tương tác với ánh sáng khi so sánh với các mô hình thời gian thực truyền thống.

Việc tách biệt vật liệu và ánh sáng ở cốt lõi của phương pháp PBR giúp dễ dàng tạo ra các nội dung thực tế trông chính xác trong mọi điều kiện ánh sáng.

## 2. Material

Material có thể sử dụng một trong các mô hình vật liệu như:

* Lit (tiêu chuẩn)
* Subsurface
* Cloth
* Unlit
* Specular glossiness (legacy)

Đối với mô hình vật liệu khác nhau sẽ cần sử dụng đến các đặc tính khác nhau. Tổng hợp một số đặc tính của material được sử dụng như trong Bảng 1.

Bảng 1. Một số đặc tính của material

|  |  |
| --- | --- |
| **Đặc tính** | **Ý nghĩa** |
| baseColor | Áp dụng cho bề mặt phi kim loại và kim loại |
| Metalic | Bề mặt là chất điện môi (0,0) hay chất dẫn điện (1,0). Thường được sử dụng dưới dạng giá trị nhị phân (0 hoặc 1) |
| Roughness (Độ nhám) | Độ mịn cảm nhận (1,0) hoặc nhám (0,0) của bề mặt. Bề mặt nhẵn thể hiện phản xạ sắc nét. |
| Reflectance | Phản xạ Fresnel ở góc tới bình thường đối với bề mặt điện môi, ảnh hưởng bởi cường độ phản xạ. |
| clearCoat | Độ bền của lớp ngoài |
| clearCoatRoughness | Cảm nhận độ mịn, độ nhám của lớp ngoài |
| Anisotropy (dị hướng) | Số lượng dị hướng hoặc theo hướng tiếp tuyến, hoặc theo hướng bit. |
| Normal | Chỉnh sửa bề mặt sử dụng bản đồ. |
| bentNormal | Sử dụng để cải thiện chất lượng chiếu sáng gián tiếp |
| postLightingColor | Màu bổ sung có thể pha trộn với kết quả của tính toán ánh sáng |
| ior | Chỉ số khúc xạ đối với vật khúc xạ |
| transmission | Xác định mức độ ánh sáng khuyếch tán của chất điện môi truyền qua vật thể, nói cách khác là độ trong suốt của vật thể. |
| absorption | Hệ số hấp thụ đối với vật bức xạ |
| thickness | Bề dày của vật khúc xạ |

Material package là một tệp nhị phân có chứa trình tạo bóng và các bit dữ liệu khác xác định vật liệu PBR, mô tả tất cả thông tin theo yêu cầu của vật liệu:

* Tên
* Thành phần sử dụng
* Material model
* Thuộc tính bắt buộc
* Biến
* Trạng thái Raster (chế độ trộn,..)
* Shader Code (độ bóng phân đoạn, tuỳ chọn độ bóng…)

**Ánh xạ với ảnh (mipmapped textures)** là việc sử dụng những tham số có thể được ánh xạ với một bức ảnh có sẵn. Ví dụ như bài thực hành của nhóm, cần có 05 ảnh đề ánh xạ 05 tham số tới 5 ảnh đó, đặc trưng cho các tính chất về màu sắc, độ nhám, độ kim loại, độ bóng,..

Để thực hiệu quá trình ánh xạ, trước hết ta cần 05 bức ảnh mẫu thể hiện tính chất vật liệu. Nhà cung cấp có cung cấp một công cụ để ta có chuyển ảnh từ dạng thông thường (.png) sang một kiểu dữ liệu binary có phần mở rộng là .ktx. Các bước thực hiện và bố cục sẽ được trình bày ở chương 2.

Ví dụ về tạo material như sau:

|  |
| --- |
| material {  name : Lit,  shadingModel : lit,  parameters : [  { type : float3, name : baseColor },  { type : float, name : roughness },  { type : float, name : clearCoat },  { type : float, name : clearCoatRoughness }  ],  }  fragment {  void material(inout MaterialInputs material) {  prepareMaterial(material);  material.baseColor.rgb = materialParams.baseColor;  material.roughness = materialParams.roughness;  material.clearCoat = materialParams.clearCoat;  material.clearCoatRoughness = materialParams.clearCoatRoughness;  }  } |

## 3. Lighting

Sự chính xác và mạch lạc của môi trường ánh sáng là điều tối quan trọng để đạt được hình ảnh hợp lý. Filament hỗ trợ một số loại đèn, chia thành hai loại, ánh sáng trực tiếp và gián tiếp:

* Chiếu sáng trực tiếp: đèn đúng giờ, đèn trắc quang, đèn khu vực.
* Ánh sáng gián tiếp: đèn dựa trên hình ảnh (IBL), cho cả đầu dò ánh sáng cục bộ và xa.

Ví dụ tạo một nguồn sáng định hướng, cũng như ánh sáng dựa trên IBL:

|  |
| --- |
| const sunlight = Filament.EntityManager.get().create();  scene.addEntity(sunlight);  Filament.LightManager.Builder(LightType.SUN)  .color([0.98, 0.92, 0.89])  .intensity(110000.0)  .direction([0.6, -1.0, -0.8])  .sunAngularRadius(1.9)  .sunHaloSize(10.0)  .sunHaloFalloff(80.0)  .build(engine, sunlight);  const backlight = Filament.EntityManager.get().create();  scene.addEntity(backlight);  Filament.LightManager.Builder(LightType.DIRECTIONAL)  .direction([-1, 0, 1])  .intensity(50000.0)  .build(engine, backlight); |

Để có được ánh sáng mạch lạc đúng cách, chúng ta phải sử dụng các đơn vị ánh sáng tôn trọng tỷ lệ giữa các cường độ ánh sáng khác nhau trong các cảnh trong thế giới thực. Các cường độ này có thể thay đổi rất nhiều, từ khoảng 800 lmlm đối với bóng đèn gia đình đến 120.000 lxlx đối với bầu trời ban ngày và ánh sáng mặt trời. Cách dễ nhất để đạt được sự đồng nhất về ánh sáng là sử dụng các đơn vị ánh sáng vật lý. Điều này sẽ cho phép toàn bộ khả năng tái sử dụng của giàn chiếu sáng. Sử dụng đơn vị ánh sáng vật lý cũng cho phép chúng tôi sử dụng máy ảnh dựa trên vật lý.

*Bảng 2* hiển thị đơn vị ánh sáng liên quan đến từng loại ánh sáng mà Filament hỗ trợ.

Bảng 2. Các loại ánh sáng và đơn vị

|  |  |
| --- | --- |
| Loại ánh sáng | Đơn vị |
| Ánh sáng định hướng | Độ sáng |
| Điểm sáng | Công suất sáng () |
| Đốm sáng | Công suất sáng () |
| Ánh sáng trắc quang | Cường độ sáng () |
| Ánh sáng mặt nạ trắc quang | Công suất sáng () |
| Ánh sáng khu vực | Công suất sáng () |
| Ánh sáng dựa trên hình ảnh | Độ chói |

# CHƯƠNG 2. THỰC HÀNH

## 1. Môi trường

Trong đề tài này, nhóm triển khai hệ thống trên môi trường như sau:

* Hệ điều hành: Ubuntu 20.04 LTS
* Trình duyệt hỗ trợ: Google Chrome và Firefox. Nhóm chưa triển khai trên các trình duyệt khác nên có thể có những lỗi tiềm ẩn khi chạy trên các trình duyệt khác.
* Máy tính có cài Python 3.7.
* Trong một số trường hợp, máy tính cần kết nối tới mạng Internet. (Do trong mã nguồn có trỏ tới một số tệp trực tuyến).

Nhóm triển khai đề tài sử dụng WebGL trên Ubuntu.

## 2. Giới thiệu chung

### 2.1. Các thành phần chính của project

* *Tệp HTML*

Tệp html có nhiệm vụ tạo ra một trang mạng để hiển thị thân thiện trên cả máy tính và thiết bị di động với canvas toàn màn hình. Tệp này sau thiết lập một vài thông số cơ bản của trang mạng như tiêu đề và vùng hiển thị thì load ba tệp JavaSripts gồm: Filament.js, gl-matrix-min.js và tệp Javascript điều khiển vật thể

* Tệp Filament.js và gl-matrix-min.js

Tệp Filament.js có nhiệm vụ tải xuống các asset và biên dịch mô-đun WASM của Filament. Tệp này còn bao gồm các tiện tích cấp cao, ví dụ như để đơn giản hóa việc loading các textures KTX từ JavaScript. Tệp gl-matrix-min.js là một thư viện nhỏ gồm các hàm vector toán học.

* Tệp Javascript điều khiển vật thể

Đây là tệp thiết lập chính của project. Tệp này gồm ba phần chính là phần constructor, phần render và phần resize.

Phần Constructor có nhiệm vụ tạo ra các thực thể như là đối tượng hiển thị, ánh sáng, bối cảnh… Ở đây, sau khi scene và engine được tạo, đối tượng hiển thị sẽ được tạo ra rồi thiết lập các thuộc tính như vậy liệu(material), kết cấu vật liệu(textured material), kết cấu bề mặt(mipmapped textures), vị trí,… Vật liệu của đối tượng không được tạo trực tiếp từ tệp này mà được load từ một tệp được compile từ công cụ matc. Tương tự với thuộc tính kết cấu vật liệu và kết cấu bề mặt. Tiếp theo, bối cảnh cũng được load từ tệp ngoài và ánh sáng được tạo ra xung quanh đối tượng. Cuối cùng, camera được tạo và các hàm render, resize được gọi để khởi tạo ra cảnh và khung nhìn.

Đối với những đối tượng đơn giản như hình cầu có thể được tạo ra và quy định các thông số ngay trong tệp main.js. Tuy nhiên với những đối tượng phức tạp hơn thì cần phải load vật thể từ tệp ngoài. Các vật thể này được tạo ra từ các công cụ tạo hình 3D, tuy nhiên ở giới hạn của project này, nhóm chỉ sử dụng các vật thể được tạo ra sẵn và sử dụng tool filamesh để xuất ra dưới dạng định dạng lưới nhị phân.

Phần Render có nhiệm vụ xác định điểm nhìn và kết xuất ra cảnh. Đây là hàm mà trình duyệt gọi mỗi khi cần phải vẽ lại, thường là 60 lần mỗi giây.

Phần Resize có nhiệm vụ điều chỉnh khung nhìn và canvas. Bên cạnh đó, nó điều chỉnh độ phân giải của bề mặt được render ra ckhi kích thước cửa sổ thay đổi.

### 2.2 Chạy project

Sau khi tạo các tệp và thiết lập các thành phần như trên, một local server cần được tạo để hiển thị kết quả. Một terminal được mở ở thư mục project và sử dụng câu lệnh như sau:

*Python3 -m http.server*

Sau đó, để xem kết quả hiển thị của project cần truy cập đến địa chỉ http://localhost:8000 và chọn tệp HTML.

## 3. Thiết lập các thành phần cần thiết

### 3.1. Tạo lâp tệp HTML

Tệp HTML là tệp có nhiệm vụ tạo lên giao diện web mà chúng ta nhìn thấy. Đối với đề tài này, tệp HTML sẽ có dạng như sau:

|  |
| --- |
| <!DOCTYPE html>  <**html** lang="en">  <**head**>  <**title**>Da phuong tien Nang cao</**title**>  <**meta** charset="utf-8">  <**meta** name="viewport" content="width=device-width,user-scalable=no,initial-scale=1">  <**style**>  **body** { **margin**: 0; **overflow**: **hidden**; }  **canvas** { touch-action: **none**; **width**: 100%; **height**: 100%; }  </**style**>  </**head**>  <**body**>  <**canvas**></**canvas**>  <**script** src="//unpkg.com/filament/filament.js"></**script**>  <**script** src="//unpkg.com/gl-matrix@2.8.1/dist/gl-matrix-min.js"></**script**>  <**script** src="lucy.js"></**script**>  </**body**>  </**html**> |

Ở đây, chúng ta có thể quy định được tiêu đề trang, thiết lập chế độ hiển thị responsive, gọi tới các thư viện Javascript phục vụ cho việc hiển thị, xử lý đối tượng và tính toán các hàm, ma trận toán học. Phần quan trọng nhất là việc ta gọi tới tệp Javascript tạo lập và điều khiển vật thể 3D. Cụ thể trong ví dụ này, ta đang gọi tới lucy.js.

### 3.2 Tạo lập tệp Javascript điều khiển vật thể

Như đã đề cập ở trên, tệp Javascript sẽ thực hiện các tính năng như sau:

* Thiết lập chương trình, khung nhìn.
* Thiết lập đối tượng vật thể.
* Thiết lập vật liệu.
* Thiết lập ánh sáng.
* Thiết lập camera.

Cấu trúc cơ bản của một tệp Javascript như sau:

|  |
| --- |
| **class** App {  constructor() {  *// TODO: create entities*  **this**.render = **this**.render.bind(**this**);  **this**.resize = **this**.resize.bind(**this**);  window.addEventListener('resize', **this**.resize);  window.requestAnimationFrame(**this**.render);  }  render() {  *// TODO: render scene*  window.requestAnimationFrame(**this**.render);  }  resize() {  *// TODO: adjust viewport and canvas*  }  }  Filament.init([filamat\_url, filamesh\_url, sky\_large\_url, ibl\_url], () => {  window.Fov = Filament.Camera$Fov;  window.LightType = Filament.LightManager$Type;  window.app = new App(document.getElementsByTagName('canvas')[0]);  }); |

Chương trình sẽ tạo ra một lớp App, trong đó gọi hàm tạo và quy định việc xuất (render) và việc thay đổi kích cỡ góc nhìn (resize).

Chúng ta phải tạo ra khung hình để nhìn thấy như sau:

|  |
| --- |
| **this**.canvas = document.getElementsByTagName('canvas')[0];  **const** engine = **this**.engine = Filament.Engine.create(**this**.canvas);  **this**.scene = engine.createScene();  **this**.triangle = Filament.EntityManager.get().create();  **this**.scene.addEntity(**this**.lucy); |

Ngoài ra, các yếu tố khác sẽ được đề cập dưới dây, gồm: tạo lập đối tượng, tạo lập vật liệu, ánh sáng. Các yếu tố này đều được gọi từ tệp Javascript tới những tệp khác trong cùng dự án.

### 3.3 Tạo lập tệp đối tượng filamesh.

Như đã trình bày ở trên, đối tượng mà chúng ta nhìn thấy ở dạng 3D sẽ được lưu dưới dạng một tệp filamesh. Để có được tệp filamesh, nhà cung cấp có hỗ trợ một công cụ để chuyển từ tệp đối tượng 3D có phần mở rộng à .obj sang phần mở rộng là .filamesh.

Các tools (công cụ) hỗ trợ đều được nhà cung cấp đưa ra tại đường dẫn: <https://github.com/google/filament/tree/main/tools>

Trong phạm vi đề tài, nhóm không tạo tệp .obj vì vượt ra khỏi phạm vi môn học và sử dụng một số phần mềm đặc biệt khác. Do đó, nhóm sử dụng tệp .obj được cung cấp dưới dạng open source và được sử dụng một cách miễn phí.

Để chuyển từ tệp .obj sang tệp .filamesh, nhóm sử dụng tool filamesh với câu lệnh sau: filamesh --compress <input\_obj\_tệp> <output\_filamesh\_tệp>

Ví dụ: filamesh --compress lucy.obj lucy.filamesh

Khi đó, ta đã có một tệp đối tượng 3D mang tên lucy.filamesh để sử dụng bằng cách gọi tới từ tệp Javascript phía trên. Ta có thể áp dụng với nhiều đối tượng khác nhau, không giới hạn, chỉ cần là tệp .obj.

### 3.4 Vật liệu

Tương tự như trên, nhóm không tự tạo ra vật liệu mới mà sử dụng các vật liệu có sẵn như plastic.mat, textured.mat. Tuy nhiên, nhóm thay đổi các thông số (tham số) đặc trưng cho vật liệu để thấy được sự thay đổi của vật liệu trên bề mặt đối tượng 3D.

Tệp vật liệu sẽ là một tệp có định dạng .mat, được viết theo một quy ước có sẵn. Trong đó thể hiện các thông số đặc trưng cho loại vật liệu. Tuy nhiên, chương trình của nhóm không thể đọc trực tiếp dữ liệu .mat mà chỉ đọc được dữ liệu .filamat. Đây là một tệp đã được dịch từ dạng chữ đơn thuần (plain text) sang dạng binary để chương trình có thể hiểu được.

Ở đây, chúng ta sử dụng công cụ matc với câu lệnh như sau:

matc -a opengl -p mobile -o textured.filamat textured.mat

với textured.filamat là tên tệp binary sau khi dịch, còn textured.mat là tệp vật liệu đầu vào.

Khi đó, ta đã có một tệp textured.filamat. Tương tự, ta cũng có thể sử dụng nhiều vật liệu khác.

Để thiết lập các đặc tính như màu sắc, độ nhám, độ bóng, độ phản xạ, … của vật liệu, ta có hai cách: đó là ánh xạ với ảnh mẫu hoặc thiết lập tham số. Nhóm sẽ giới thiệu cụ thể về hai cách trên ở phía dưới.

#### 3.4.1 Tạo kết cấu vật được ánh xạ với ảnh.

Xem xét một đoạn mô tả vật liệu như hình dưới:

|  |
| --- |
| material {  name : textured,  requires : [ uv0 ],  shadingModel : lit,  parameters : [  { type : sampler2d, name : albedo },  { type : sampler2d, name : roughness },  { type : sampler2d, name : metallic },  { type : float, name : clearCoat },  { type : sampler2d, name : normal },  { type : sampler2d, name : ao }  ],  } |

Ta chú ý có thể thấy được trong các tham số của vật liệu, có một số tham số có kiểu là sampler2d. Đây là những tham số mà có thể được ánh xạ với một bức ảnh có sẵn. Nếu ta sử dụng vật liệu như trên, ta cần có 05 ảnh đề ánh xạ 05 tham số tới 5 ảnh đó, đặc trưng cho các tính chất về màu sắc, độ nhám, độ kim loại, độ bóng,..

Để thực hiệu quá trình ánh xạ, trước hết ta cần 05 bức ảnh mẫu thể hiện tính chất vật liệu. Nhà cung cấp có cung cấp một công cụ để ta có chuyển ảnh từ dạng thông thường (.png) sang một kiểu dữ liệu binary có phần mở rộng là .ktx. Đó là công cụ mipgen và được sử dụng như sau:

|  |
| --- |
| *# Create mipmaps for base color and two compressed variants.*  mipgen albedo.png albedo.ktx  mipgen --compression=astc\_fast\_ldr\_4x4 albedo.png albedo\_astc.ktx  mipgen --compression=s3tc\_rgb\_dxt1 albedo.png albedo\_s3tc.ktx  *# Create mipmaps for the normal map and a compressed variant.*  mipgen --strip-alpha --kernel=NORMALS --linear normal.png normal.ktx  mipgen --strip-alpha --kernel=NORMALS --linear --compression=etc\_rgb8\_normalxyz\_40 **\**  normal.png normal\_etc.ktx  *# Create mipmaps for the single-component roughness map and a compressed variant.*  mipgen --grayscale roughness.png roughness.ktx  mipgen --grayscale --compression=etc\_r11\_numeric\_40 roughness.png roughness\_etc.ktx  *# Create mipmaps for the single-component metallic map and a compressed variant.*  mipgen --grayscale metallic.png metallic.ktx  mipgen --grayscale --compression=etc\_r11\_numeric\_40 metallic.png metallic\_etc.ktx  *# Create mipmaps for the single-component occlusion map and a compressed variant.*  mipgen --grayscale ao.png ao.ktx  mipgen --grayscale --compression=etc\_r11\_numeric\_40 ao.png ao\_etc.ktx |

Trong ví dụ này, ta sử dụng 05 bức ảnh png gồm: albedo.png, roughness.png, normal.png, metallic.png và ao.png để chuyển về dạng .ktx. Ngoài việc chuyển trực tiếp thì công cụ còn thực hiện việc nén tệp .ktx để sử dụng với các màn hình có độ phân giải và kích cỡ màn hình thấp hơn.

Khi đó, ta sẽ có những tệp .ktx đặc trưng cho các tính chất kết cấu của vật liệu.

Tương tự với những vật liệu khác, ta sẽ cần sử dụng những bộ ảnh mẫu khác nhau về hình ảnh và số lượng, tùy theo số lượng tham số, đặc trưng tham số.

#### 3.4.2 Tạo đặc trưng vật liệu bằng cách thiết lập tham số

Xem xét vật liệu plastic.mat có khai báo kiểu vật liệu như sau:

|  |
| --- |
| material {  name : Lit,  shadingModel : lit,  parameters : [  { type : float3, name : baseColor },  { type : float, name : roughness },  { type : float, name : clearCoat },  { type : float, name : clearCoatRoughness }  ],  } |

`Ta thấy, ở đây, các tham số có kiểu dữ liệu là các kiểu dữ liệu số học, có thể gán bằng các số liệu cụ thể. Chính vì vậy, chúng ta không dùng phương pháp ánh xạ ảnh mẫu đối với các vật liệu như plastic.mat. Chúng ta sẽ thiết lập trực tiếp tham số. Ví dụ như:

|  |
| --- |
| **const** material = engine.createMaterial(filamat\_url);  **const** matinstance = material.createInstance();  **const** red = [0.8, 0.0, 0.0];  matinstance.setColor3Parameter('baseColor', Filament.RgbType.sRGB, red);  matinstance.setFloatParameter('roughness', 0.5);  matinstance.setFloatParameter('clearCoat', 1.0);  matinstance.setFloatParameter('clearCoatRoughness', 0.3); |

Ở đây, chúng ta đang thiết lập các tham số baseColor, roughness, clearCoat, clearCoatRoughness với giá trị lần lượt là [0.8, 0, 0], 0.5, 1.0, 0.3. Trong trường hợp này, ta không ánh xạ với một ảnh mẫu nào cả.

### 3.5 Ánh sáng

Nhà cung cấp cho phép chúng ta sử dụng hai loại ánh sáng, bao gồm: ánh sáng mặt trời và ánh sáng định hướng. Ta cùng xem xét hai loại ánh sáng trên.

#### 3.5.1 Ánh sáng mặt trời

Ta xem xét đoạn mã nguồn sau:

|  |
| --- |
| **const** sunlight = Filament.EntityManager.get().create();  scene.addEntity(sunlight);  Filament.LightManager.Builder(LightType.SUN)  .color([0.98, 0.92, 0.89])  .intensity(110000.0)  .direction([0.6, -1.0, -0.8])  .sunAngularRadius(1.9)  .sunHaloSize(10.0)  .sunHaloFalloff(80.0)  .build(engine, sunlight); |

Trước hết, ta cần khởi tạo một thực thể mang tên sunLight (hoặc tên tùy ý), sau đó thêm thực thể đó vào màn hình khung cảnh.

Filament cung cấp một lớp LightManager giúp quản lý và xây dựng hình ảnh. Ở đây, ta có thể gọi tới kiểu ánh sáng là SUN, quy định màu sắc (color), mật độ (intensity), hướng (direction), các đặc tính của mặt trời. Và đưa những yếu tố đó gán với thực thể sunLight.

Từ đó, ta có được ánh sáng mặt trời như mong muốn.

#### 3.5.2 Ánh sáng định hướng

Tương tự như trên, nhưng ánh sáng định hướng thì không cần thiết lập các đặc tính như sunAngularRadius, sunHaloSize, SunHaloFalloff.

|  |
| --- |
| **const** backlight = Filament.EntityManager.get().create();  scene.addEntity(backlight);  Filament.LightManager.Builder(LightType.DIRECTIONAL)  .color([0.98, 0.92, 0.89])  .direction([-1, 0, 1])  .intensity(50000.0)  .build(engine, backlight); |

### 3.6 Camera

Để tạo ra đối tượng camera, ta khai báo:

**this**.camera = **this**.engine.createCamera();

Để quy định góc nhìn và vị trí đặt camera, ta khai báo như sau:

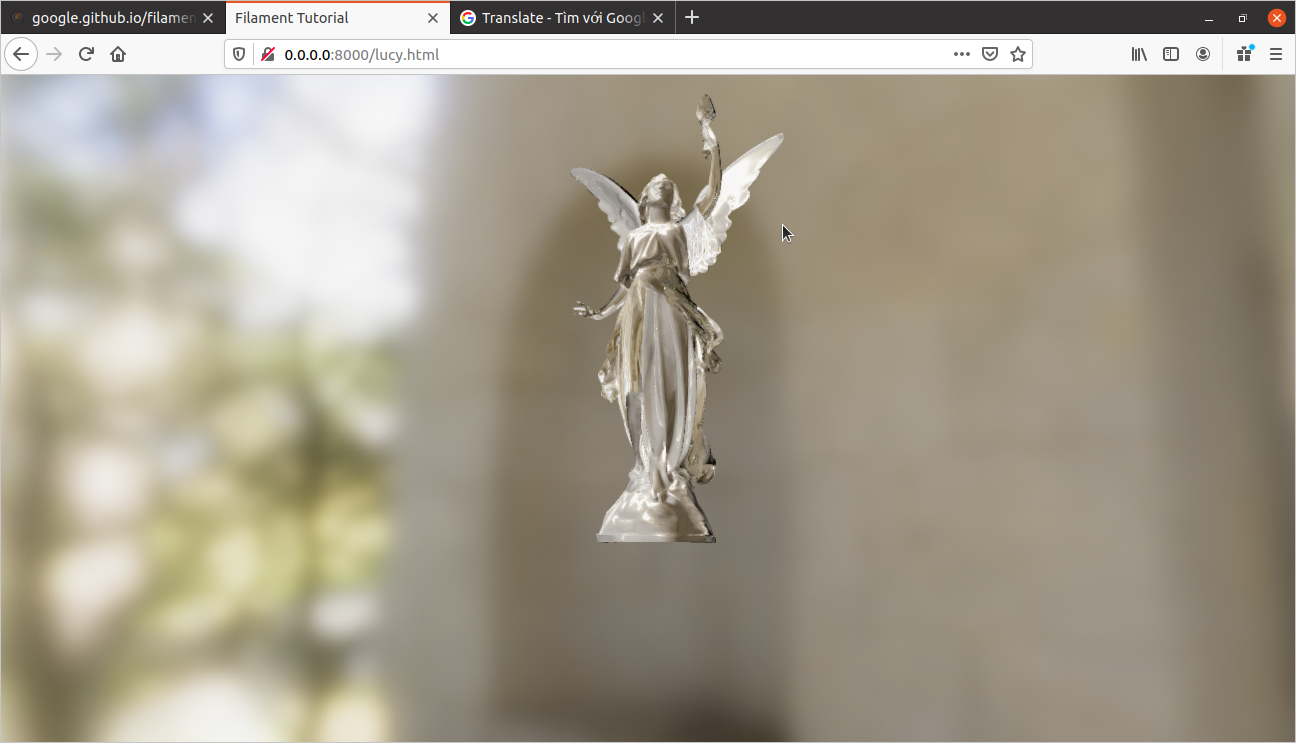
**const** eye = [0, 0, 4], center = [0, 0, 0], up = [0, 1, 0];

**this**.camera.lookAt(eye, center, up);

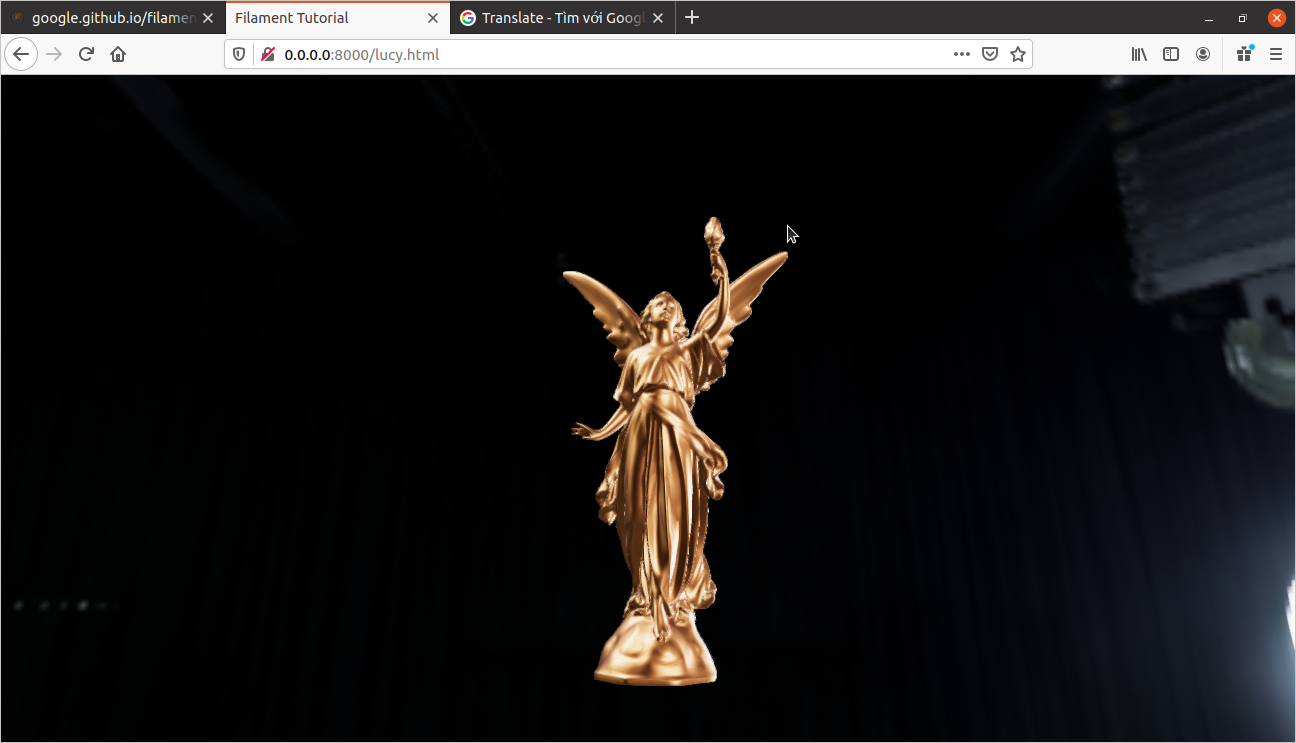
Với eye là vị trí đặt camera, center là vị trí trung tâm của hướng nhìn, up là hướng lên của camera.

Trên đây, nhóm đã giới thiệu về cách triển khai dự án Filament trên nền tảng Web, với hệ điều hành Ubuntu 20.04 LTS. Nhóm đã đạt được kết quả như sau:

# CHƯƠNG 3: KẾT QUẢ

Chương trình với đối tượng lucy.obj, vật liệu là texturedLit.filamat, có thêm ánh sáng mặt trời và ánh sáng định hướng.

Hình 1. Đối tượng đồ họa với vật liệu texturedLit.filamat

Chương trình với vật liệu aiDefault.mat ở trong scence tối, với thiết lập: baseColor = [0.8, 0.5, 0.3], metallic = 1.0, roughness = .5, reflectance = 0.7

Hình 2. Đối tượng đồ họa với vật liệu aiDefault.mat

# DANH MỤC THAM KHẢO

1. <https://google.github.io/filament/documentation/>
2. <https://github.com/google/filament/blob/main/web/docs/tutorial_triangle.md>
3. <https://github.com/google/filament/blob/main/web/docs/tutorial_redball.md>
4. <https://github.com/google/filament/blob/main/web/docs/tutorial_suzanne.md>