ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

----□&□----

BÁO CÁO ĐỒ ÁN:

XÂY DỰNG TRÒ CHƠI HYPER RAIDER SỬ DỤNG THƯ VIỆN THREE.JS

Giảng viên: TS. Mai Tiến Dũng
Môn học: Đồ họa Máy tính – CS105.M21

Thành viên nhóm:

STT	MSSV	Họ Tên
1	19521388	Hoàng Tiến Dũng
2	19522410	Nguyễn Thành Trọng

TP. HỒ CHÍ MINH - 06/2022

LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện tại Trường Đại học Công Nghệ Thông Tin, chúng em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành đồ án môn học của mình.

Chúng em xin gửi lời cảm ơn chân thành đến thầy TS. Mai Tiến Dũng – Giảng viên phụ trách lớp CS105.M21 – Môn Đồ họa Máy tính đã tận tâm hướng dẫn, truyền đạt những kiến thức cũng như kinh nghiệm cho chúng em trong suốt thời gian học tập.

Trong quá trình làm đồ án môn học, mặc dù nhóm chúng em đã cố gắng nhưng chắc chắn sẽ không tránh được những sai sót không đáng có. Mong nhận được sự góp ý cũng như kinh nghiệm quý báu của các các thầy và các bạn sinh viên để được hoàn thiện hơn và rút kinh nghiệm cho những môn học sau. Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, tháng 06 năm 2022.

MỤC LỤC

I. GIỚI THIỆU ĐỀ TÀI

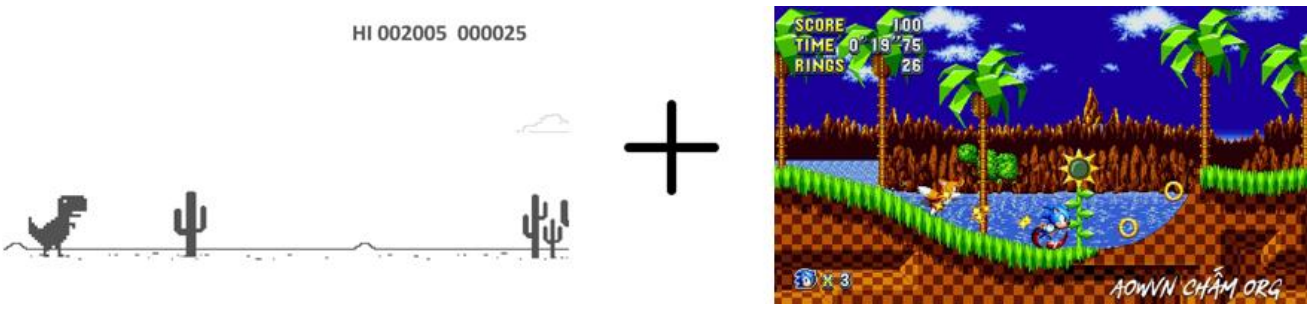
1. Lý do chọn đề tài

Đồ họa máy tính là một lĩnh vực của khoa học máy tính nghiên cứu về cơ sở toán học, các thuật toán cũng như các kĩ thuật để cho phép tạo, hiển thị và điều khiển hình ảnh trên màn hình máy tính. Đồ họa máy tính có liên quan ít nhiều đến một số lĩnh vực như đại số, hình học giải tích, hình học họa hình, quang học,... và kĩ thuật máy tính, đặc biệt là chế tạo phần cứng.

Các công cụ, thư viện hỗ trợ cho Đồ họa máy tính đang dần trở nên mạnh mẽ. Các nhà phát triển trò chơi có thể dễ dàng sử dụng những công cụ đó để phát triển các thể loại game 3D. Các loại game 3D rõ ràng đã thu hút rất nhiều người chơi. Do đó, trong đồ án môn học này, dựa vào kiến thức được tiếp thu từ môn Đồ họa máy tính, chúng tôi đã xây dựng Hyper Raider Game bằng thư viện Three.js.

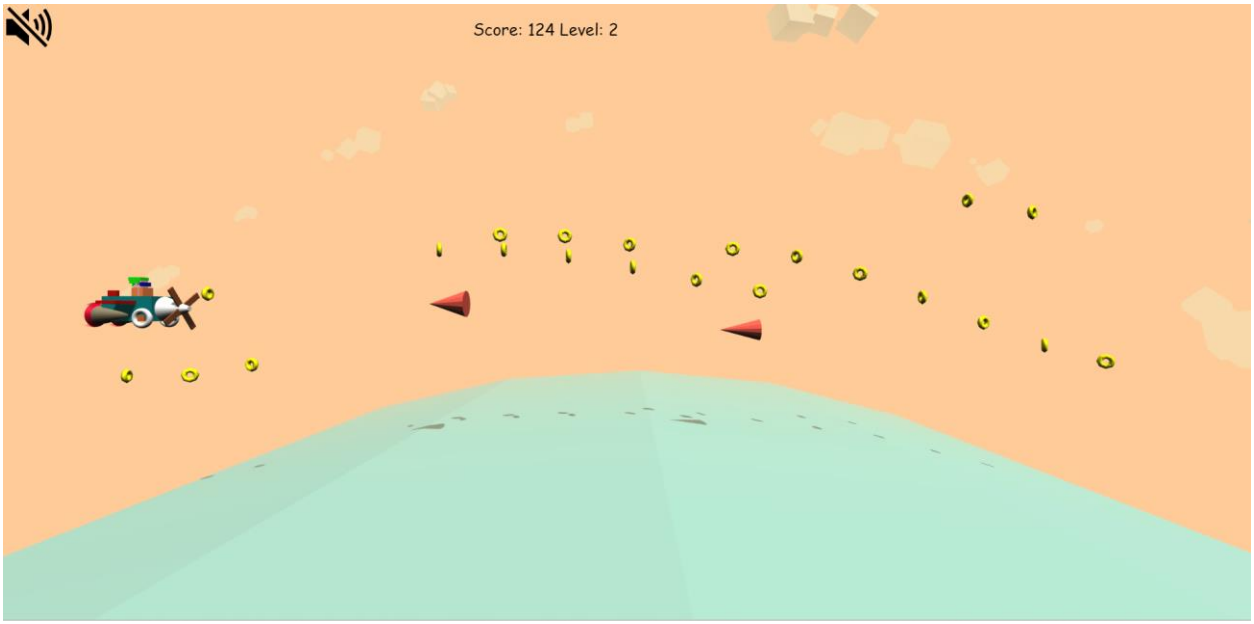
2. Tổng quan đề tài

Từ trước đến giờ chúng ta đã khá quen thuộc với các game liên quan đến máy bay, ví dụ: bắn gà, bắn máy bay,… Nhóm chúng tôi đã lấy ý tưởng hình tượng từ các trò chơi đó kết hợp với ý tưởng từ game khủng long và game sonic để xây dựng Hyper Raider Game.

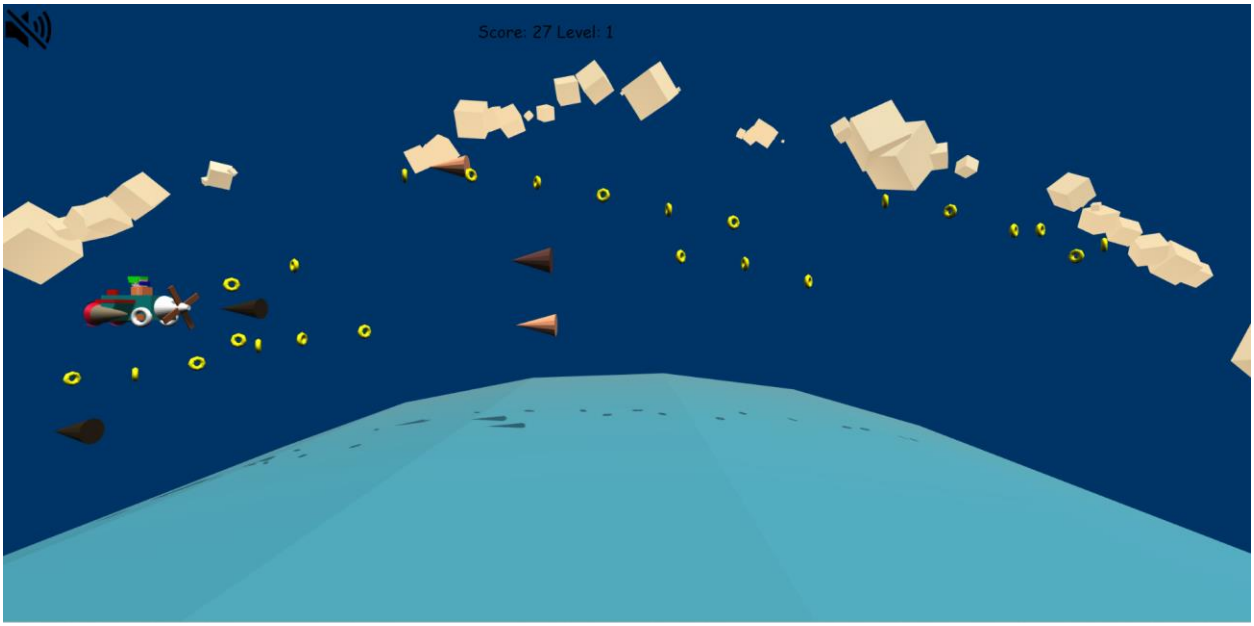


Hình 1:Ý tưởng của Hyper Raider Game.

Hyper Raider Game sẽ có 2 giao diện chính: sáng, tối.



Hình 2: Giao diện sáng.



Hình 3: Giao diện tối.

Về luật chơi:

- Người chơi điều khiển máy bay để tránh các vật cản (hình nón) và ăn các đồng xu để tăng điểm.
- Độ khó sẽ tăng dần theo thời gian và tỉ lệ thuận với level.
- Khi chạm vào vật cản thì kết thúc trò chơi.

II. NỘI DUNG

1. Three.js

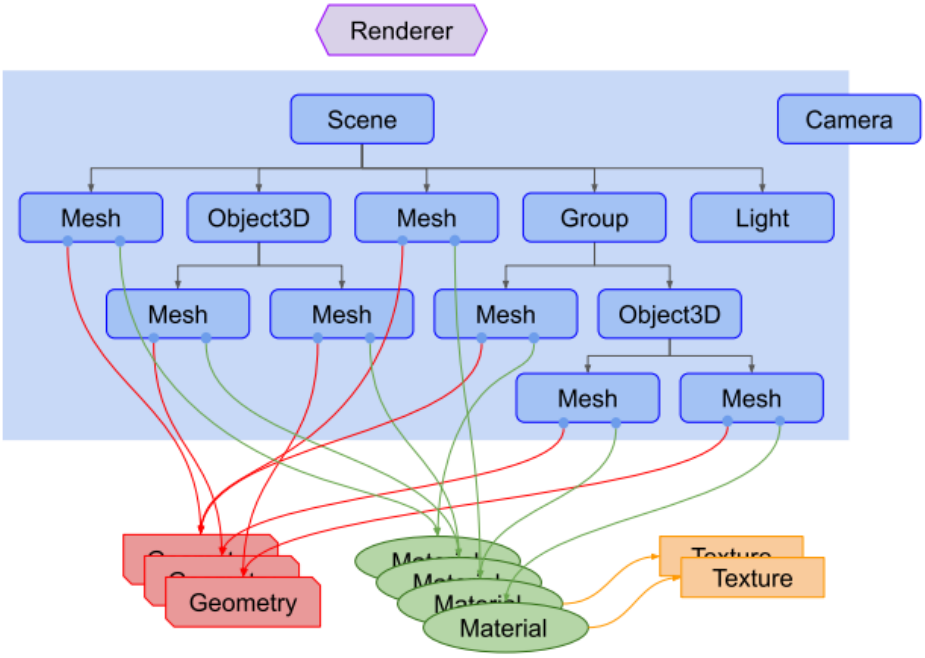
Three.js là một thư viện của Javascript giúp tạo và hiển thị đồ họa 3D trên web.

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

2. Lý thuyết

a. Cấu trúc cơ bản của một 3D app

Một cấu trúc cơ bản của một 3D app nó sẽ gồm các phần như là Scene, Camera, Renderer, Light và Mesh.

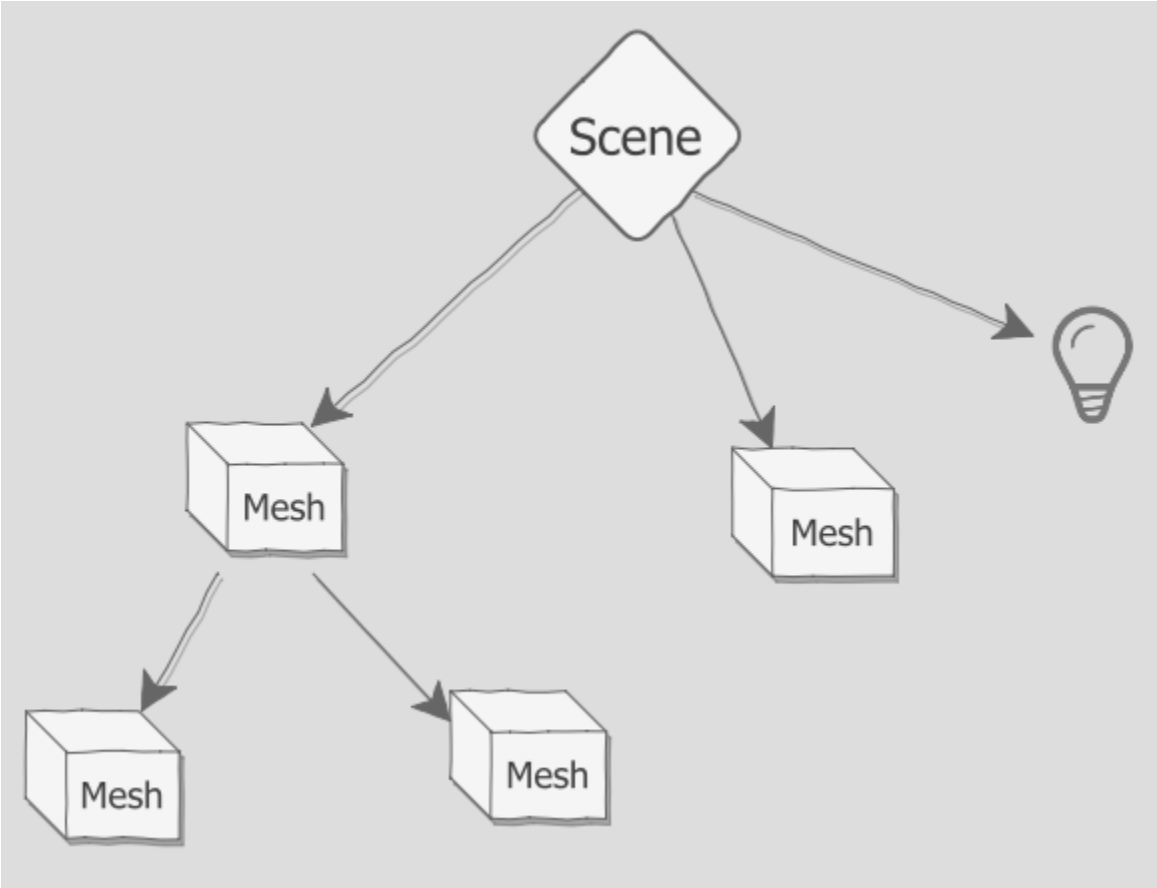


Hình 4: Sơ đồ cấu trúc cơ bản của một 3D app

b. Scene

Scene giống như một không gian 3D mà trong đó bạn có thể đặt các đối tượng như Mesh và Light. Scene là một cấu trúc mà đôi khi còn được gọi là scene graph. Một scene graph là một cấu trúc mà giữ tất cả các thông tin cần thiết của cảnh. Trong Three.js, điều đó có

nghĩa là Scene chứa tất cả các đối tượng, nguồn sáng, và các đối tượng khác cần thiết để render. Như cái tên ám chỉ, scene graph không chỉ là một mảng các đối tượng; nó bao gồm một tập các node dạng cây.

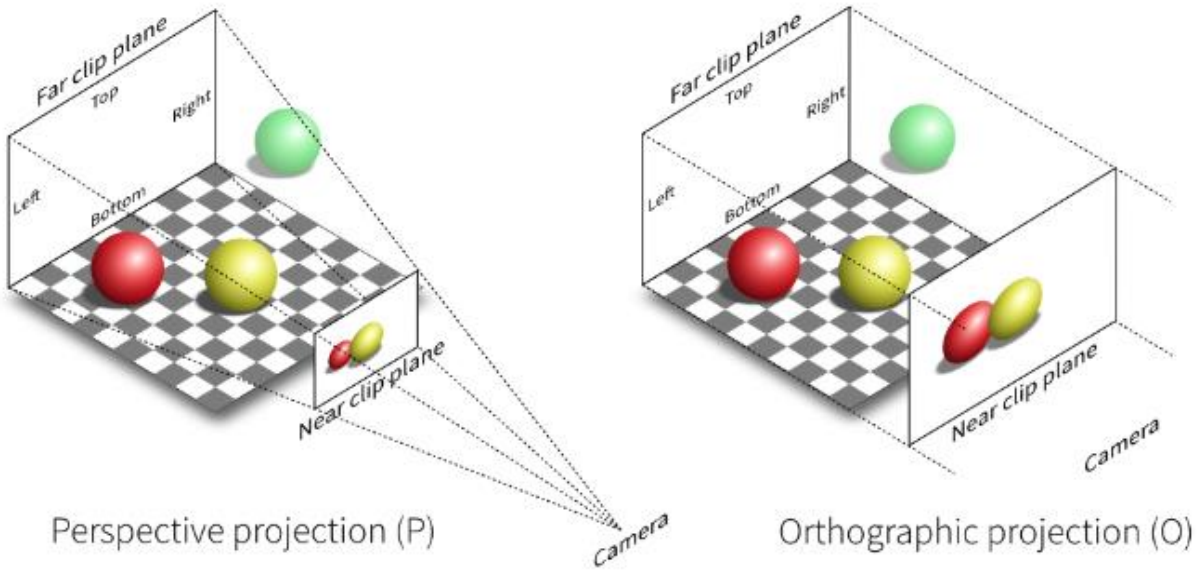


Hình 5: Cây sơ đồ Scene.

c. Camera

Thành phần Camera trong ThreeJS được xem giống với camera ngoài đời thuật với chức năng lưu lại khung hình của Scene với góc chiếu được đưa vào Camera. Từ đó Renderer sẽ vẽ ảnh từ góc chiếu mà Camera thu được lên web.

Trong Three.js, chúng ta có hai loại Camera là OrthographicCamera và PerspectiveCamera.



Hình 6: Minh họa PerspectiveCamera và OrthographicCamera.

Tuy nhiên, chúng ta sẽ chỉ tập trung vào PerspectiveCamera vì nó giống thế giới thực nhất. Đối tượng càng xa Camera 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.

d. Renderer

Renderer có thể hiểu chính là bộ xử lý cảnh vật trong Scene mà Camera lấy được lên Canvas trên web để chúng ta nhìn thấy. Trong Three.js, chúng ta có các Renderer sau: WebGLRenderer, WebGL1Renderer, CSS2DRenderer, CSS3DRenderer, SVGRenderer. Tuy nhiên, chúng ta gần như chỉ sử dụng WebGLRenderer để tận dụng sức mạnh của WebGL.

e. Canvas

Là một phần tử HTML, Canvas là nơi Renderer vẽ lên đây.

Nói chung để ví dụ cho dễ hiểu thì 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 cảnh - scence gì. Lúc này điện thoại đóng vai trò là renderer lấy ra các cảnh - scene được camera ghi lại và chiếu lên màn hình điện thoại - canvas cho chúng ta nhìn.

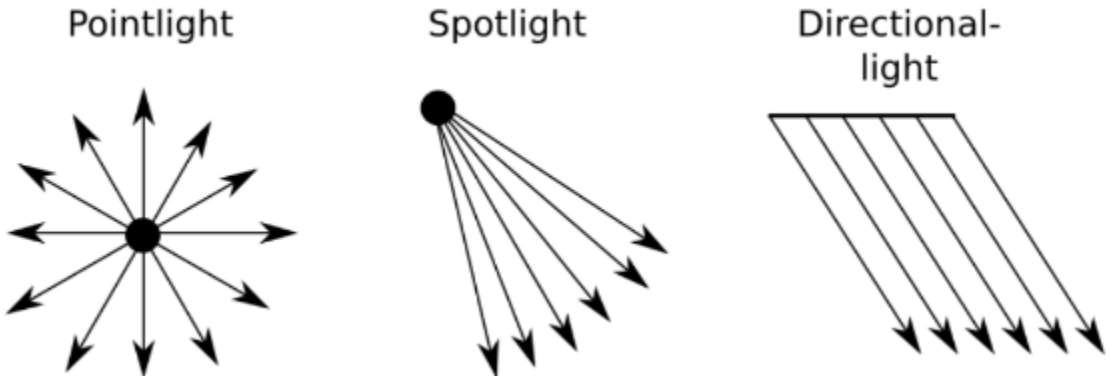
f. Light

Ở thế giới thật, các tia sáng có thể chiếu thẳng trực tiếp vào đối tượng, hoặc có thể va chạm vào các bề mặt khác nhau và phản xạ hoặc khuếch tán trước khi chạm vào đối tượng. Tuy nhiên, các máy tính hiện nay không thể có đủ sức mạnh để mô phỏng toàn bộ việc này ở real-time.

Three.js có các kiểu Light (nguồn sáng) sau:

- AmbientLight
- HemisphereLight
- DirectionalLight
- PointLight
- SpotLight
- RectAreaLight

Các kiểu Light khác nhau sẽ tạo ra các hiệu ứng khác nhau.

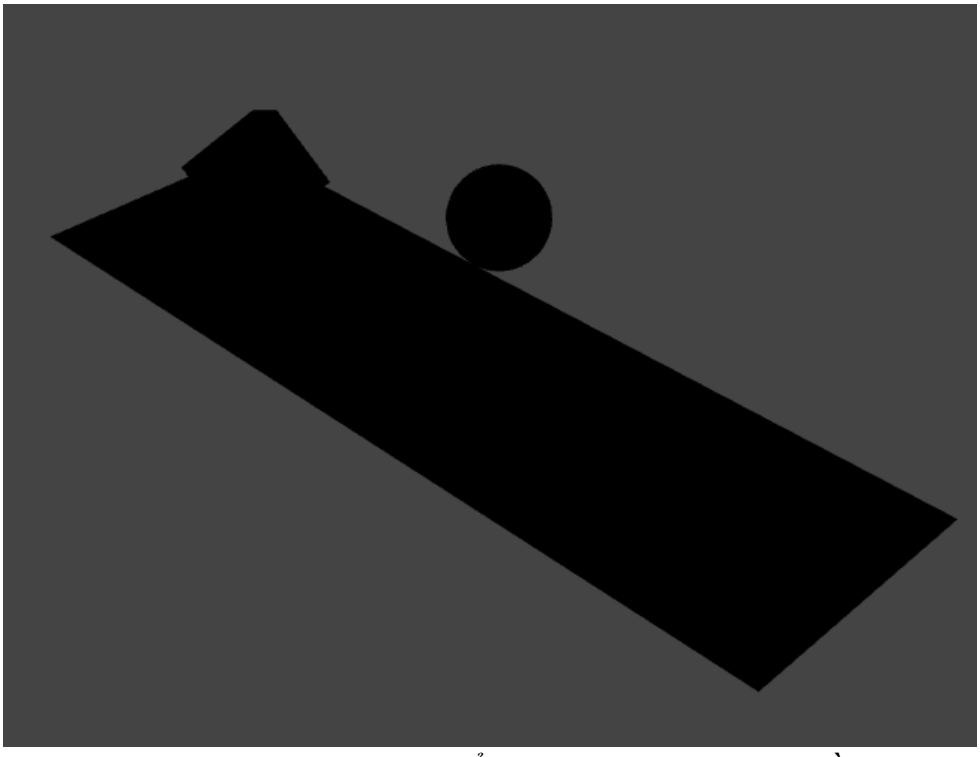


Hình 7: Các loại tia sáng của của Light sẽ khác nhau.

Trong bài đồ án này, nhóm sẽ sử dụng HemisphereLight và DirectionalLight để mô phỏng lại ánh sáng trong trò chơi.

i. NoLight

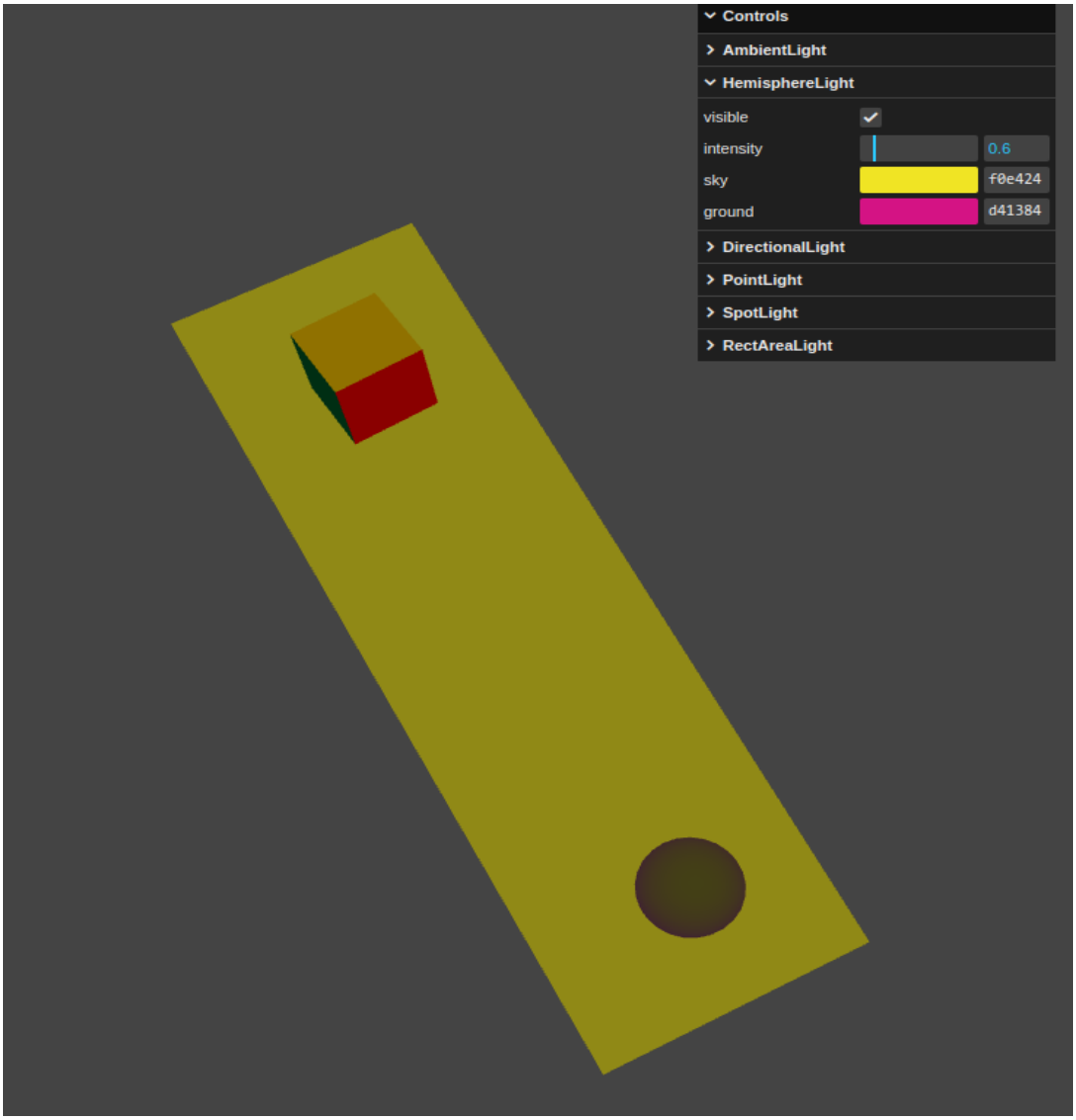
Chúng ta có thể thấy các đối tượng chỉ là các khối màu đen, mặc dù chúng ta đã truyền thuộc tính color cho MeshPhongMaterial.



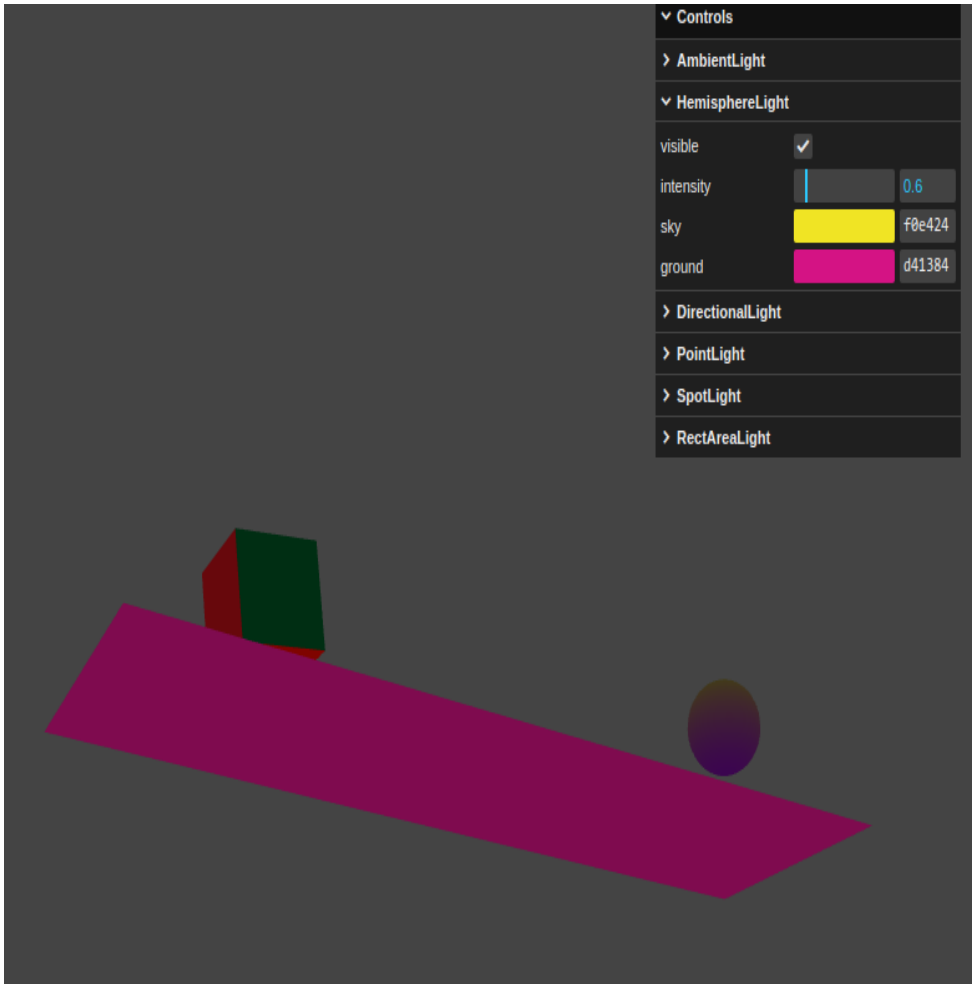
Hình 8: Minh họa các vật thể khi không có thành phần Light trong Scene.

ii. HemisphereLight

Đây là nguồn sáng đặc biệt và có thể được sử dụng để tạo các ngoại cảnh trông tự nhiên hơn bằng cách mô phỏng ánh sáng mạnh từ bầu trời và ánh sáng phản xạ nhẹ hơn từ mặt đất.



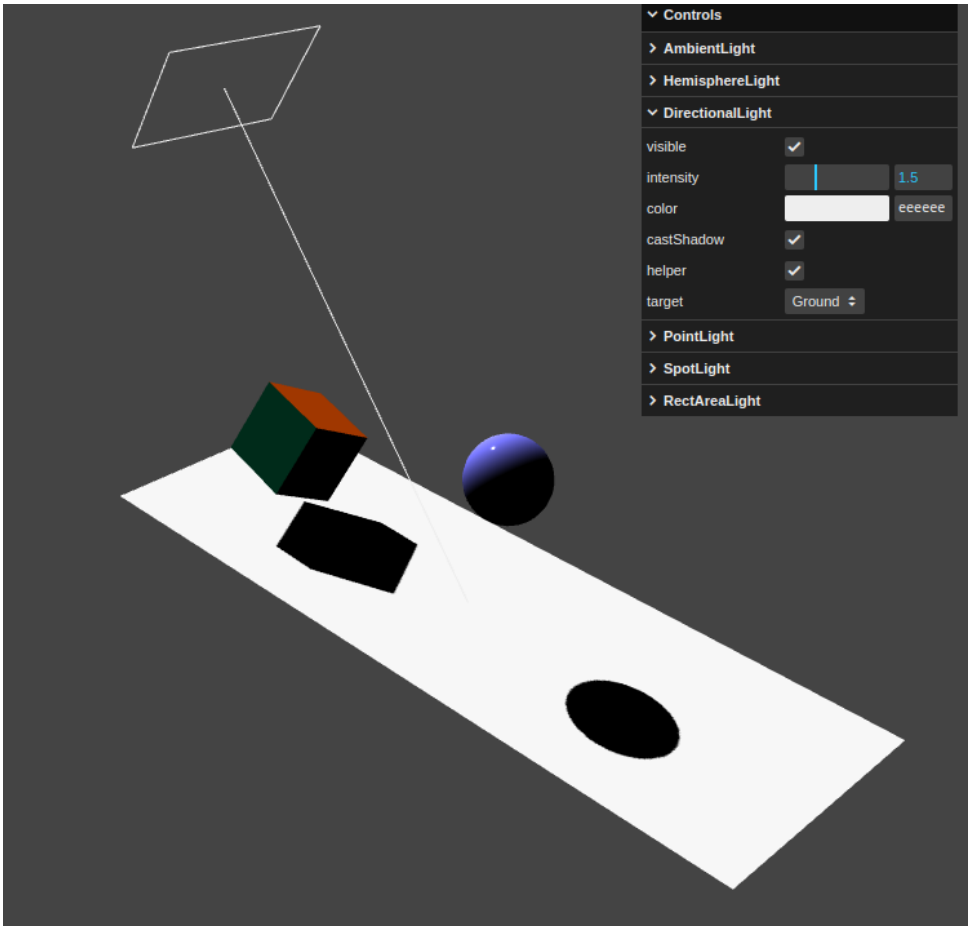
Hình 9: Ví dụ về HemisphereLight Sky.



Hình 10: Ví dụ về HemisphereLight Ground.

iii. DirectionalLight

Đây là nguồn sáng mà các tia sáng chiếu song song theo một chiều, ví dụ như ánh sáng mặt trời. Toàn bộ không gian được DirectionalLight chiếu với cùng một cường độ.



Hình 11: Ví dụ về DirectionalLight.

g. Âm thanh

- Nhạc nền:

Nhạc nền của Hyper Raider Game được lấy trên soundcloud. Người chơi có thể tắt mở theo tùy ý bằng cách tắt mở nút âm thanh ở trên cùng bên trái.

Hình 12: Nhạc nền.

- Nhạc khi va chạm:

Âm thanh khi va chạm được tải từ web pixabay trong đó có va chạm với vật thể và đồng xu.

Hình 13: Nhạc khi chạm vào đồng xu.

Hình 14: Nhạc khi chạm vào vật cản.

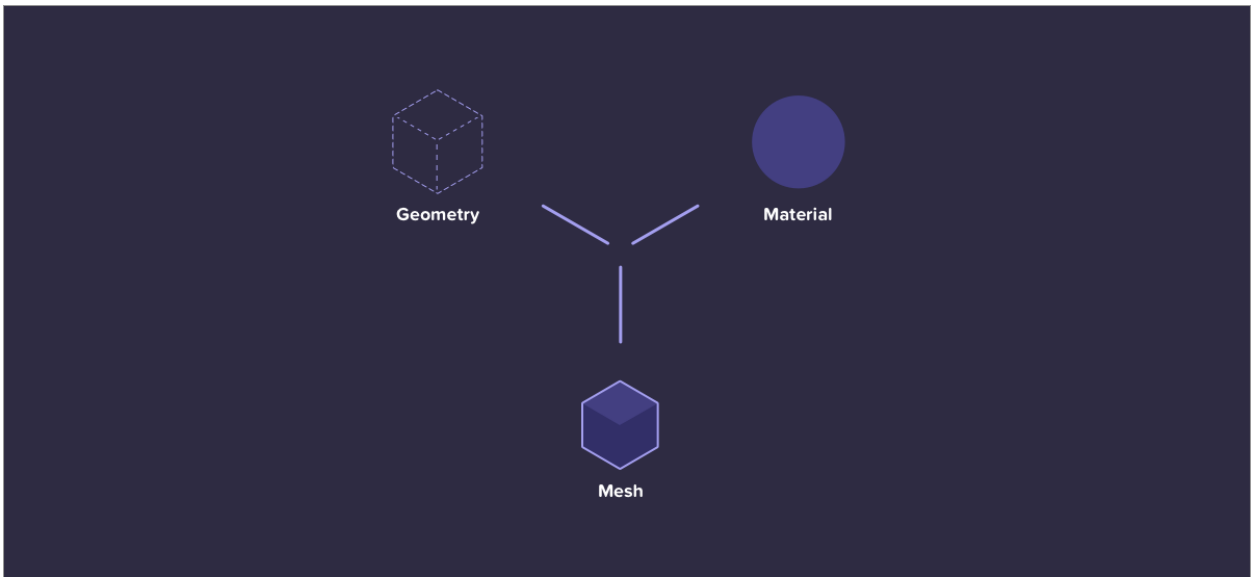
Khi chạm vào đồng xu phát ra âm thanh Ding 1, tới đồng xu tiếp theo nếu âm thanh trước đang còn phát thì cho dừng và phát âm thanh mới.

Khi chạm vào vật cản thì phát âm thanh Crash và kết thúc trò chơi.

h. Đối tượng trong game

- Đối tượng Mesh:

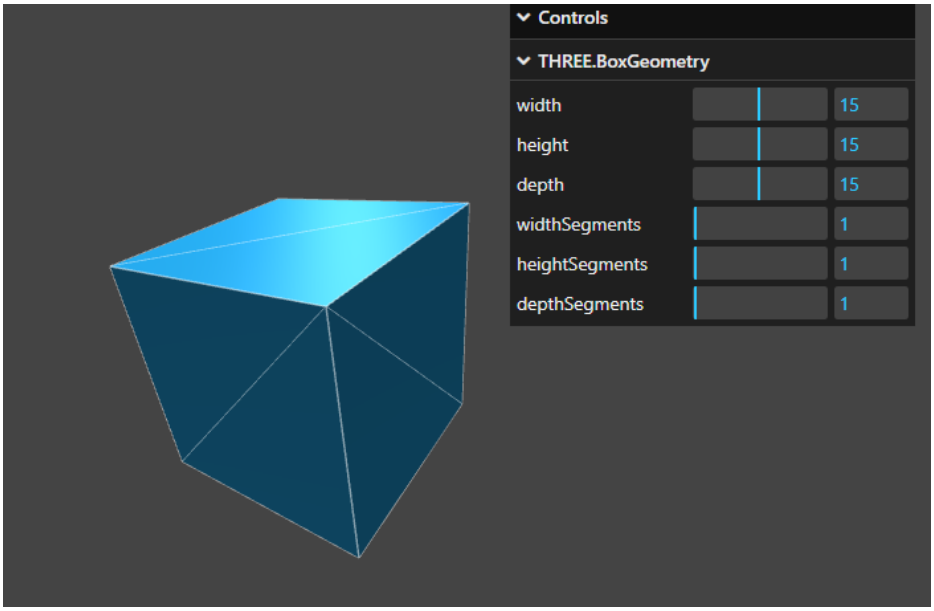
Như đã khái quát ở trên, một đối tượng Mesh được tạo từ Geometry và Material. Trong đó, Geometry như bộ khung xương và Material như bộ da.



Hình 15: Đối tượng Mesh.

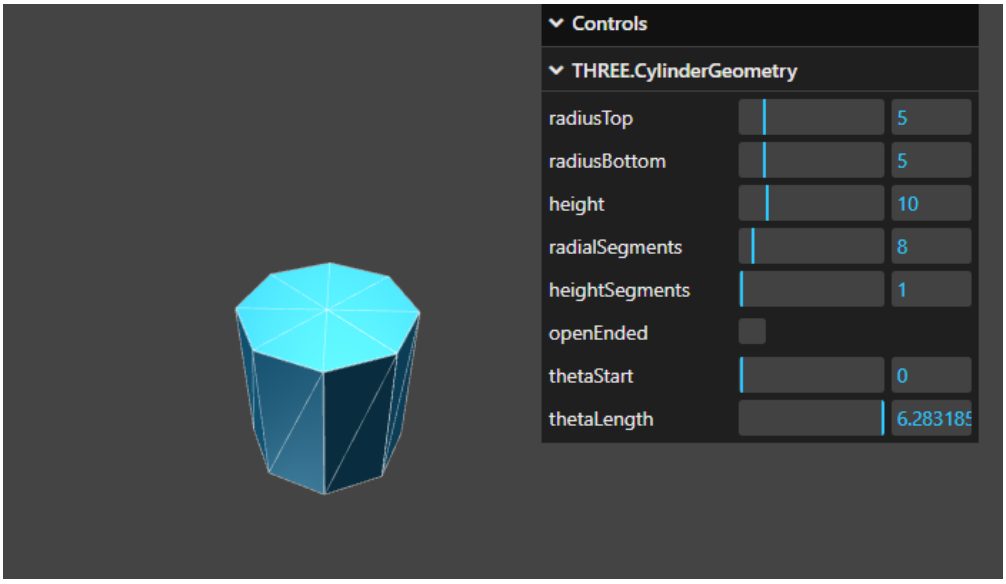
Cụ thể:

- Geometry là là một tập các điểm (đỉnh) và các mặt kết nối với các điểm đó. Thee.js có một tập nhiều các Geometry sẵn có, một số Geometry mà chúng tôi đã sử dụng là:
  - BoxGeometry



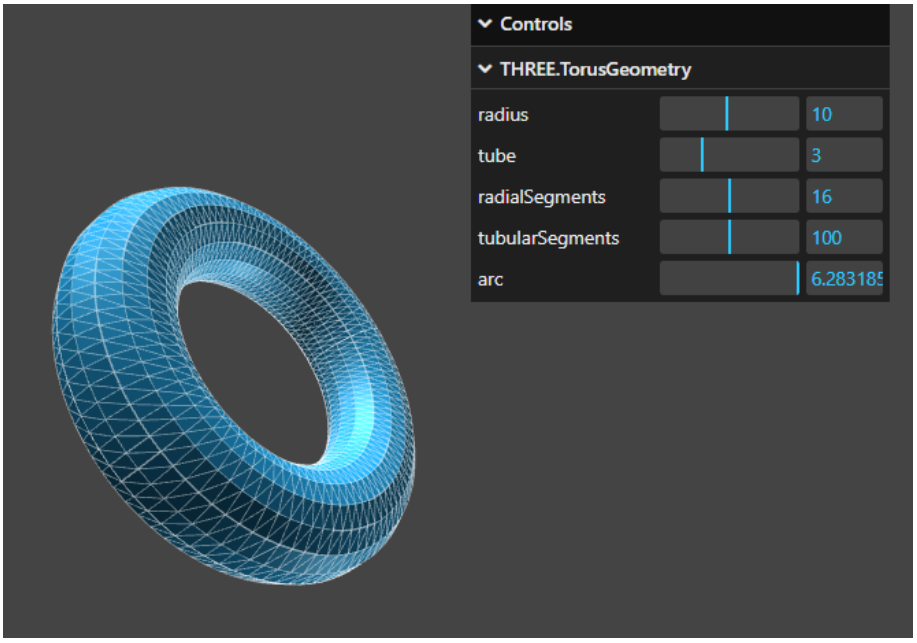
Hình 16: BoxGeometry.

  - CylinderGeometry



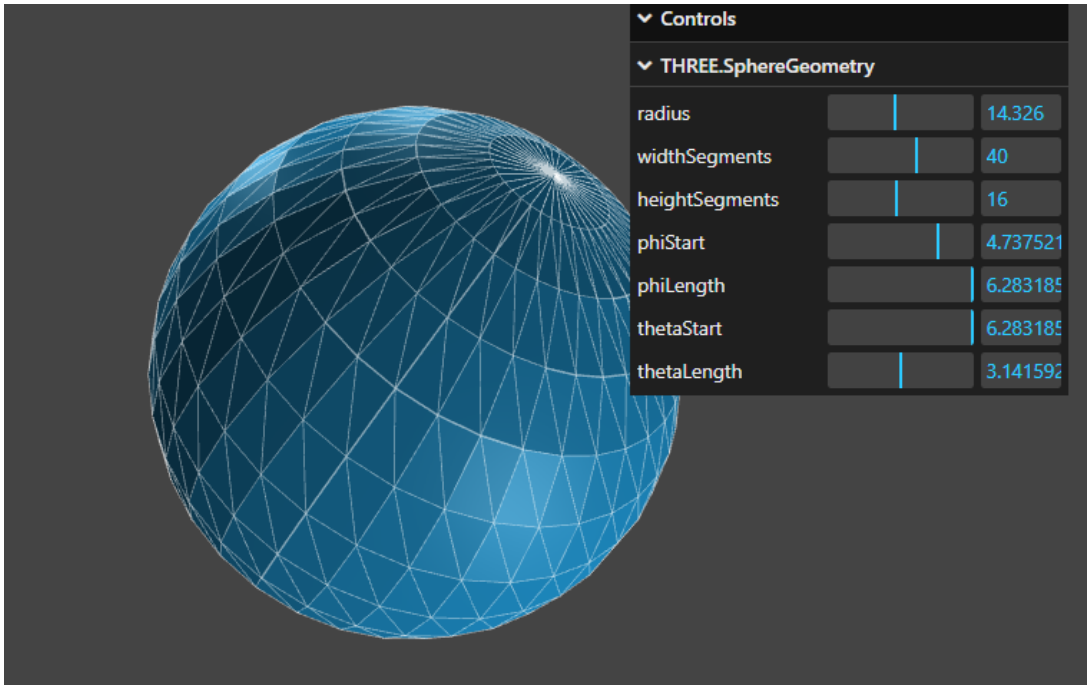
Hình 17: CylinderGeometry.

  - TorusGeometry



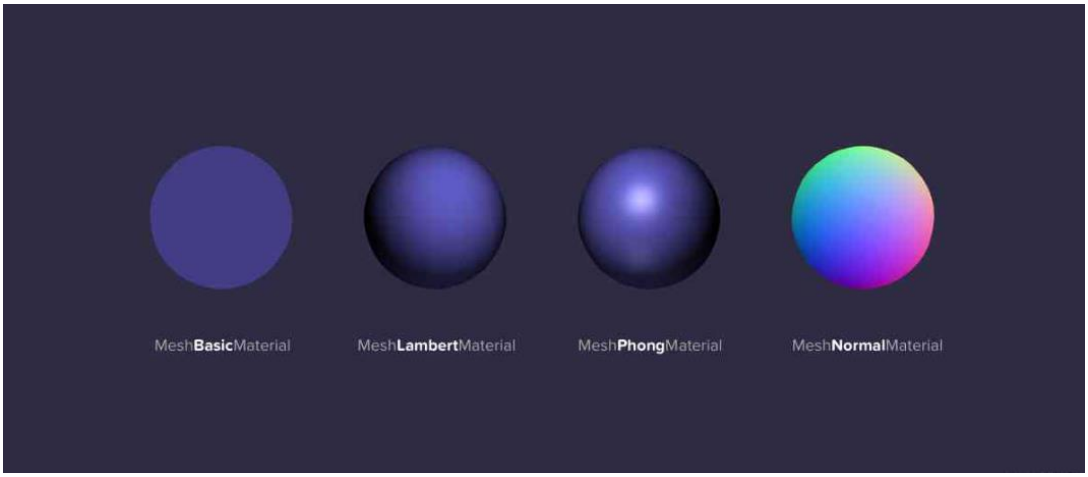
Hình 18: TorusGeometry.

- SphereGeometry



Hình 19: SphereGeometry.

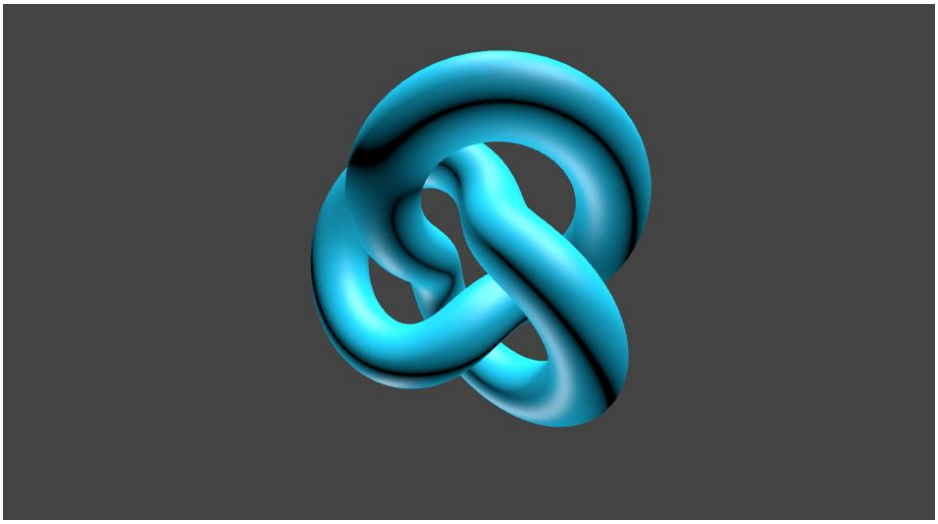
- Material xác định màu của một điểm trên đối tượng (Mesh), quyết định đối tượng trông như thế nào.



Hình 20: Các Material phổ biến.

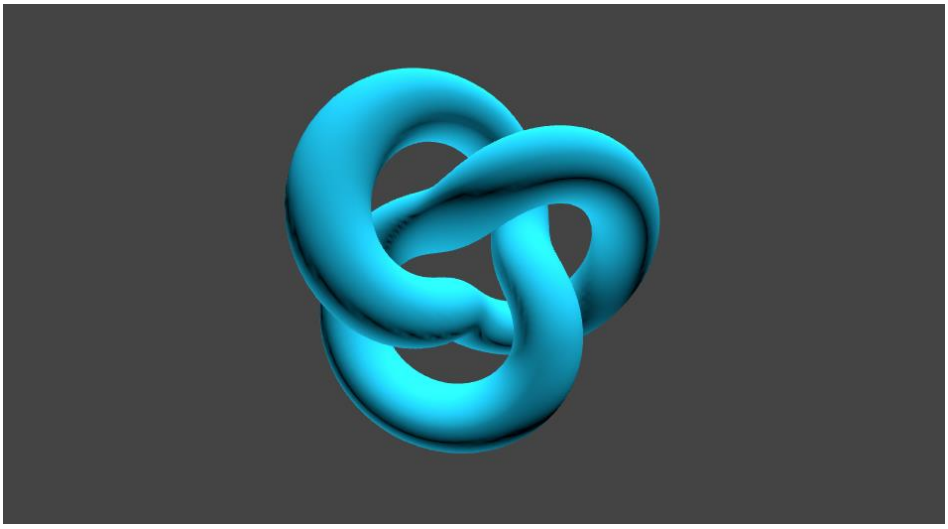
Các Material được sử dụng trong game là:

- MeshPhongMaterial: sử dụng ánh sáng, tạo đối tượng trông sáng bóng và nó tính toán ánh sáng ở tất cả các pixel.



Hình 21: MeshPhongMaterial.

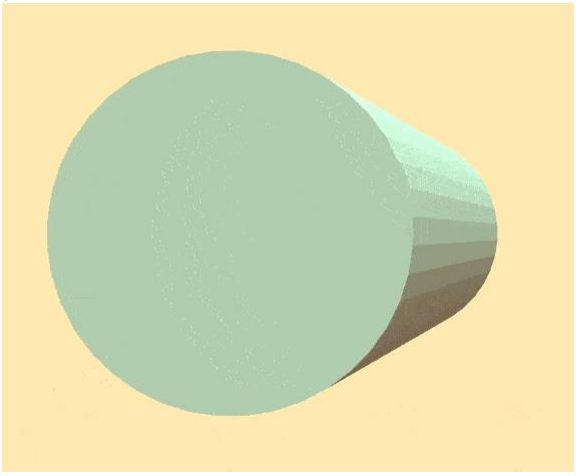
- MeshLambertMaterial: sử dụng ánh sáng, tạo đối tượng mờ, không sáng bóng và chỉ tính toán ánh sáng ở các đỉnh.



Hình 22: MeshLambertMaterial.

- Các vật thể trong Hyper Raider Game:
  - Biển:

    Biển được tạo từ CylinderGeometry và MeshPhongMaterial. Sau khi tạo đối tượng Mesh, ta xoay đối tượng theo trục X một góc 90^o. Tiếp theo cố định vị trí vật thể và xoay liên trục theo trục Z thì được như hình dưới.

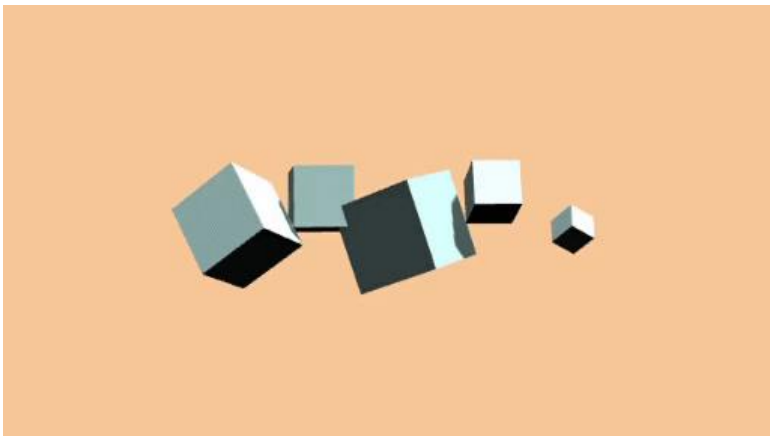


Hình 23: Mô phỏng mặt biển.

```
this.geom = new CylinderGeometry(700, 700, 600, 40, 10);
this.geom.applyMatrix4(new Matrix4().makeRotationX(-Math.PI/2));
this.mat = new MeshPhongMaterial({
  color: Colors.blue,
  transparent: true,
  opacity: 0.6,
  flatShading: true,
});
this.mesh = new Mesh(this.geom, this.mat);
this.mesh.receiveShadow = true;
```

Hình 24: Code tạo mặt biển.

- Bầu trời:
  - Bầu trời được tạo bởi nhiều đám mây cách đều nhau trên một đường tròn.
  - Mỗi đám mây gồm 3-5 khối lập phương ghép lại với nhau với cách thức:
  - Mỗi đám mây cách nhau theo trục X một đoạn 20 đơn vị, cách nhau theo trục Y và Z một đoạn ngẫu nhiên từ 0 đến 10.
  - Xoay các đám mây theo trục Y và Z theo một góc ngẫu nhiên.
  - Scale mỗi khối lập phương theo một tỉ lệ ngẫu nhiên. Việc tạo các số ngẫu nhiên giúp đám mây trông chân thực hơn.



Hình 25: Mô phỏng một đám mây.

- Mỗi khối lập phương được tạo từ BoxGeometry và MeshPhongMaterial

```
createCloud(cloudGeometry, cloudMaterial, nBlocs) {
  // Tạo đám mây
  for(let i = 0; i < nBlocs; i++){
    let m = new Mesh(cloudGeometry, cloudMaterial);
    //Set vị trí và xoay cho mỗi cube
    m.position.x = i * 20;
    m.position.y = Math.random() * 10;
    m.position.z = Math.random() * 10;
    m.rotation.z = Math.random() * Math.PI * 2;
    m.rotation.y =  Math.random() * Math.PI * 2;
    //Set size cho cube
    let s = 0.1 + Math.random() * 1.7;
    m.scale.set(s, s, s);
    // Đổ bóng
    m.castShadow = true;
    m.receiveShadow = true;
    // Thêm cube vào group
    this.mesh.add(m);
    this.listUse.push(m);
  }
```

Hình 26: Code tạo đám mây.

```
createClouds(nClouds){
  for(let i = 0; i < nClouds; i++){
    let cloud = new Cloud();
    let a = Math.PI * 2 / nClouds * i;
    // Khoảng cách từ tâm tới đám mây
    let h = 1600 + Math.random() * 100;
    // Vị trí đám mây
    cloud.setPosition(Math.cos(a) * h,
                      Math.sin(a) * h + 100, - 400 - Math.random() * 400)
    // Xoay đám mây hướng vào tâm
    cloud.mesh.rotation.z = a + Math.PI / 2;
    // Set độ sâu của đám mây
    let s = 0.5 + Math.random() * 2;
    cloud.mesh.scale.set(s, s, s);
    this.mesh.add(cloud.mesh);
    this.listUse.push(cloud);
  }
}
```

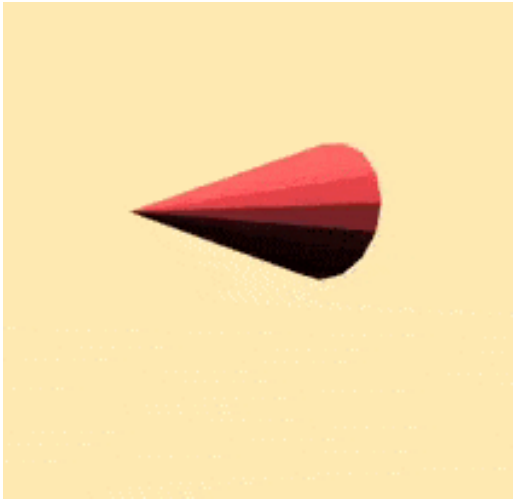
Hình 27: Code tạo bầu trời.

- Đồng xu:

- Mỗi đồng xu được tạo từ TorusGeometry và MeshPhongMaterial.
- Số đồng xu trong một chuỗi là một số ngẫu nhiên từ 1 đến 10.
- Xác suất mỗi chuỗi xuất hiện ở mỗi lần loop là 2%.
- Một chuỗi bắt đầu ở vị trí 0^o và kết thúc ở vị trí 180^o.

Hình 28: Một chuỗi đồng xu.

- Vật cản:
  - Mỗi vật cản được tạo từ CylinderGeometry và MeshPhongMaterial.
  - Vật cản sẽ xuất hiện càng nhiều theo thời gian chơi.
  - Giống như đồng xu, vật cản sẽ biến mất khi ở 180^o.

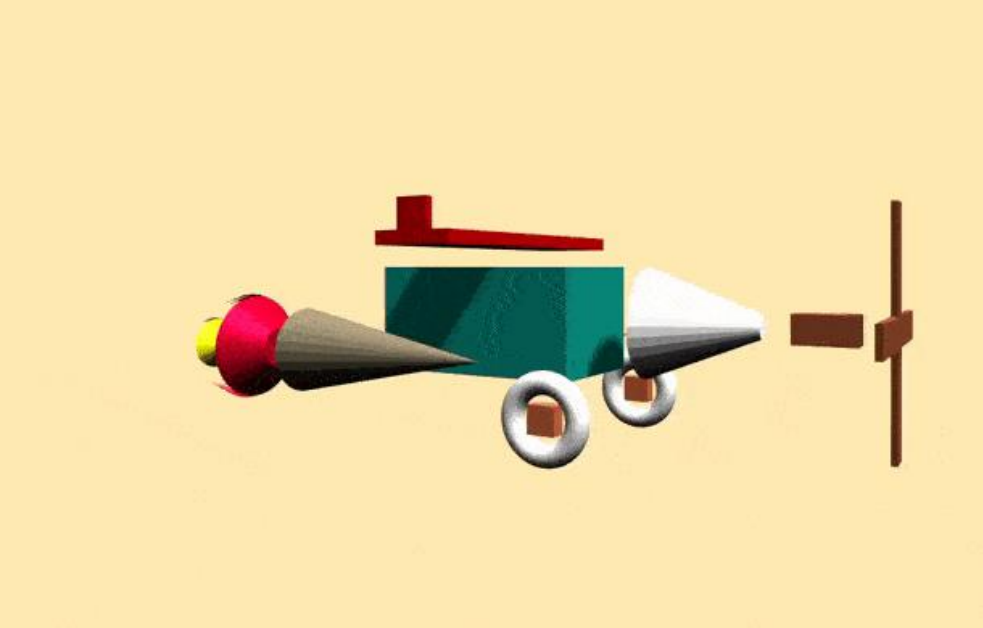


Hình 29: Mô phỏng vật cản.

- Máy bay và phi công:
    - Máy bay:

Máy bay sẽ gồm các phần sau như:

- Đầu máy bay
- Thân máy bay
- Cánh quạt
- Cánh máy bay
- Đuôi máy bay
- Bánh xe
- Bộ phận đẩy



Hình 30: Máy bay.

- Phi công:

Phi công sẽ gồm các phần sau như:

- Đầu
- Tóc

- Thân
- Kính
- Tay
- Tai

Hình 31: Phi công.

i. Chuyển động liên tục

Trong Threejs, nếu muốn thực hiện cảnh động, chúng ta cần render cảnh nhiều lần theo một tần suất nào đó. Javascript hiện nay có một giải pháp là hàm requestAnimationFrame.

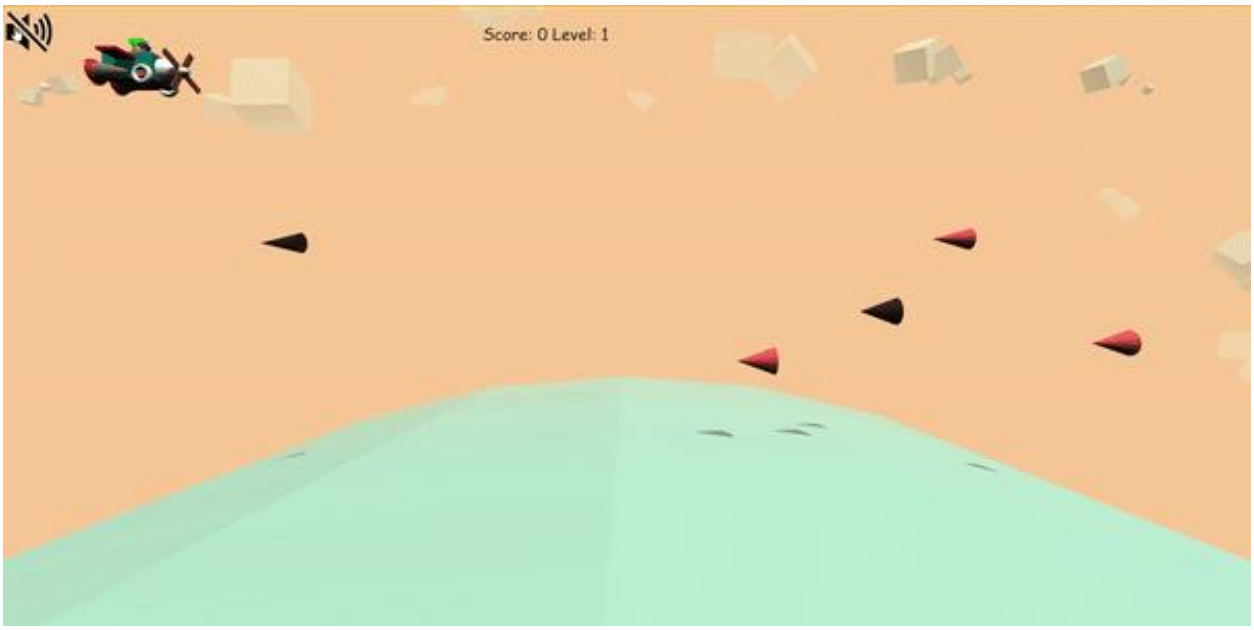
- The **window.requestAnimationFrame()** thông với trình duyệt rằng muốn thực hiện cảnh động và yêu cầu trình duyệt gọi một hàm chỉ định để cập nhật liên tục.
- Tần suất gọi hàm sẽ phụ thuộc vào trình duyệt, thường là 60 lần trên 1s.
- Trong hàm render, chúng ta cần gọi hàm requesAnimationFrame với tham số là phương thức render.
- Trong hàm render gọi hàm update (hàm thực hiện cập nhật).

```
update() {
    // Cập nhật lại các đối tượng ở đây
}

render() {
    this.update();
    this.renderer.render(this.scene, this.camera);
    requestAnimationFrame(this.render.bind(this));
}
```

Hình 32: Code thực hiện cảnh động trong three.js.

- Kết quả khi thực hiện cảnh động trong game bằng Thee.js:



Hình 33: Cảnh động trong Hyper Raider Game.

j. Di chuyển máy bay

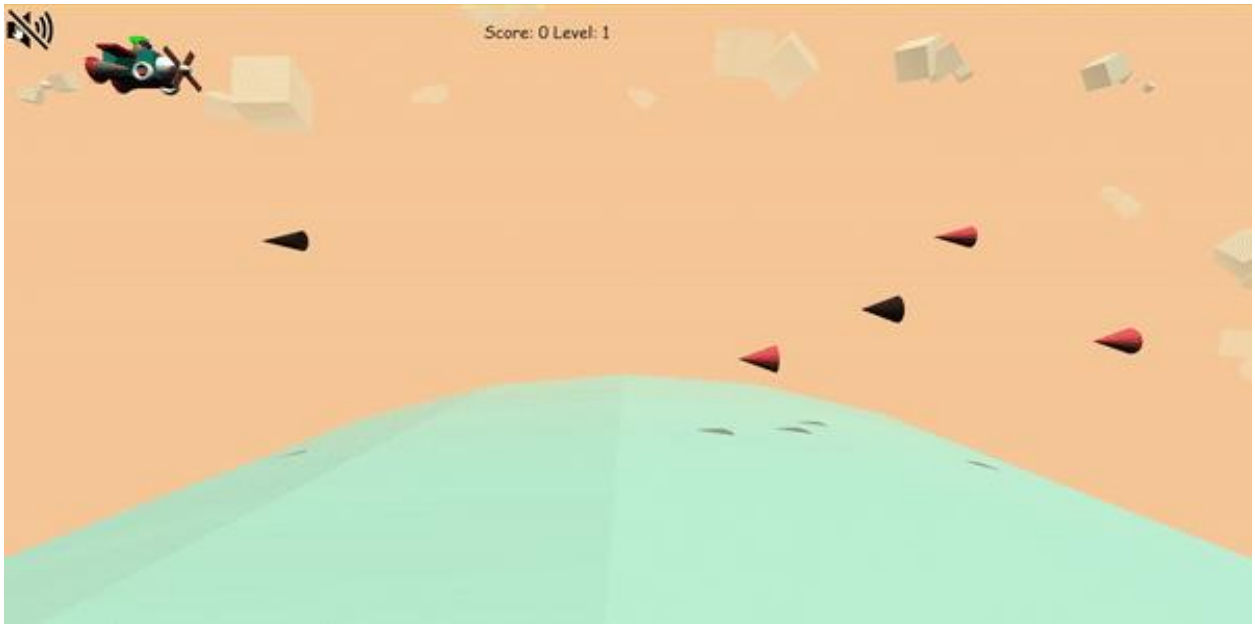
Máy bay sẽ di chuyển theo vị trí của con chuột đã được chuẩn hóa. Tức là với mỗi vị trí con chuột ta chuẩn hóa nó để làm chuyển động của máy bay trông mượt mà hơn (chi tiết có thể xem ở phần code).

```
document.addEventListener("mousemove", (event) => {
  const width = canvas.clientWidth;
  const height = canvas.clientHeight;
  var tx = -1 + (event.clientX / width) * 2;
  var ty = 1 - (event.clientY / height) * 2;
  mousePos = { x: tx, y: ty };
  this.plane.updatePlane(mousePos);
});
```

Hình 34: Code thực hiện cập nhật vị trí máy bay.

3. Kết quả

- Link demo: https://j2team.dev/go/hyper_raider_game
- Link github: https://github.com/htdung167/CS105.M21
- Một số hình ảnh của Hyper Raider Game:



Hình 35: Hyper Raider Game.

III. KẾT LUẬN

Ngày nay, 3D là lĩnh vực tiềm năng được nhiều nhà nghiên cứu quan tâm. Việc mô hình hóa vật thể đã hỗ trợ con người trong rất nhiều lĩnh vực như bất động sản, nội thất, y học, vũ trụ, giải trí,...

Trong đồ án môn học này, chúng tôi đã đưa ra các khái niệm cơ bản của Three.js từ cách tiếp cận từ thấp nhất. Qua đồ án này, chúng tôi đã biết và hiểu nhiều kiến thức về cấu tạo một mô hình 3D, cách cài đặt một mô hình 3D, kiến

thức về camera, ánh sáng,… Chúng tôi cần phát triển và hoàn thiện trò chơi hơn để Hyper Raider Game có thể tiếp cận tới nhiều người dùng.

IV. HƯỚNG PHÁT TRIỂN

- Chúng tôi sẽ có phát triển nhiều mẫu máy bay cũng như background hơn để người chơi có thể tùy chọn.
- Gameplay sẽ đa dạng hơn, thêm các chỉ số năng lượng cho máy bay để bắt buộc người chơi ăn điểm để có thể tiếp tục chơi tiếp cũng như nếu va chạm với vật cản cũng sẽ không gameover ngay mà chỉ làm tiêu hao năng lượng. Gameover khi thanh năng lượng về 0.
- Cải thiện chất lượng hình ảnh của các vật thể trong game.
- Phát triển phần back-end, kết nối với cơ sở dữ liệu để người chơi có thể xem xếp hạng của mình so với người chơi khác.

V. TÀI LIỆU THAM KHẢO

[1] https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene

[2] https://viblo.asia/s/hoc-threejs-co-ban-Q75wqJJ9ZWb

[3] https://tympanus.net/codrops/2016/04/26/the-aviator-animating-basic-3d-scene-threejs/

[4] https://viblo.asia/p/tao-1-canh-dong-3d-don-gian-tren-web-voi-threejs-BYjv449xvxpV?fbclid=IwAR3DUv9DjOIH4l7sMnRmBqr55Sffea3Rt5jPUNMRVv6nAcT5zL2a3Pd62Fk#_sap-xong-13

VI. BẢNG PHÂN CÔNG CÔNG VIỆC

THÀNH VIÊN	CÔNG VIỆC	ĐÁNH GIÁ
Hoàng Tiến Dũng	- Tìm hiểu cách sử dụng thư viện Three.js - Xây dựng các vật thể: bầu trời, đồng xu, bóng. - Xây dựng phần âm thanh cho game, tăng level. - Tham gia thiết kế slide và viết report. - Tìm hiểu cách deploy sản phẩm.	Hoàn thành tốt

Nguyễn Thành Trọng	- Tìm hiểu cách sử dụng thư viện Three.js - Xây dựng các vật thể: biển, vật cản, máy bay và phi công - Cài đặt cách tính điểm và hiện điểm - Tham gia thiết kế slide và viết report. - Tìm hiểu cách deploy sản phẩm.	Hoàn thành tốt