**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**
**TRƯỜNG ĐẠI HỌC SÀI GÒN**
**KHOA CÔNG NGHỆ THÔNG TIN**

# TIỂU LUẬN

## ***ĐỀ TÀI:*** TÌM HIỂU CÔNG CỤ UNITY VÀ XÂY DỰNG GAME 2D

## MÔN: CÁC CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI

**THÀNH VIÊN THỰC HIỆN:**

1. Nguyễn Tuấn Anh — MSSV: 3118410013
2. Tăng Chí Chung — MSSV: 3118410043
3. Nguyễn Ngọc Tiến Em — MSSV: 3118410094

**Thành Phố Hồ Chí Minh, tháng 4 năm 2022**

# MỤC LỤC

**MỤC LỤC** **1**

**MỞ ĐẦU** **2**

**CHƯƠNG 1: Giới thiệu về Unity** **3**

1. Giới thiệu 3
2. Tổng quan 3
3. Các mức phí liên quan 4
4. Lịch sử phát triển 5
5. Hướng tiếp cận 5
6. Ưu điểm 5
7. Nhược điểm 6
8. Unity Teams 7
9. Các công trình liên quan 7

**CHƯƠNG 2: Các thành phần cơ bản và demo** **8**

1. Các thành phần cơ bản của Unity: 8
2. Demo: 9
   - Example 1: Light in game 9
   - Example 2: Chuyển scene 10
   - Example 3: Tương tác mở rương 11
   - Example 4: Prefab 13
   - Example 5: Animation Controller 13
   - Example 6: Magic Title Game 15
3. Các phần nâng cao 18
   - a. Object Pooling: 18
   - b. Attack System with correct frame : 18
   - c. OOP system: 19
   - d. Health bar: 20
   - e. Death Effect: 20
   - f. Raycast: 21
   - g. LineRenderer: 21
   - h. Gacha System: 22
   - i. ScriptObject: 23
   - j. Boss 24
   - k. Character 27
   - l. Level Design 28

**Tài liệu tham khảo** **29**

# MỞ ĐẦU

Ngành công nghiệp game chứng kiến sự đi lên mạnh mẽ trong nhiều năm qua khi mà không ít những studio mới nổi đã có được danh tiếng nhờ những sản phẩm indie khác biệt so với phần còn lại. Điều này là do sự tiến bộ không ngừng của các engine làm game, cũng như quan niệm nhìn nhận game có phần tiến bộ hơn trước. Rất nhiều game engine đã vượt qua chính mục đích tạo ra nó là làm game mà trở thành những công cụ hỗ trợ đắc lực trong nhiều lĩnh vực khác như phim ảnh và kiến trúc. Bằng mối quan tâm sâu sắc đến ứng dụng của các engine chúng tôi lựa chọn đề tài:" TÌM HIỂU CÔNG CỤ UNITY VÀ XÂY DỰNG GAME 2D" để thu được những kiến thức nền tảng cũng như cách sử dụng một engine phổ biến như unity.

# CHƯƠNG 1: Giới thiệu về Unity

## 1. Giới thiệu

Unity là một game engine đa nền tảng được phát triển bởi Unity Technologies ,mà chủ yếu để phát triển video game cho máy tính, consoles và điện thoại. Lần đầu tiên nó được công bố chạy trên hệ điều hành OS X, tại Apple's Worldwide Developers Conference vào năm 2005, đến nay đã mở rộng 27 nền tảng.

## 2. Tổng quan

Unity hỗ trợ đồ họa 2D và 3D, các chức năng được viết chủ yếu qua ngôn ngữ C#.

Hai ngôn ngữ lập trình khác cũng được hỗ trợ: Boo, đã bị loại cùng với việc phát triển Unity 5_ và UnityScript bị loại vào tháng 8 năm 2017 sau khi phát hành Unity 2017.1. UnityScript là một ngôn ngữ lập trình độc quyền có cú pháp tương tự JavaScript.

Phần mềm nhắm mục tiêu các đồ họa APIs sau: Direct3D trên Windows và Xbox One; OpenGL trên Linux, macOS, và Windows; OpenGL ES trên Android và iOS; WebGL trên web; và APIs độc quyền trên các máy chơi video game. Ngoài ra, Unity hỗ trợ APIs cấp thấp như Metal trên iOS và macOS và Vulkan trên Android, Linux, và Windows, cũng như Direct3D 12 trên Windows và Xbox One.

Trong 2D games, Unity cho phép nhập sprites và một renderer thế giới 2D tiên tiến. Đối với 3D games, Unity cho phép thiết lập các đập điểm kỹ thuật của các kết cấu và độ phân giải mà công cụ trò chơi hỗ trợ, cung cấp các hỗ trợ cho bump mapping, reflection mapping, parallax mapping, cảnh không gian ambient occlusion (SSAO ), hiệu ứng bóng đổ bằng cách sử dụng shadow maps, render thiết lập toàn cảnh đến hiệu ứng.Unity cũng cung cấp các dịch vụ cho nhà phát triển, bao gồm: Unity Ads, Unity Analytics, Unity Certification, Unity Cloud Build, Unity Everyplay, Unity API, Unity Multiplayer, Unity Performance Reporting and Unity Collaborate.

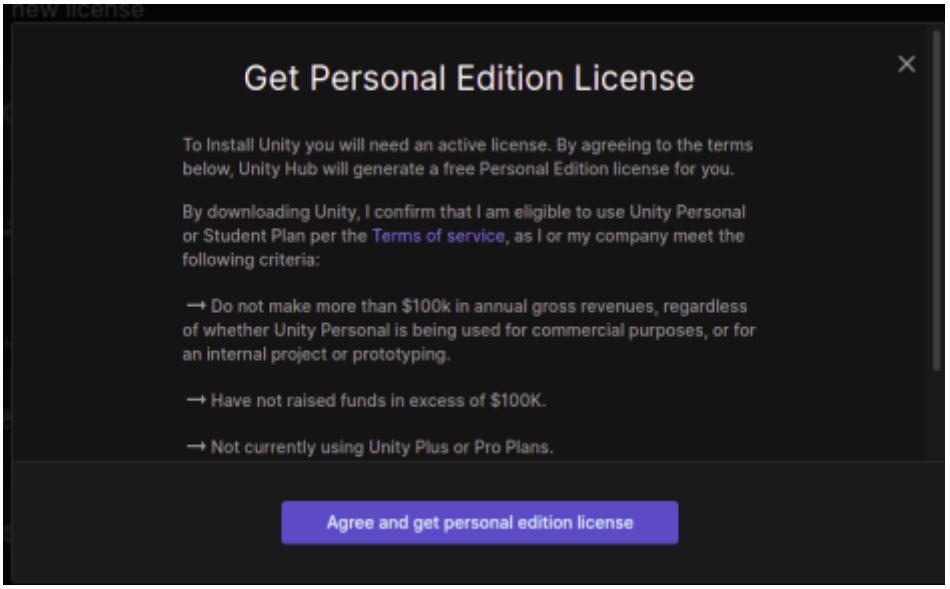
### 3. Các mức phí liên quan

Unity có 4 sự lựa chọn đối với người dùng. [1]Đây là danh sách các lựa chọn hiện có:

| Giấy phép | Hỗ Trợ Mọi Chức Năng Và Thiết Bị | Splash Screen | Hàng Đợi Tạo Đám Mây | Đa Người Chơi | Năng Lực Doanh Thu | Báo Cáo Hiệu Năng | Mã Nguồn Và Hỗ Trợ Cao Cấp | Giá Định Kỳ |
|---|---|---|---|---|---|---|---|---|
| Personal | Có | Made With Unity | Standard | 20 người | $100K | Không | Không | Miễn Phí |
| Plus | Có | Tùy Chỉnh Hoạt Ảnh Hoặc Không | Priority | 50 người | $200K | Có | Không | 35$ / tháng |
| Pro | Có | Tùy Chỉnh Hoạt Ảnh Hoặc Không | Concurrent Builds | 200 người | Unlimited | Có | Có | $125 / tháng |
| Enterprise | Có | Tùy Chỉnh Hoạt Ảnh Hoặc Không | Dedicated Build Agents | Tùy Chỉnh Số Người | Unlimited | Có | Có | Thương Lượng |

Bảng 1.1 Các giấy phép sử dụng

Ngoài ra khi sản phẩm có doanh thu trên 100.000 đô thì phải mua bản Pro version và trả thêm phụ phí.




Hình 1. Mức phí chi trả khi sử dụng Unity.

## 4. Lịch sử phát triển

Unity game engine khởi động năm 2005, nhằm "dân chủ hóa" phát triển game bằng cách cung cấp cho nhiều nhà phát triển hơn. Năm tiếp theo, Unity được vinh danh là á quân trong hạng mục Best Use of Mac OS X Graphics tại Apple Design Awards của Apple Inc.Unity ban đầu được phát hành cho Mac OS X, sau đó bổ sung hỗ trợ cho Microsoft Windows và trình duyệt Web.

Unity 2.0 ra mắt năm 2007 với khoảng 50 tính năng mới. Bản phát hành bao gồm một công cụ địa hình được tối ưu hóa cho môi trường 3D, real-time dynamic shadows, directional lights và spotlights, phát lại video và các tính năng khác. Bản phát hành cũng bổ sung các tính năng nhờ đó các nhà phát triển có thể cộng tác dễ dàng hơn.Nó bao gồm một Networking Layer để các nhà phát triển tạo game nhiều người chơi dựa trên User Datagram Protocol, cung cấp Network Address Translation, State Synchronization, và Remote Procedure Calls.

Unity 3.0 ra mắt tháng 9 năm 2010 với tính năng mở rộng các tính năng đồ họa của engine cho máy tính để bàn và video game consoles.Ngoài hỗ trợ Android, Unity 3 còn có tính năng tích hợp công cụ Beast Lightmap của Illuminate Labs, deferred rendering, một built-in tree editor, kết xuất phông chữ gốc, ánh xạ UV tự động và bộ lọc âm thanh, cùng nhiều thứ khác.

Tháng 11 năm 2012, Unity Technologies phát hành Unity 4.0. Phiên bản này bổ sung các hỗ trợ cho DirectX 11 và Adobe Flash, các công cụ hoạt ảnh mới có tên Mecanim, và quyền truy cập vào bản xem trước Linux.

## 5. Hướng tiếp cận

- Nhóm tiếp cận với Unity thay vì các công cụ khác như Unreal, Gamemaker, Renpy… một phần là vì tính tiện dụng của các công cụ thiết kế, cũng như cộng đồng sử dụng Unity ở Việt Nam đặc biệt phát triển.
- Ngoài ra công cụ Unity còn hỗ trợ nhiều hệ điều hành như Window hay Linux.
- Unity cũng hỗ trợ các chuẩn đầu ra như exe, webgl, android.

## 6. Ưu điểm

Unity có rất nhiều tính năng tuyệt vời:

- Unity có một cộng đồng rất lớn về asset và plugin – trong đó có rất nhiều resources free và có nhiều thứ rất đáng bỏ tiền
- Tính dễ sử dụng của Unity cho người mới bắt đầu đã tiến thêm một bước nữa khi công ty thông báo vào tháng 7 năm 2020 rằng công cụ tạo kịch bản trực quan Bolt hiện sẽ được đưa vào tất cả các kế hoạch của Unity trong tương lai mà không phải trả thêm phí.
- Unity có bộ công cụ rất trực quan và editor có thể mở rộng bằng plugins

- Unity hỗ trợ rất nhiều định dạng asset khác nhau và có thể tự động chuyển đổi đến định dạng phù hợp nhất với nền tảng thích hợp
- Unity hỗ trợ nhiều nền tảng: di động, desktop, web và console
- Việc triển khai đến các nền tảng khác nhau cũng khá dễ quản lý
- Bạn có thể dễ dàng xây dựng một game 3D mà không cần cấu hình quá phức tạp
- Unity bản free có hầu hết những tính năng quan trọng nhất
- Unity bản trả phí phù hợp với các developer chuyên nghiệp
- Unity cho phép bạn xây dựng các công cụ của riêng mình
- Unity rất tốt cho các nhà phát triển VR

## 7. Nhược điểm

Tuy nhiên Unity cũng có vài nhược điểm mà bạn cần cân nhắc:

- Việc hợp tác rất khó khăn. Unity sử dụng một server asset rất hiệu quả để hỗ trợ các đội phát triển phần mềm hợp tác với nhau. Tuy nhiên nếu bạn không sử dụng nó thì việc chia sẻ code và asset giữa các thành viên trong team có thể gây ra những vấn đề nghiêm trọng. Lựa chọn tốt nhất là sử dụng một số công cụ quản lý resource bên ngoài nhưng có một vài binary file không thể merge được với nhau và việc cập nhật asset có thể gây nên một số vấn đề trong scenes, mất kết nối đến script và các đối tượng khác
- Hiệu năng chưa thật sự ấn tượng cho đến khi Unity 5 ra mắt. Unity 5 đã chạy hầu hết trên một luồng duy nhất và hầu như không sử dụng thêm 1 nhân phụ nào trên các thiết bị di động. Bộ biên dịch chưa được tối ưu tốt cho các bộ xử lý ARM trên hầu hết các thiết bị di động. Để giải quyết vấn đề này thì Unity đã quyết định transpile (source-to-source compiler) sang C++ và sử dụng LLVM để tối ưu được nhiều hơn thay vì giải quyết vấn đề này trực tiếp trên các phiên bản sau này
- Mã nguồn của engine không được công bố kể cả cho những người dùng chấp nhận trả tiền. Điều đó có nghĩa là nếu bạn gặp một bug với engine bạn phải chờ Unity fix chúng trong các bản tiếp theo. Điều này có thể gây nên những vấn đề nghiêm trọng với project của bạn
- Unity không phù hợp với các dự án lớn như những tựa game đình đám AAA.

## 8. Unity Teams

Bao gồm 25 GB bộ nhớ đám mây, các bản dựng tự động và chỗ cho toàn bộ nhóm của bạn. 30 ngày miễn phí. Sau đó, $9 mỗi tháng cho các nhóm có quy mô tối đa 3 người. $7 cho mỗi thành viên nhóm bổ sung mỗi tháng.

| | Basic | Advanced |
|---|---|---|
| Cloud storage | 1GB | 25GB |
| Included team seats | 3 | 3 |
| Version History | 90 day | unlimited |
| Integrations | 1 | unlimited |

Các gói mua thêm:

- $5 cho mỗi 25GB lưu trữ mỗi tháng
- $7 cho mỗi người tham gia mỗi tháng

## 9. Các công trình liên quan

- Các tựa game nổi tiếng không sử dụng code mà nhờ vào công cụ hỗ trợ là Playmaker hay bolt để làm ra như : Gris, The First Tree.
- Các tựa game 2D, 3D được cộng đồng game thủ quan tâm : CupHead, Genshin Impact, Among Us.
- Kể cả ở Việt Nam cũng có những tựa game từ Unity đến từ công ty Amanotes : Magic Titles 3, Beat Blader 3D và 111dots Studio : Free Fire và những công ty khác làm ra những tựa game trên mobile.

# CHƯƠNG 2: Các thành phần cơ bản và demo

## 1. Các thành phần cơ bản của Unity:

Scene là thành phần cơ bản nhất của Unity, scene là khoảng không gian cho phép các object được hoạt động và tương tác ở trong đó

Dưới scene là các object là thành phần chính mô tả gameplay, các object không chỉ là các vật thể hữu hình mà còn là các khối chứa nhiều thứ như ánh sáng, âm thanh, các lệnh. Mọi sự tồn tại trong scene đều cần một object đại diện. Một object tồn tại trong scene đều có 2 trạng thái là active và deactive. Trạng thái deactive cũng dừng các script mà object đang chạy.

Chứa trong các Object là các component. Component giữ vai trò quan trọng trong việc thể hiện đặc tính của object. Đơn cử như component Spriterenderer giữ vai trò thể hiện hình thái của object nếu component này bị tắt đi thì vật thể sẽ không có hình dạng

Thấp nhất trong các thành phần của game là các mã lệnh hay script, hầu hết các component trong game đều có các api tương ứng để thay đổi trong quá trình vận hành game. Script thể hiện sự tương tác giữa các object thông qua xử lý component.

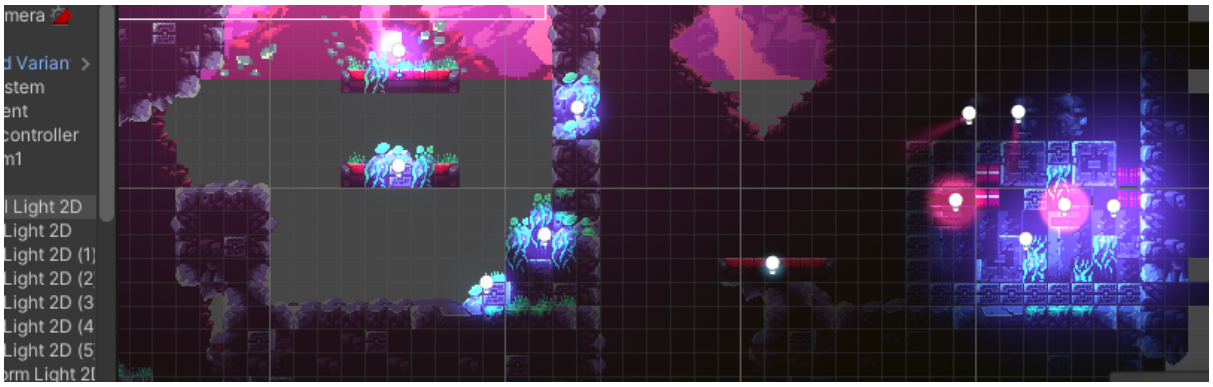
## 2. Demo:

**Example 1: Light in game**

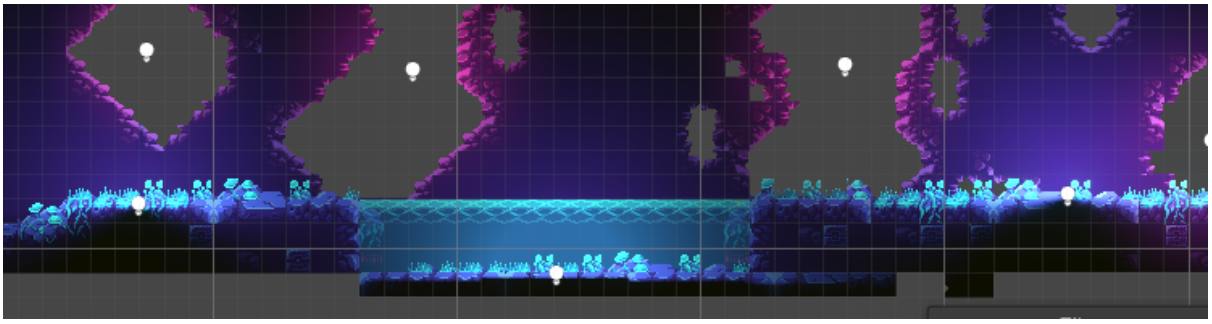
Tổng quan : Chúng ta làm về ánh sáng

Có các loại ánh sáng như: point light, free form light,sprite light,..

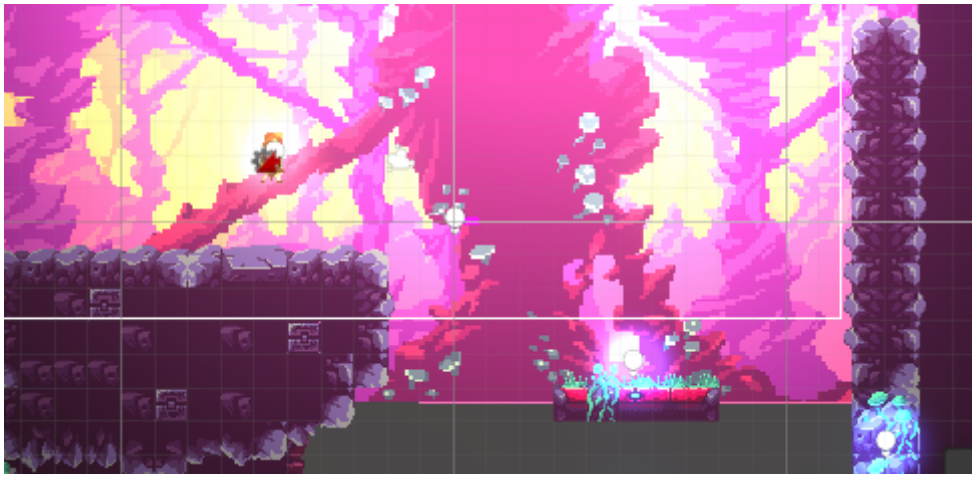
Các hình ảnh demo:




Hình 2. Demo ánh sáng trong Unity 1.



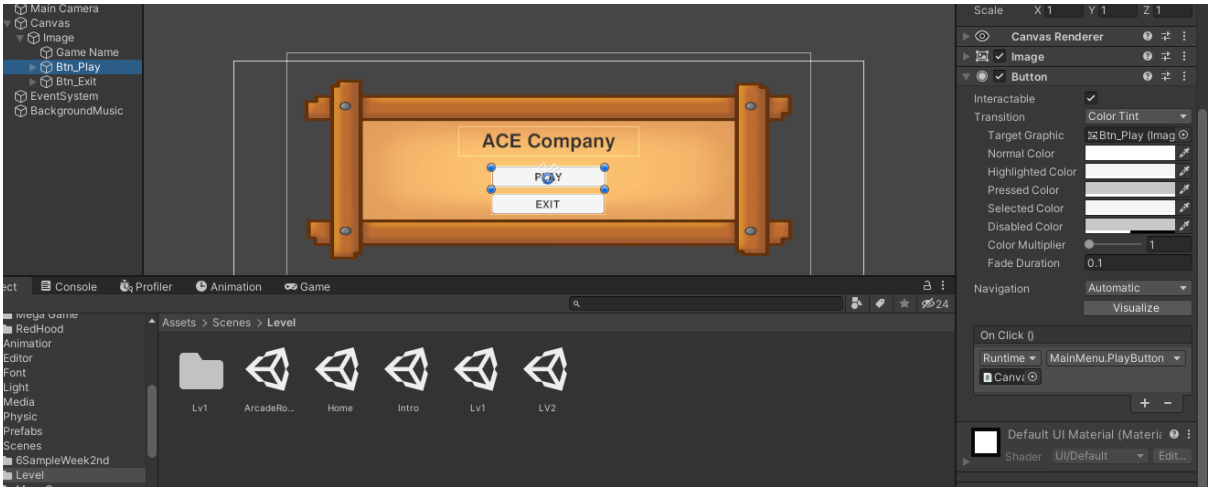
Hình 3. Demo ánh sáng trong Unity 2.



Hình 4. Demo ánh sáng trong Unity 3.

**Example 2: Chuyển scene**

Ta sẽ tạo script để bắt sự kiện khi người chơi nhấn vào các nút chuyển scene thì nó sẽ chuyển theo scene mà ta mong muốn, ta có thể tùy chỉnh hướng đi của scene :

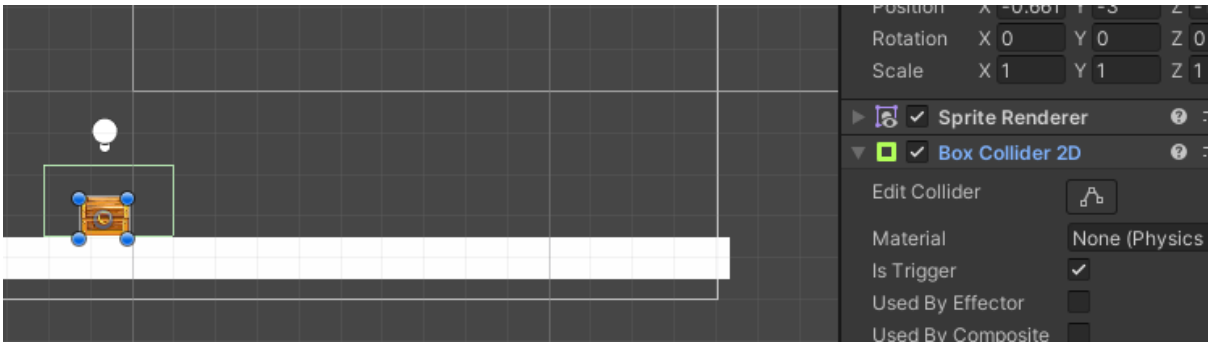



Hình 5. Demo chuyển scene.

**Example 3: Tương tác mở rương**

Tổng quan: khi ta tới gần rương và bấm phím E, nó sẽ mở.

Trong object rương ta sẽ tạo một collider 2D kèm theo trigger để nhận phím nó có chạm vào nhân vật hay không.




Hình 6. Object rương kèm Collider để tương tác.

Sau đó ta kèm theo đoạn script để khi người chơi bấm phím E thì rương có hoạt ảnh mở và hiển thị ra các vật phẩm bên trong.


Hình 7. Sự kiện mở rương.

Các trường hợp ngoại lệ khi người chơi bấm phím E lại hay rời khỏi rương thì rương sẽ tự đóng lại kèm theo hoạt ảnh đóng rương. Với cơ chế như sau:

- Rương chỉ kích hoạt trạng thái đóng và mở khi và chỉ khi người chơi trong tầm tương tác của rương.
- Nếu người chơi nhấn nút chỉ định, ở đây chúng ta sẽ set là nút E thì rương sẽ đổi trạng thái từ mở sang đóng và ngược lại.
- Nếu người chơi đi ra khỏi tầm kiểm soát của nó - Collider thì rương sẽ luôn luôn đóng lại.

Code Snippet in ChestController.cs

```
1   private void Update()
2       {
3           if (_isEnable)
4           {
5               if (Input.GetKeyDown(KeyCode.E))
6               {
7                   // change value before display
8                   _isOpen = !_isOpen;
9                   chestLogic();
10              }
11          }
12      }
13
14      void chestLogic()
15      {
16          if (_isOpen)
17          {
18              Debug.Log("Player opening the chest");
19          }
20          else
21          {
22              Debug.Log("Chest closed");
23          }
24          invertoryChestMenu.gameObject.SetActive(_isOpen);
25          anim.SetBool("IsOpen", _isOpen);
26      }
27
28      void chestLogic(bool flag)
29      {
30          _isOpen = flag;
31          chestLogic();
32      }
33
34      private void OnTriggerEnter2D(Collider2D other)
35      {
36          if (other.tag == "Player")
37          {
38              Debug.Log("Player eneter chest's zone");
39              _isEnable = true;
40          }
41      }
42
43      private void OnTriggerExit2D(Collider2D other)
44      {
```

```
45          if (other.tag == "Player")
46          {
47              Debug.Log("Player leave chest's zone");
48
49              _isEnable = false;
50              chestLogic(false);
51          }
52      }
```

**Example 4: Prefab**

Tạo một prefab player có thể di chuyển trong bất kì scene nào, nó có thể dùng lại nhiều lần.

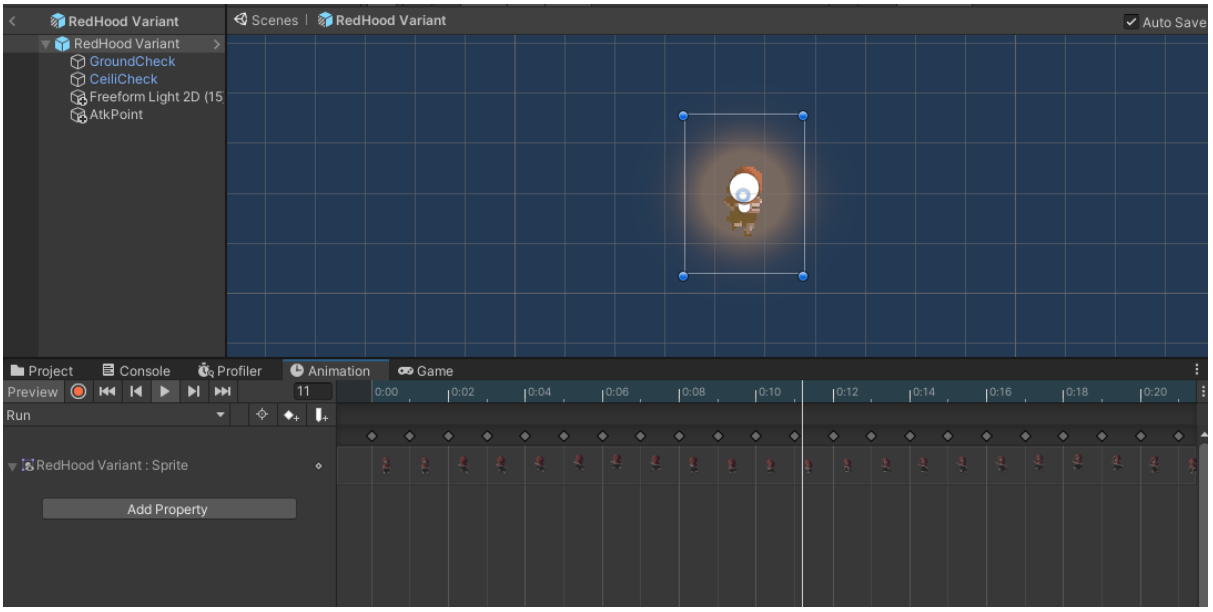

Hình 8. prefab player.

Ta tạo 1 prefab với tên là Red Hood tượng trưng cho Player, nó sẽ bao gồm: tag, collider, rigidbody, Sprite và các script di chuyển.

**Example 5: Animation Controller**

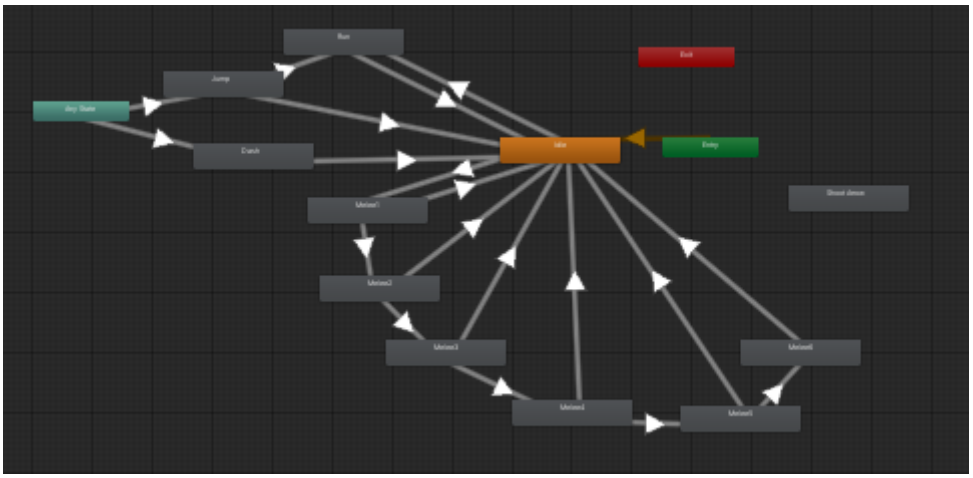
Một animation của một đối tượng, thực thể bao gồm nhiều animator của đối tượng, thực thể đó như: hoạt ảnh di chuyển, nhảy, tấn công, phòng thủ, …. .

Giao diện quản lý animation của một object:

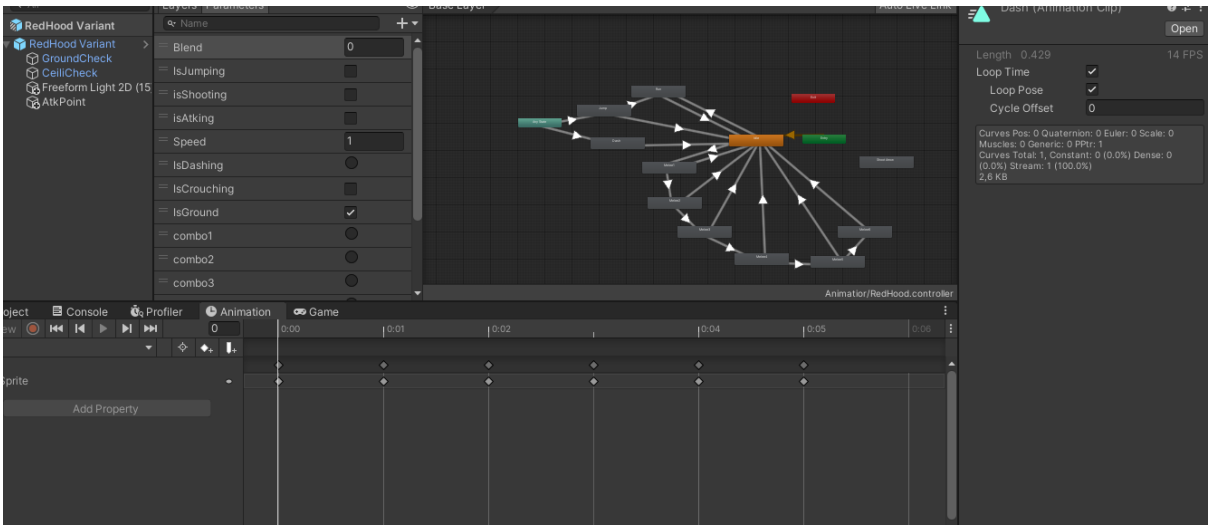



Hình 9. Giao diện quản lý animation của Unity.

Giao diện quản lý các animator trong animation bao gồm các thành phần di chuyển, nhảy, đánh, và đứng im:



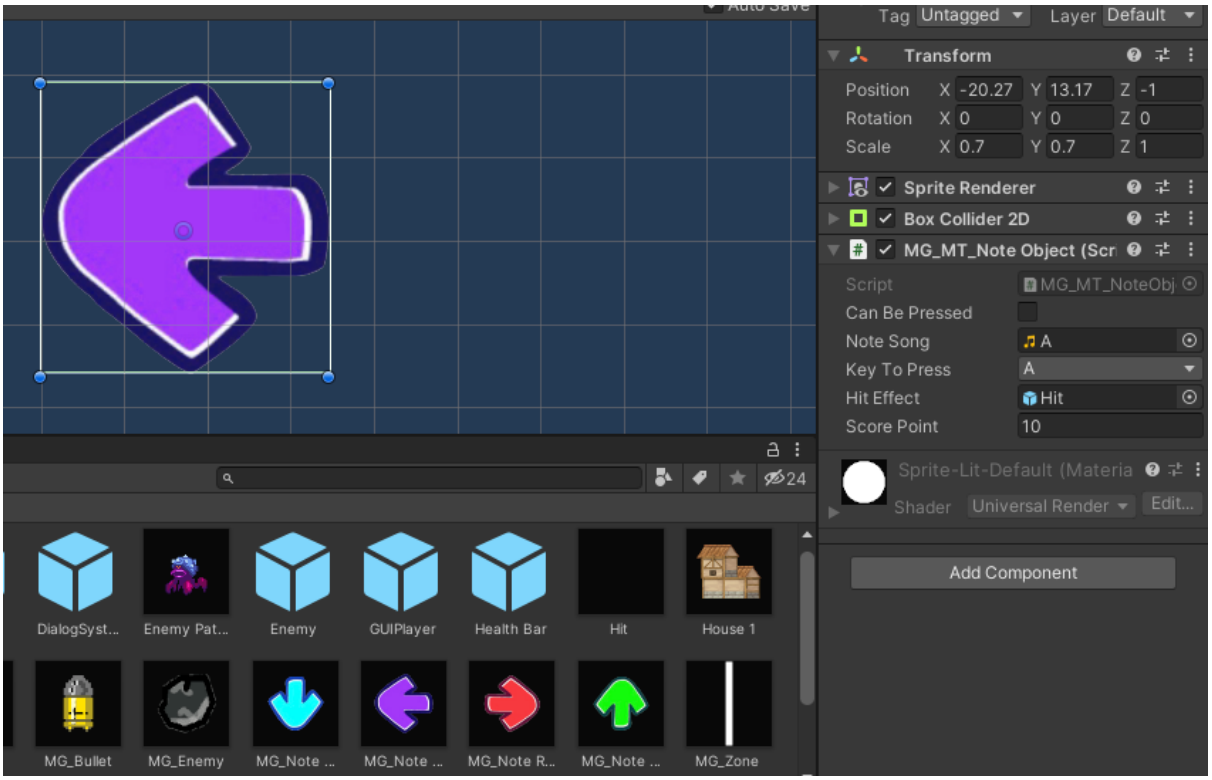
Hình 10. Giao diện quản lý Animation của Unity.




Hình 11. Giao diện quản lý các giá trị của Animation.

**Example 6: Magic Title Game**

Làm về game magic title, là một tựa game nhấn nút theo vị trí thanh nhạc rơi xuống. Đầu tiên, ta tạo ra các script cho prefab nốt nhạc. Bao gồm các thông tin như keypress, score point cho nó. Sau đó ta thêm hình ảnh tương ứng cho từng nốt: trên dưới trái phải.

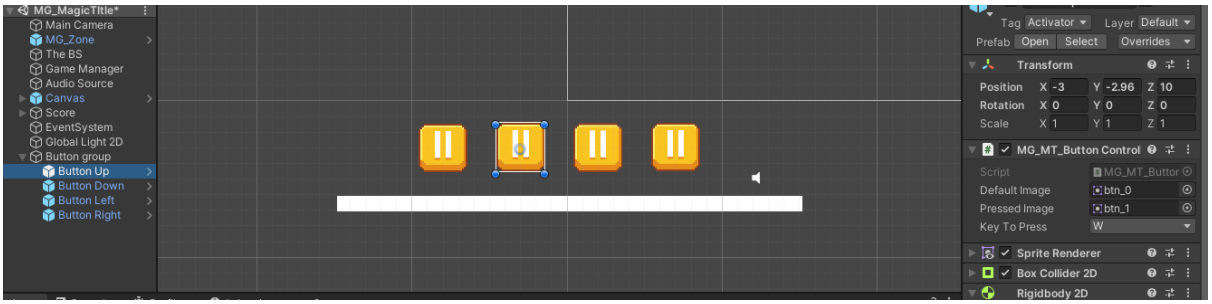



Hình 12. nốt bên trái trong prefab

Sau đó ta sẽ tạo ra một object đề quản lý tất cả các nốt bằng cách cho random nốt theo hướng ở một vị trí nhất định. Trong đó, có các thông tin kèm theo như là thời gian di chuyển của các nốt, thời gian đợi đề spawn 1 nốt khác.

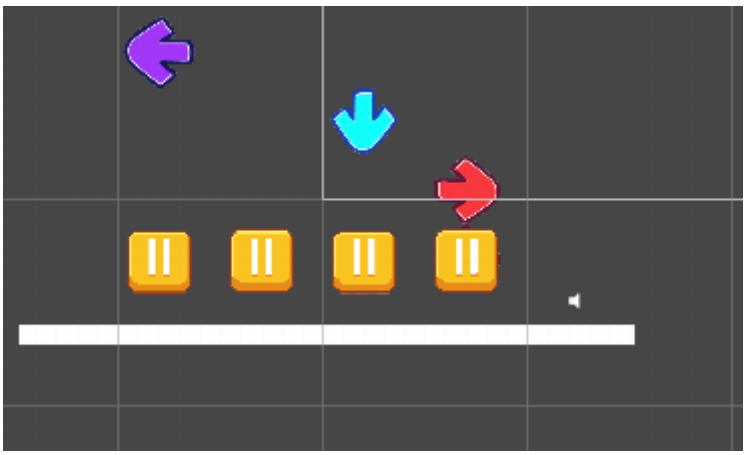
```
int pos = Random.Range(0, 4);

Debug.Log("Create new note at range : " + pos);

var temp = Instantiate(_prefab[pos], new Vector3(-8 + (pos * 5), 10, 0), Quaternion.identity);
temp.transform.parent = gameObject.transform;
```

Kế tiếp ta tạo ra các button để có thể nhấn được nốt tương ứng, nó sẽ bắt sự kiện bàn phím để so sánh xem có nốt nào gần đó hay không mà kích hoạt.




Hình 13. Giao diện trò chơi phiên bản edit game.

Sau đó, khi các nốt không nhấn được sẽ bị xóa khỏi màn hình để giảm dung lượng cho game. Bằng cách tạo ra một zone, khi nốt chạm vào nó sẽ biến mất. Khi các nốt chạm vào thanh màu trắng sẽ biến mất.

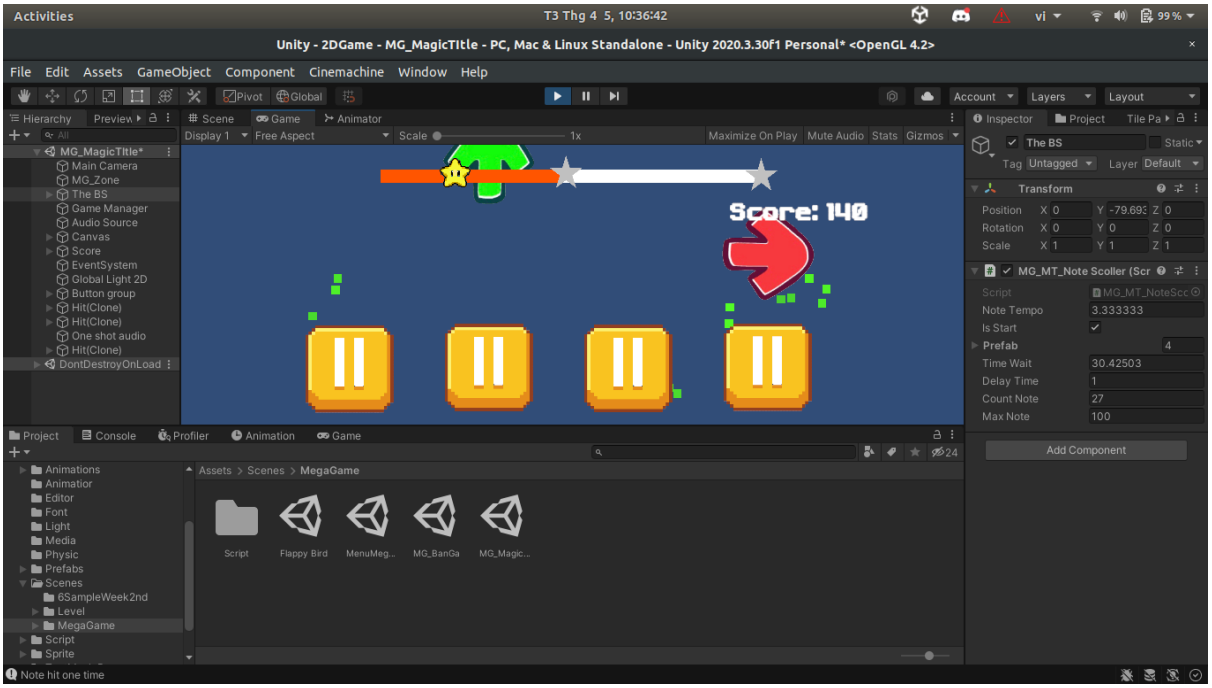


Hình 14. Giao diện các nốt và nút nhấn trong game.

Cuối cùng ta sẽ tạo 1 thanh slide để hiển thị score cho người chơi có thể thấy được mình được bao nhiêu điểm trong một màn chơi.


Hình 15. Giao diện thanh scoreboard trong Unity.

Demo:




Hình 16. Giao diện khi chơi game Magic Title.

## 3. Các phần nâng cao

**a. Object Pooling:**

Một design pattern phổ biến trong mọi quy trình thiết kế ứng dụng. Trong đó quy định rằng một object được tạo ra sẽ tồn tại suốt vòng đời ứng dụng. Việc này chủ yếu nói đến các object được sử dụng thường xuyên với tần suất cao như đạn trong các trò chơi bắn súng hay đôi khi là các cư dân, bởi việc phải tạo mới liên tục các object này làm lãng phí bộ nhớ và thời gian xử lý. Người ta xây dựng các pool chứa các object cần tái chế với số lượng tính toán trước. Khi cần dùng thì thay vì tạo ra chúng họ chỉ cần lôi ra khỏi pool và khi không cần nữa thì thay vì xóa bỏ thì chỉ cần deactive chúng rồi trả về pool. CPU xử lý active cho object tốt hơn nhiều so với việc sinh ra và loại bỏ chúng. Điều này cũng giảm tải cho cơ chế dọn rác Garbage Collector (GC).

**b. Attack System with correct frame :**

-Khó khăn của việc xây dựng giải thuật để gây sát thương lên các đối tượng khác thông qua hành động là phải liên tục tính toán thời gian sự kiện xảy ra sao cho khớp với hoạt ảnh để cho ra trải nghiệm hoạt ảnh mượt mà nhất. Một hoạt ảnh luôn có 3 giai đoạn gồm chuẩn bị, thực hiện, kết thúc mà chỉ có giai đoạn thực hiện của nó là được phép chạy các function và mỗi hoạt ảnh đều có các giai đoạn với độ dài khác nhau nên với một object nhiều hoạt ảnh việc tính toán thời gian diễn ra function sẽ là cực kỳ phức tạp. Tuy nhiên vấn đề sẽ được giải quyết khi kết hợp với tính năng Action Event.

Action Event là một tính năng của Unity giúp các hàm của đối tượng có thể được gọi vào các thời điểm ứng với từng khung hình. Action Event cũng là công cụ mạnh để biểu diễn các hoạt động cần sự đồng bộ cao cũng như các tương tác vật lí thông qua hành động.

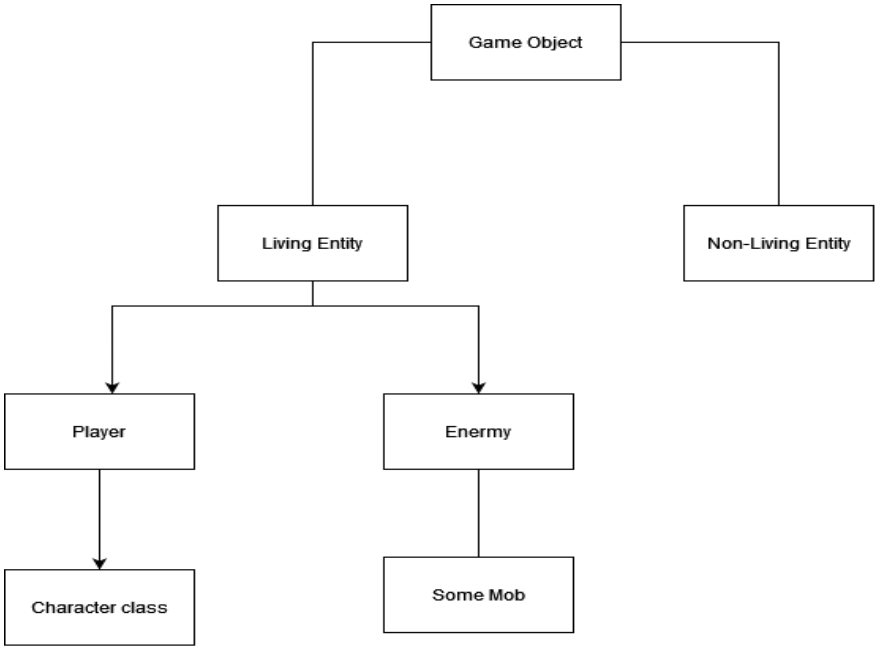
Sử dụng công cụ Action Event để gán hàm tấn công vào đúng frame của hoạt ảnh. Việc này giúp cho các hàm chỉ có thể kích hoạt vào đúng khung hình chỉ định. Đồng bộ các hành động giữa code và hoạt ảnh mà không cần phải tính toán các mốc thời gian sao cho khớp vì bản chất Action Event gán function vào chính xác mốc thời gian dựa trên hoạt ảnh chúng ta chọn. Giờ đây việc viết mã chỉ cần quyết định việc gì sẽ diễn ra còn thời điểm sẽ do Actionevent quyết định. Điều tiện ích hơn là cho dù chúng ta có dùng các thuật toán gia tốc hay giảm tốc độ hoạt ảnh thì frame thực hiện function vẫn không hề thay đổi.

Xây dựng các đòn đánh liên tục(combo) nhờ action event chỉ cần chỉ định khoảng thời gian có thể chèn thêm một input trên timeline của animation ta có thể quyết định nối tiếp chuỗi đòn đánh hoặc làm mới chuỗi đòn đánh, sao cho đòn đánh đủ thời gian để thể hiện frame đặc trưng của nó mà vẫn có thể bỏ qua hoạt ảnh kết thúc để tiếp tục hiển thị các đòn đánh sau.

Hình 17. Chỉ có 1 frame trong animation kích hoạt được sát thương đòn đánh

**c. OOP system:**

Tạo ra các class theo hướng đối tượng.




Hình 18. Mô hình OOP trong game.

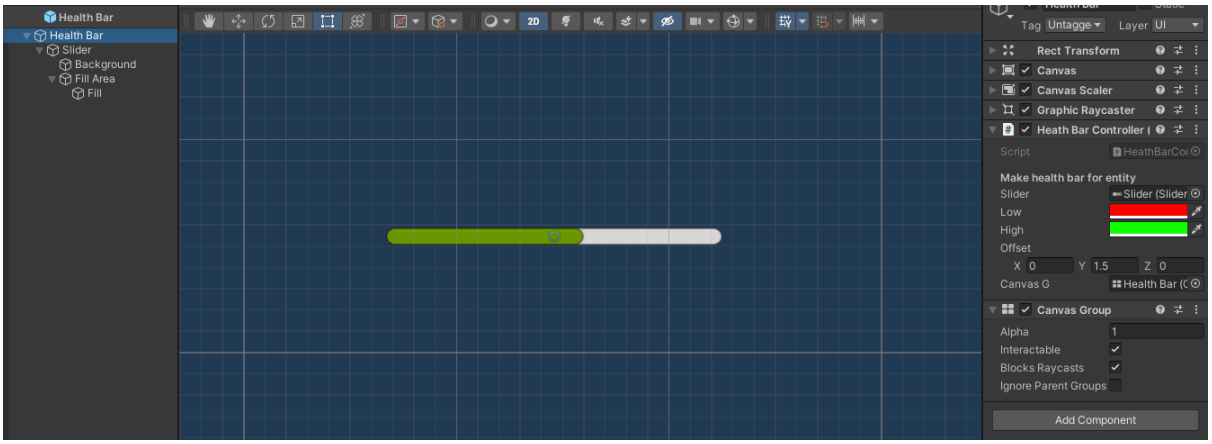
- Lớp cha hay còn gọi là lớp Abstract (Living Entity) chứa đựng tất cả thông tin chung, thống nhất cho 1 loại đối tượng như về vật thể sống bao gồm: tên, máu và các phương thức chung: nhận damage.
- Lớp con của nó như Player và Enemy sẽ khác nhau về mặt thanh máu nên mỗi lớp sẽ có một hàm riêng để xử lý. Ngoài ra lớp Player kế thừa từ trên thì có thêm một kho lưu trữ đồ (inventory).
- Các lớp con của Enemy sẽ có thêm những phương thức khác của nó như thêm khả năng bay, nhận biết xung quanh và các hiệu ứng đặt biệt.

**d. Health bar:**

Health bar dùng chung cho các living entity,ở player health bar này thể hiện dưới dạng GUI nhưng bản chất vẫn là healthbar

Healthbar gồm các thành phần:

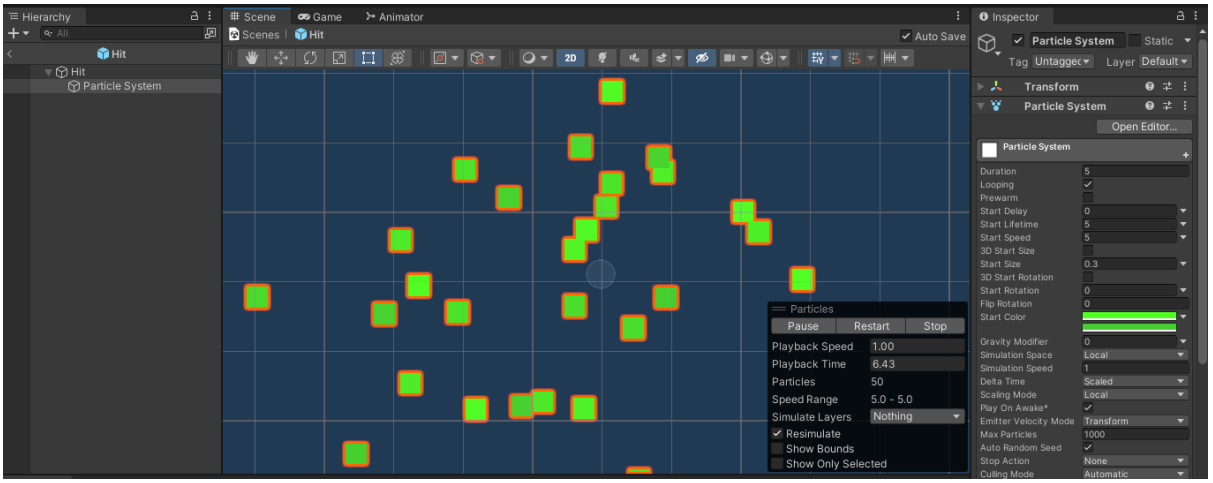
- Canvas: phần bao chứa mọi object thuộc về UI
- Slider: gồm fill react(phần làm đầy) và fill icon(phần cuối fill react)




Hình 19. Cơ chế hoạt động của health bar.

**e. Death Effect:**

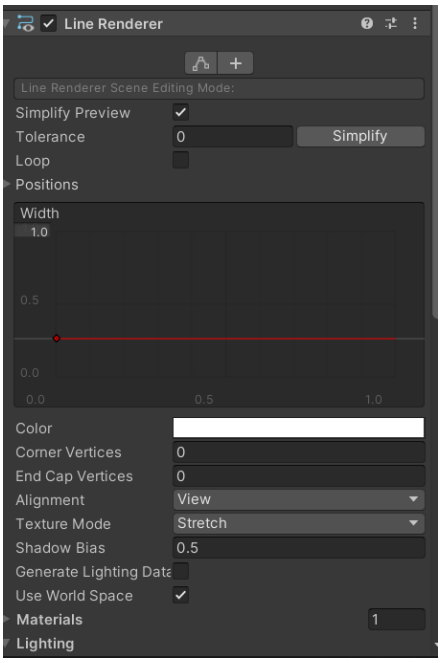
- Dead Effect sử dụng 2 phương thức gồm : Generate particle và spawn prefab.
- Generate particle: sử dụng particle system để tạo ra các hạt chuyển động trong không gian.
- Spawn prefab : tạo một object mới chứa animation khác để thể hiện dead effect.




Hình 20. Hiệu ứng particle khi một vật thể biến mất.

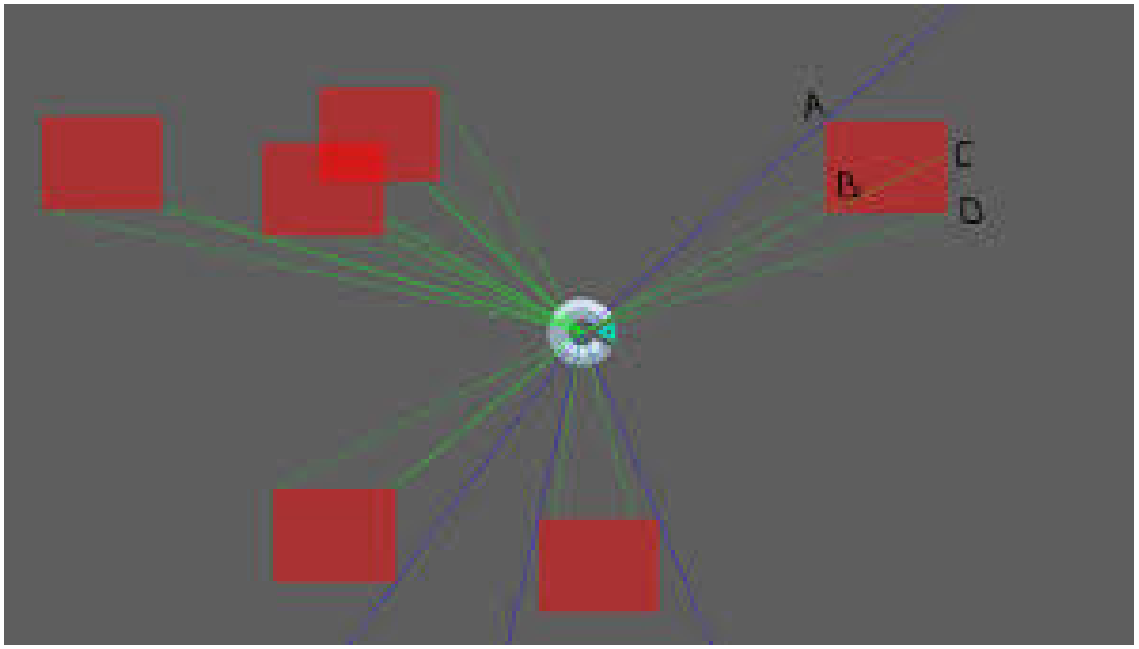
**f. Raycast:**

Sử dụng raycast để vẽ đối tượng dạng tia trên Scene. Một Raycast cơ bản gồm điểm vector bắt đầu,vector kết thúc. Raycast là vô hình và chỉ dùng để thể hiện các tương tác giữa các tia và vật thể.




**g. LineRenderer:**

Một component tương tự raycast nhưng chỉ dùng để thể hiện các tia có thể thấy và không có các thành phần xử lý va chạm.  LineRenderer gồm các position và sẽ vẽ các tia nối liên tiếp các position đó lại.



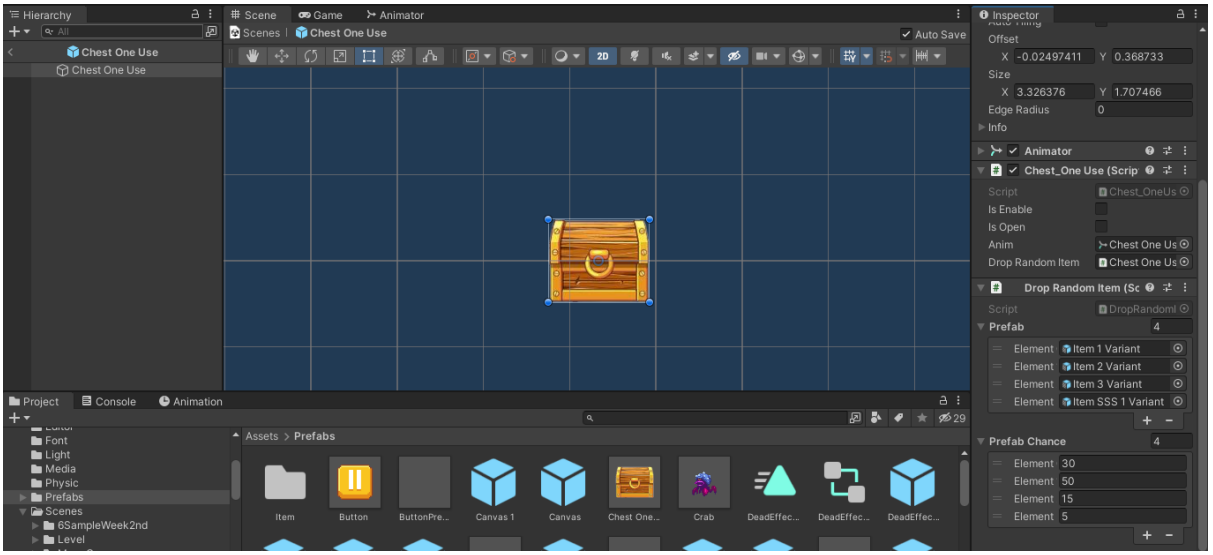

Hình 21. Raycast Unity

**h. Gacha System:**

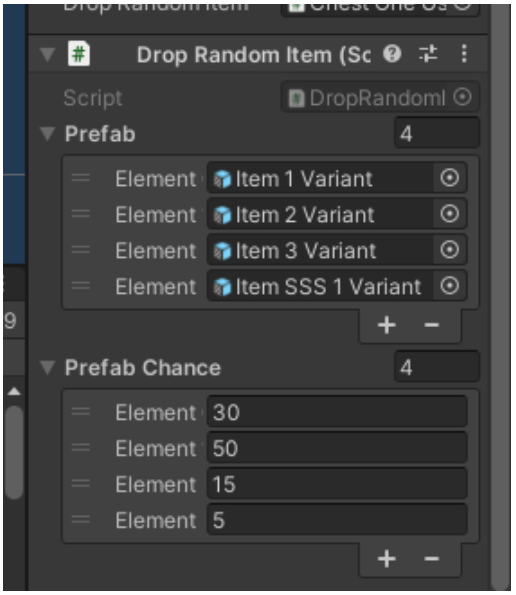
Hệ thống này gồm 3 phần:

- Random Number: Số ngẫu nhiên của mỗi lần quay
- Pool item: chứa các item chỉ định có thể rơi ra. Các item này được gán chỉ số dropchance từ trước.
- Accumulated Probability: gọi là xác suất tích lũy của lần quay

Cơ chế: Cộng tổng các tỉ lệ lại và cho Random number trong khoảng đó để chuẩn hóa lại tỉ lệ của chúng. Lưu ý là drop chance là một số tự nhiên chứ không phải tỉ lệ, các tỷ lệ hiển thị với người chơi luôn được tính toán sau. Cho một vòng lặp đến khi có vật phẩm được chọn. Nếu Random Number thấp hơn xác suất tích lũy thì vật phẩm tại vòng lặp đó được chọn để rơi ra.




Hình 22. Object rương có cơ chế gacha.




Hình 23. Các thành phần trong gacha.

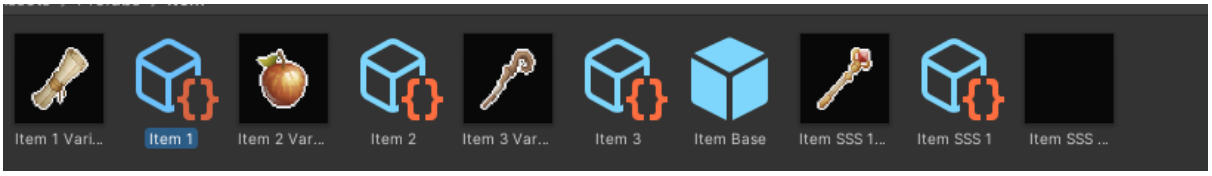
**i. ScriptObject:**

Tạo ra một class loại dữ liệu dành riêng cho các vật phẩm, ngoài ra ta có thể dùng cái này cho những thứ như: vũ khí, coin, … những thứ mà có đặt tính chung và chỉ thay đổi tên gọi hay giá trị.

```
[CreateAssetMenu(menuName = "Item", fileName = "New Item")]
4 references
public class Item : ScriptableObject//, IPointerDownHandler
{
    2 references
    public int itemID;
    2 references
    public string itemName;
    0 references
    public string itemDes;


    2 references
    public Sprite itemSprite;
    0 references
    public int itemPrice;

}
```

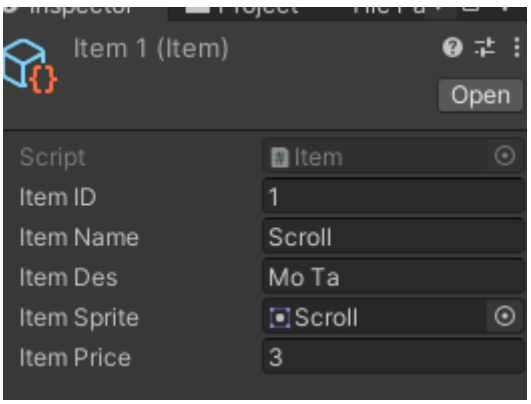
Hình 24. Tạo class Item kế thừa ScriptableObject.

Sau đó ta tạo ra mỗi sản phẩm tương ứng với mỗi đối tượng:



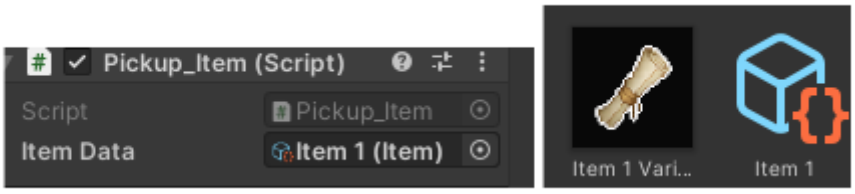

Hình 25. Các Object được tạo ra từ ScriptableObject.

Ta có thể dùng 1 file code cho nhiều sản phẩm với những mức độ khác nhau:




Hình 26. Parameter của ScriptableObject

Sau đó ta tạo 1 Object chứa liên kết đến cái trên để có thể hiển thị cho người xem, mỗi vật phẩm tương ứng với 1 Scripable Object.




Hình 27 Cách triển khai một ScriptableObject lên Object.

Và như thế ta đã tạo ra các vật phẩm một cách nhanh chóng không cần lặp lại nhiều lần.

**j. Boss**

Trong Thiết kế Game, Boss là những kẻ địch lớn do game điều khiển, hoạt động dựa vào Scripting(kịch bản có sẵn) hoặc cao cấp hơn là AI(trí tuệ nhân tạo). Cuộc chiến giữa người chơi và Boss được gọi là Boss Fight – Đấu trùm, giết trùm.

Các Boss Fight thường xuất hiện ở các giai đoạn cao trào trong mạch game. Thông thường, các giai đoạn này sẽ được chia thành các mốc cụ thể và người chơi có thể dễ dàng nhận biết được các thiết kế này. Boss sẽ mạnh hơn rất nhiều so với các đối thủ mà người chơi đã gặp trước đó.

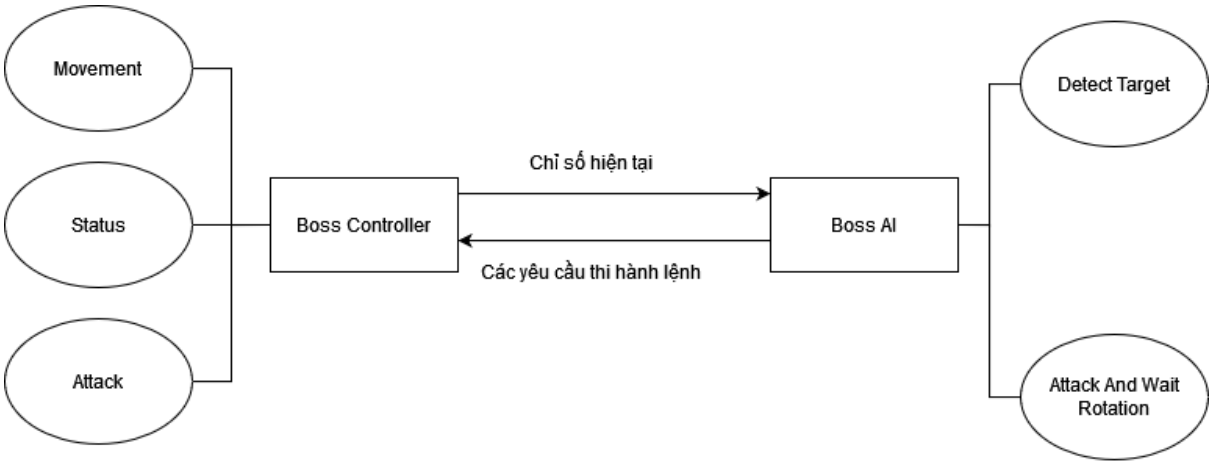
Tuyến nhân vật Boss phổ biến trong nhiều thể loại game. Và đặc biệt xuất hiện với tần suất cao ở những game có một cốt truyện rõ ràng. Ví dụ như: RPG, Adventure, Arcade,…

Khi Thiết kế Boss trong game, bạn cần đảm bảo truyền đạt được cho người chơi các mục tiêu sau

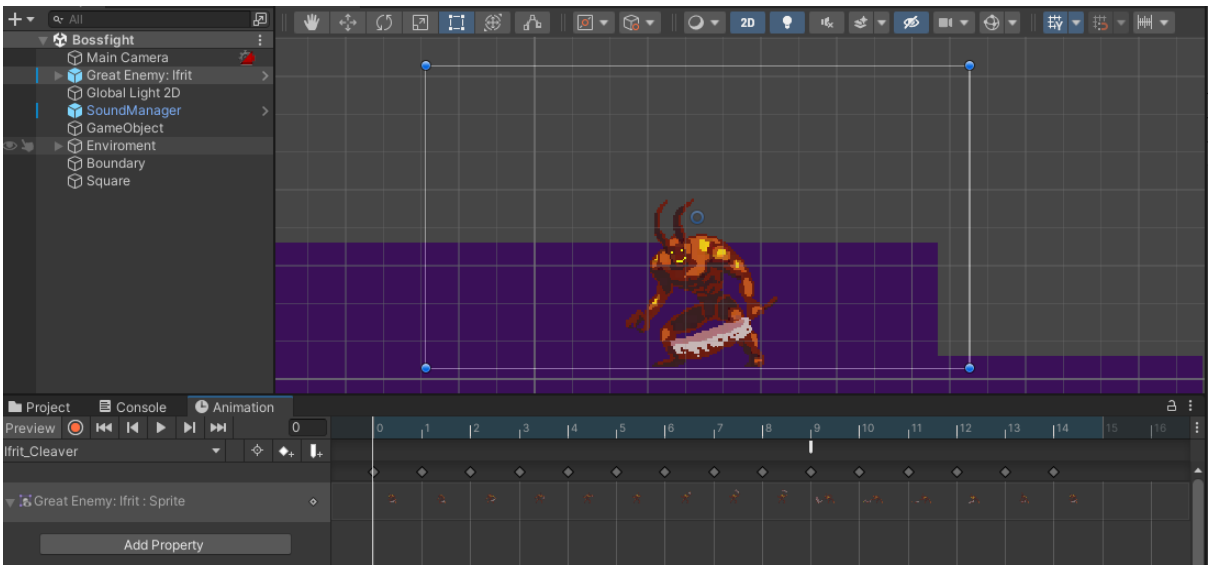
- Làm cho người chơi mong đợi các trận Boss Fight.
- Đấu Boss là nơi người chơi thể hiện các kỹ năng mình vừa học được.
- Boss Fight là một cột mốc mà người chơi cần đạt được.
- Các boss cần có những kỹ năng riêng biệt hay signature move để phân biệt với các kẻ địch khác
- Boss fight cần phù hợp với sức mạnh của người chơi thời điểm đó, tránh gây ra những màn đấu bất công

Hình ảnh 28 King Dice trong Cuphead là
một trong những thiết kế boss tốt nhất

Ở đây chúng ta sẽ khởi tạo một con Boss để chiến đấu với người chơi trong một không gian nhất định là Boss room : Demo Fight. Đầu tiên ta sẽ tạo cho nó những animation để có hoạt ảnh di chuyển qua lại và tung ra những cú đánh khác nhau.
Sơ đồ cấu tạo hành vi Boss:

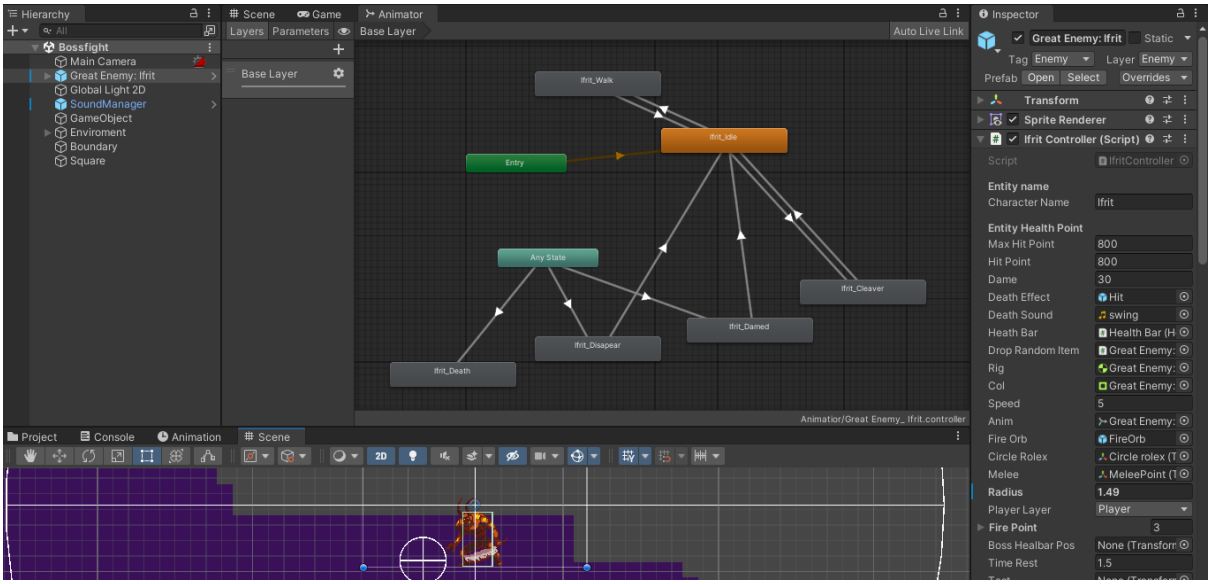



Hình 29 Sơ đồ xử lí của boss

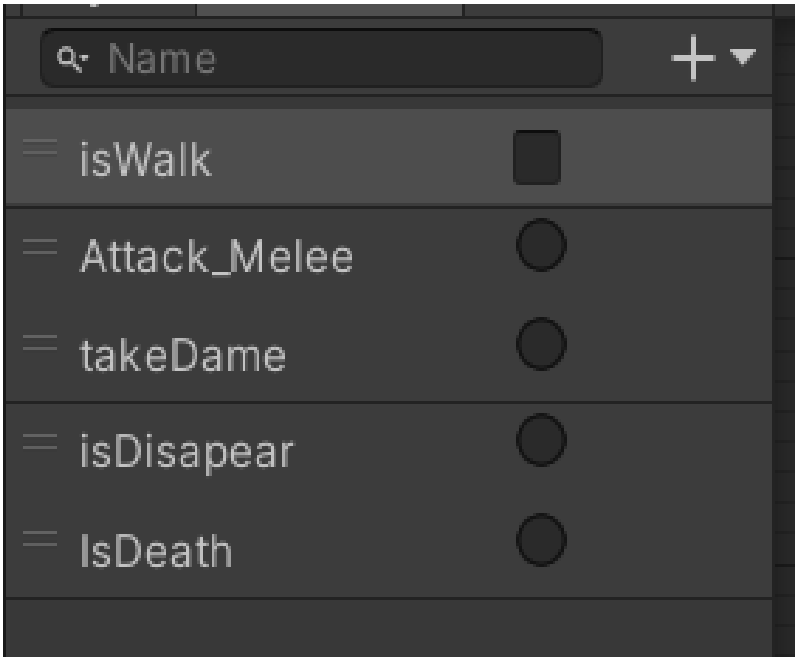



Hình 30. Tạo boss với animation.

Behavior Tree: Animator này cũng là bảng các trạng thái của Boss được kích hoạt bởi các nhóm parameter

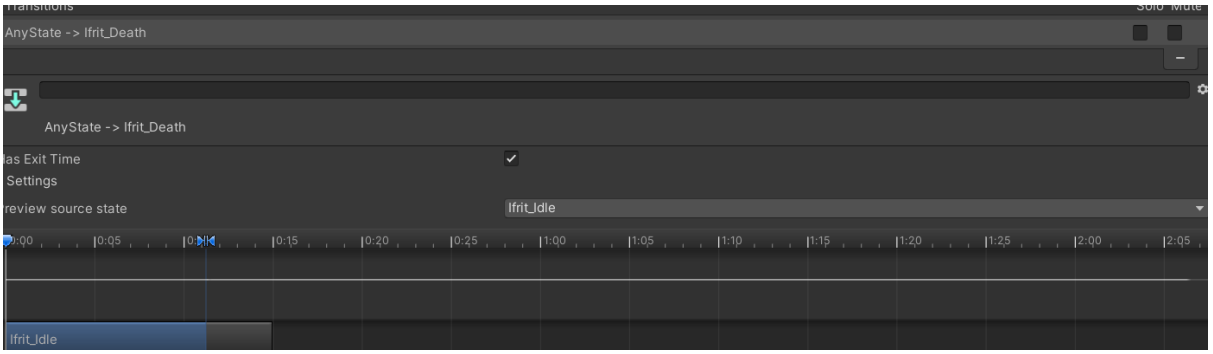



Hình 31. Các animator cho boss.




Hình 32. Các parameter của animator.

Đặt thêm các setting về HasExitTime để chỉ định các hoạt ảnh nào có thể bị ngắt và các hoạt ảnh nào phải được hoàn thành




Hình 33. Cấu hình của 1 transition giữa các state của Animator

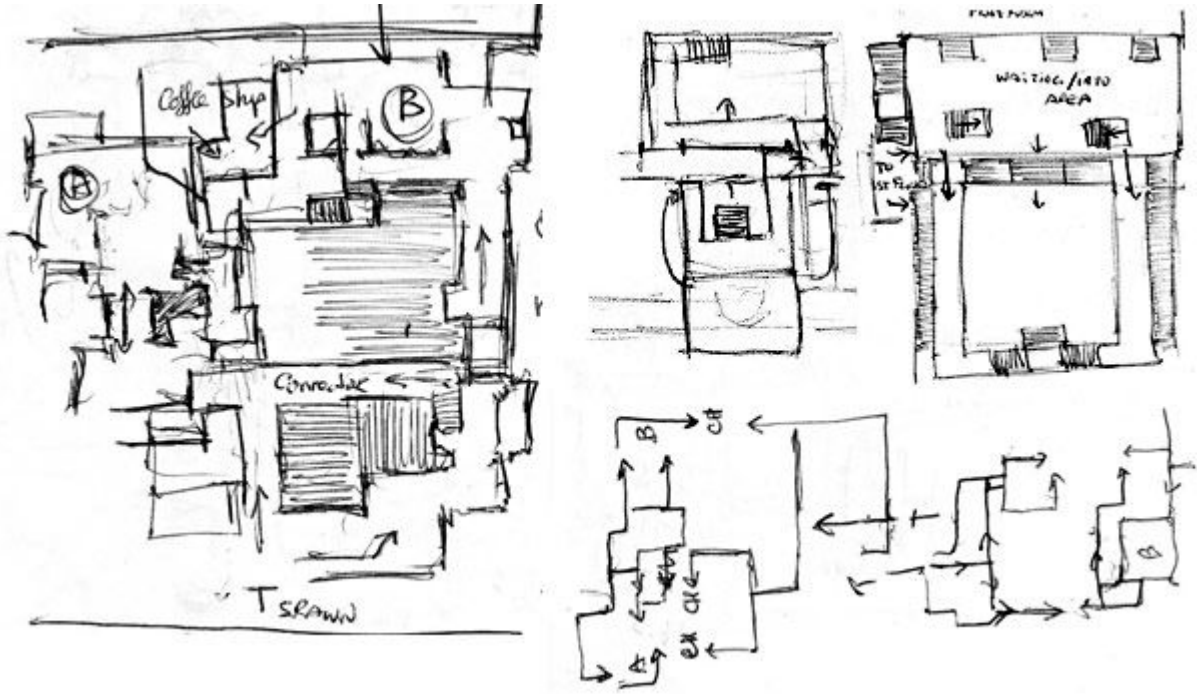
**k. Character**

Character là các nhân vật của trò chơi, có thể là người chơi và các npc. Thiết kế nhân vật hầu như không có một khuôn mẫu nào. Tuy nhiên vẫn phải đảm bảo một số tiêu chí:

- Nhân vật chính cần có sự tiến bộ sau mỗi giai đoạn, tiến bộ này thể hiện qua việc học thêm các khả năng mới, có thêm các công cụ mới
- Các kỹ năng đóng vai trò chìa khóa để đi tới giai đoạn tiếp theo hoặc chí ít là có nhiều tác dụng về sau nếu không thiết kế kĩ năng đó sẽ không còn ý nghĩa

- Các nhân vật phụ thường phải đóng một vai trò cụ thể nếu không phải là gameplay thì cũng là cốt truyện, một nhân vật phụ tròn vai là khi dù trong vai trò nào họ cũng thể hiện được tính cách cá nhân

**l. Level Design**




Hình 34. Sơ đồ thiết kế map CSGO bản phác thảo.

Thiết kế màn chơi là một trong những công việc khó nhất của phát triển game. Không hề có quy chuẩn nào và đòi hỏi sự hiểu biết của game designer về dự án. Những thiết kế thường xuyên phải được điều chỉnh thậm chí là làm lại từ đầu sau nhiều patch.
Một thiết kế đạt yêu cầu tối thiểu đáp ứng các yêu cầu sau:

- Cung cấp đủ các công cụ để người chơi có thể vượt qua
- Đa dạng trong cách giải quyết
- Phần thưởng xứng đáng với độ khó
- Thiết kế các yếu tố như quái và bẫy cần tính toán sao cho không thể bị bỏ qua
- Thể hiện đúng tính chất của bối cảnh
- Người chơi buộc phải làm quen và tìm cách vượt qua bằng những công cụ đã có

# Tài liệu tham khảo


1. (n.d.). Wikipedia, the free encyclopedia. Retrieved May 23, 2022, from https://www.youtube.com/c/Freecodecamp
2. (n.d.). Unity Forum. Retrieved May 26, 2022, from https://forum.unity.com/
3. (n.d.). raywenderlich.com | High quality programming tutorials: iOS, Android, Swift, Kotlin, Flutter, Server Side Swift, Unity, and more! Retrieved May 26, 2022, from https://www.raywenderlich.com/
4. //. (n.d.). // - Wikipedia. Retrieved May 26, 2022, from https://vn1lib.org/s/unity
5. Afzal Hussain, Haad Shakeel, Nasir Uddin, & Faizan Hussain. (n.d.). *Unity Game Development Engine: A Technical Survey*. Unity Game Development Engine: A Technical Survey. https://www.researchgate.net/publication/348917348_Unity_Game_Development_Engine_A_Technical_Survey
6. *Brackeys*. (n.d.). YouTube. Retrieved May 23, 2022, from https://www.youtube.com/c/Brackeys
7. *Cẩm nang vài điều về Thiết kế Boss trong Game*. (n.d.). Thiết kế Game. Retrieved May 25, 2022, from https://thietkegame.com/huong-dan-chuyen-sau/cam-nang-thiet-ke-boss-trong-game/
8. *Cẩm nang vài điều về Thiết kế Boss trong Game*. (n.d.). Thiết kế Game. Retrieved May 26, 2022, from

https://thietkegame.com/huong-dan-chuyen-sau/cam-nang-thiet-ke-boss-trong-game/

9. *Code Monkey*. (n.d.). YouTube. Retrieved May 23, 2022, from https://www.youtube.com/c/CodeMonkeyUnity
10. *C# Unity Developer 2D Coding: Learn to Code Video Games*. (n.d.). Udemy. Retrieved May 26, 2022, from https://www.udemy.com/course/unitycourse/
11. *Danh sách khóa học*. (n.d.). Danh sách khóa học | How Kteam. Retrieved May 26, 2022, from https://www.howkteam.vn/learn/lap-trinh/unity3d-7-20
12. *Download C# Game Programming Cookbook for Unity 3D Free PDF*. (n.d.). OiiPDF.COM. Retrieved May 26, 2022, from https://oiipdf.com/c-game-programming-cookbook-for-unity-3d
13. *Jason Weimann*. (n.d.). YouTube. Retrieved May 23, 2022, from https://www.youtube.com/c/Unity3dCollege
14. Kelly, C. (n.d.). *Programming 2D Games by Charles Kelly.pdf*. Index of /. Retrieved May 26, 2022, from https://canvas.projekti.info/ebooks/Programming%202D%20Games%20by%20Charles%20Kelly.pdf
15. *Learn Game Development Without Coding Experience*. (n.d.). Unity. Retrieved May 26, 2022, from https://unity.com/learn
16. Marzan, J. (n.d.). *head-first-csharp/fourth-edition: Code and graphics for the projects in the 4th edition of Head First C#.* GitHub. Retrieved May 26, 2022, from https://github.com/head-first-csharp/fourth-edition

17. Pereira, V. (n.d.). *Download Learning Unity 2D Game Development by Example free PDF by Venita Pereira*. OiiPDF.COM. Retrieved May 26, 2022, from https://oiipdf.com/learning-unity-2d-game-development-by-example
18. *Sai Game*. (n.d.). YouTube. Retrieved May 23, 2022, from https://www.youtube.com/c/SaiGame
19. *Sebastian Lague*. (n.d.). YouTube. Retrieved May 23, 2022, from https://www.youtube.com/c/SebastianLague
20. Trujillo, L. (n.d.). *(PDF) Pro Unity Game Development with C# | Lux Trujillo*. Academia.edu. Retrieved May 26, 2022, from https://www.academia.edu/11201938/Pro_Unity_Game_Development_with_C_
21. *Unity*. (n.d.). Papers With Code. Retrieved May 26, 2022, from https://paperswithcode.com/task/unity
22. *Unity3d Research Papers*. (n.d.). Academia.edu. Retrieved May 26, 2022, from https://www.academia.edu/Documents/in/Unity3d
23. *Unity for Absolute Beginners*. (n.d.). PDF Drive. Retrieved May 26, 2022, from https://www.pdfdrive.com/unity-for-absolute-beginners-e32683741.html
24. *Unity Publications*. (n.d.). Unity. Retrieved May 26, 2022, from https://unity.com/publications