# CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

## 1.1. . Giới thiệu

Trong thời đại CNTT như hiện nay, sản phẩm công nghệ ngày càng chịu sự đánh giá khắt khe hơn từ phía những người dùng, đặc biệt là về sản phẩm Game được nhận rất nhiều sự đánh giá từ phía các Game thủ, hay chỉ là những người chơi bình thường. Ngành công nghiệp Game hiện nay có thể nói là bùng nổ, với tốc độ phát triển đến chóng mặt, rất nhiều những Game hay và hấp dẫn đã được ra đời trong thời gian qua. Phía sau những Game phát triển và nổi tiếng như vậy đều có một GE. GE là một công cụ hỗ trợ, một Middleware giúp người phát triển viết Game một cách nhanh chóng và đơn giản, đồng thời cung cấp khả năng tái sử dụng các tài nguyên và mã nguồn cao do có thể phát triển nhiều Game từ một Game Engine (GE). Từ xu hướng phát triển và những bất cập trên, đồ án này sẽ khảo sát và nghiên cứu về Engine Unity – một GE rất phổ biến và không kém mạnh mẽ hiện nay nhằm thực nghiệm việc phát triển một trò chơi (Demo) Flappy Bird.

## 1.2. Khảo sát hiện trạng

Mặc dù "cha đẻ" Nguyễn Hà Đông đã quyết định khai từ trò chơi Flappy Bird do chính mình tạo ra tuy nhiên tên tuổi của Flappy Bird vẫn đang tạo nên những "cơn sốt" trong giới công nghệ và truyền thông, khi hàng loạt các trang web và tờ báo lớn vẫn liên tục đưa tin về trò chơi này. Với sự vươn lên mạnh mẽ và bất ngờ nổi tiếng của Flappy Bird, hàng loạt các trang báo lớn đã đưa ra nghi vấn về việc Nguyễn Hà Đông sử dụng "mánh khóe" để tăng lượt download cũng như lượt đánh giá về trò chơi.

## 1.3. Đối tượng và mô tả đối tượng nghiên cứu

Nghiên cứu phát triển triển game Flappy Bird trên 2 nền tảng phổ biến nhất hiện nay là Android và IOS. Các đối tượng nghiên cứu gồm:

- Nhân vật trong game
- Phân cảnh trong game
- Màu sắc của phân cảnh
- Vật cản trong game

## 1.4. **Mục tiêu đề tài**

Mục tiêu của đề tài là thiết kế và phát triển game Flappy Bird 2D bằng công nghệ Unity phục vụ cho người dùng.

## 1.5. **Phạm vi đề tài**

Game Flappy Bird đem lại cho mọi người cảm giác chơi game giải trí sau những hàng giờ áp lực cho công việc. Giúp cho mọi người giải tỏa stress để đem lại hiệu suất cao trong công việc. Game này sẽ có một số tính năng như sau:

- Di chuyển
- Tích điểm
- Tạo game mới
- Phân cảnh
- Tạo vật cản
- Animation cho nhân vật trong game

## 1.6. **Cấu trúc của đề tài**

**Chương 1:** Tổng quan về đề tài

**Chương 2:** Cơ sở lý thuyết

**Chương 3:** Phân tích và thiết kế hệ thống game

**Chương 4:** Thực nghiệm và triển khai

**Kết luận**

Kết quả đạt được

Hướng phát triển

**Tài liệu tham khảo**

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1. **Giới thiệu về Unity**

- Lập trình Unity 2D và 3D được sử dụng dựa trên 3 ngôn ngữ lập trình đó là C#, UnityScript, Boo. Thông thường, ngôn ngữ chính mà mỗi lập trình viên Unity sẽ sử dụng phổ biến nhất là C#.
- Đến đây, chắc hẳn nhiều bạn độc đang phân vân và không biết tại sao Unity lại được sử dụng để thiết kế game rồi mà vẫn còn phải biết rõ những lập trình phía trên.
- Unity3D là một môi trường phát triển tích hợp, mạnh mẽ, hỗ trợ thao tác kéo thả, tuỳ biến giao diện nhanh chóng, trực quan.
- Cung cấp các công cụ xử lý đồ hoạ, tích hợp sẵn thư viện vật lý, tính toán va chạm...
- Hỗ trợ phát triển cả game 2D và 3D.
- Hỗ trợ nhiều nền tảng thông dụng như OSX, Linux, Window, Web, iOS, Window Phone 8, Android, PS3, BB…
- Cộng đồng người dùng, hỗ trợ rộng lớn.
- Có phiên bản miễn phí có thể chạy được trên Window và Mac OSX


Hình 2.1: Hệ điều hành

Ngày nay rất nhiều nhà phát triển game lựa chọn Unity3D để phát triển bởi khả năng hỗ trợ đa nền tảng sự mạnh mẽ tiện dụng của Unity 3D.

Đến với Unity, các bạn sẽ không cần phải băng khoăn về các vấn đề xử lý, các khái niệm đồ hoạ phức tạp... tất cả đều trở nên dễ dàng và nhanh chóng với Unity. Một số game được phát triển trên Unity như AngryBird, Elegy of War,..

## 2.2. **Phân tích công nghệ**

Unity hỗ trợ mạnh các tính năng sau:

- Tạo giao diện UI của Game như tạo drop bar, textbox,…
- Hiển thị mô hình 3D, 2D – hệ thống vật lý 2D, 3D
- Networking: Hỗ trợ tạo game chơi nhiều người cùng lúc
- Hỗ trợ nền tảng đặc biệt mới: AR – Augmented reality (Thực tế tăng cường), VR – Virtual reality (Thực tế ảo)
- AI – Hỗ trợ con bot trong màn hình game, hỗ trợ package tạo nên bot trong game
- Hỗ trợ hiển thị chữ bằng font đặc biệt

Trong tất cả các tính năng của Unity thì đề cao nhất giao diện ứng dụng Editor. Unity Editor, với nhiều công cụ (tool), hỗ trợ các đầu việc phát triển game như tạo mô hình 3D (vật lý, ánh sáng) và viết Script,… chỉ bằng thao tác kéo thả hình ảnh.

Nhờ vào tính năng này, Unity hỗ trợ tốt cho cả những vị trí khác trong quá trình phát triển Game như Game Designer.

## 2.3. **Ưu điểm và khuyết điểm của công nghệ Unity**

**Ưu điểm:**

Trên thị trường game quốc tế và Việt Nam vẫn có những GE được nhiều developer ưa chuộng như Unreal, CryEngine,…

Mỗi engine sẽ có mỗi ưu nhược điểm khác nhau và vẫn được sử dụng ở nhiều studio game. Xét riêng về sức mạnh tổng quan khi làm game, Huy chia sẻ về ba ưu điểm lớn mà Unity sở hữu:

- Editor: Với Editor, nhà phát triển không cần thiết phải viết Code để sắp đặt các đối tượng trong Game như những Engine khác mà Developer có thể kéo thả, thay đổi vị trí của từng đối tượng trong Game trực tiếp trên Editor.
- Đa nền tảng là lợi ích thứ 2 rất quan trọng với nhiều công ty cũng như developer. Vì với việc tạo ra Game mà Game đó có thể chạy được trên hầu hết những hệ điều hành quan trọng như Desktop (Mac, Window và Linux) hay Mobile (iOS, Android) hoặc Web (WebGL) thì cũng đã tiết kiệm công sức cũng như chi phí rất nhiều cho doanh nghiệp đó.

- Miễn phí: Và yếu tố cuối cùng chính là chi phí. Với Unity, miễn phí là một điểm thu hút rất nhiều Developer chọn làm việc với GE này. Tuy nhiên, với các game được tạo ra miễn phí thì bắt buộc phải có Logo Unity trong Game.

**Nhược điểm:**

- Dung lượng Unity game bundle khá lớn:
- So với những GE  khác, Unity sản xuất game có dung lượng nặng nên đây là một điểm trừ lớn. Một game tốt nhất chỉ nặng dưới 100MB. Thậm chí, game web do Unity sản xuất có thể có dung lượng lên đến cả trăm MB nên web chạy không nổi. Chính vì thế, cũng cùng một game đó thì game mobile lại chạy tốt trong khi game web lại giật, lag.
- Các phiên bản cập nhật.

## 2.4. **Lập luận về công nghệ**

Qua việc phân tích Công nghệ Unity ở trên, đề tài này lựa chọn Công nghệ Unity để phát triển game theo mục tiêu của đề tài.

# CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG GAMES

## 3.1. Các khái niệm cơ bản

### 3.1.1. Game Object

Là các đối tượng cơ bản trong lập trình Unity, đại diện cho các nhân vật, phong cảnh, đạo cụ,… Mọi đối tượng của trò chơi là Game Object.

Game Objects đại diện cho các mục của trò chơi, không gian mà để xây dựng cấp độ được gọi là bối cảnh. Khi lập trình game Unity 2D, có thể bỏ qua trục thứ 3 (hay còn gọi là trục Z)

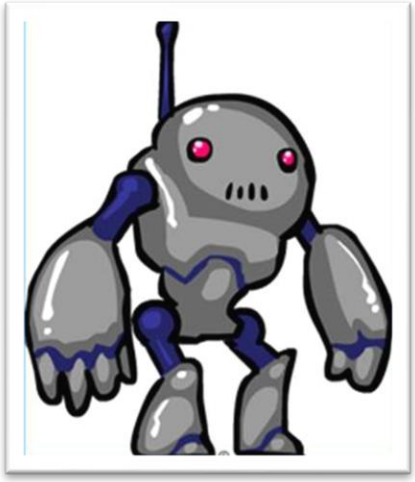
Hành vi của Game Objects được xác định bởi các khối chức năng được gọi là các thành phần. Các thành phần sau đây là cơ bản của trò chơi 2D.

### 3.1.2. Component

Một GameObject sẽ có nhiều thành phần cấu tạo nên nó như là hình ảnh (sprite render), tập hợp các hành động (animator), thành phần xử lý va chạm (collision), tính toán vật lý (physical), mã điều khiển (script), các thành phần khác... mỗi thứ như vậy gọi là một component của GameObject.

### 3.1.3. Sprite

Là một hình ảnh 2D của một game object có thể là hình ảnh đầy đủ, hoặc có thể là một bộ phận nào đó.



Hình 3.1: Sprit

### 3.1.4. Animation

Là tập một hình ảnh động dựa trên sự thay đổi liên tục của nhiều sprite khác nhau.

### 3.1.5. Key Frame

Key Frame hay Frame là một trạng thái của một animation. Có thể được tạo nên từ 1 sprite hay nhiều sprite khác nhau.

### 3.1.6. Prefabs

Là một khái niệm trong Unity, dùng để sử dụng lại các đối tượng giống nhau có trong game mà chỉ cần khởi tạo lại các giá trị vị trí, tỉ lệ biến dạng và góc quay từ môt đối tượng ban đầu.

### 3.1.7. Sounds

Âm thanh trong game.

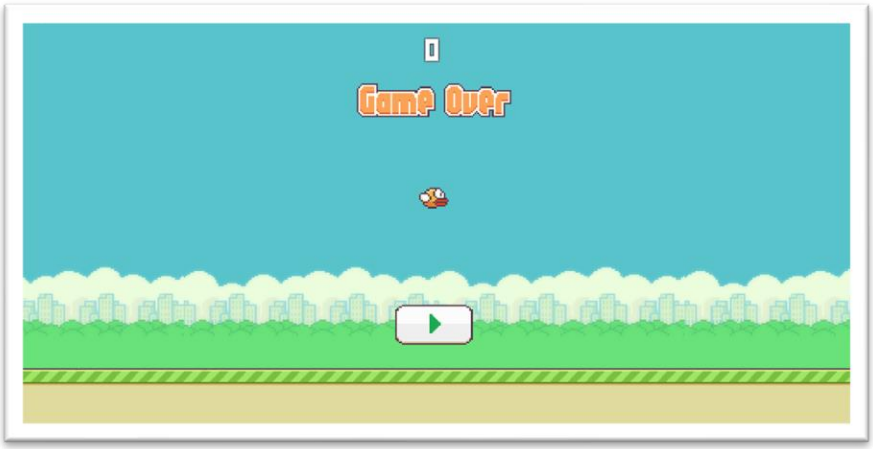
### 3.1.8. Script

Script là tập tin chứa các đoạn mã nguồn, dùng để khởi tạo và xử lý các đối tượng trong game.

Trong Unity có thể dùng C#, Java Script, BOO để lập trình Script.

### 3.1.9. Scenes

Quản lý tất cả các đối tượng trong một màn chơi của game.




Hình 3.2: Một scene game

### 3.1.10. Assets

Bao gồm tất cả những gì phục vụ cho dự án game như sprite, animation, sound, script, scenes…

### 3.1.11. Camera

Là một game object đặc biệt trong scene, dùng để xác định tầm nhìn, quansát các đối tượng khác trong game.

### 3.1.12. Transform

Thành phần sẽ xác định vị trí xoay và tỷ lệ của từng gameobject trong các bối cảnh. Mỗi một gameobject đều sẽ có 1 thành phần Transform

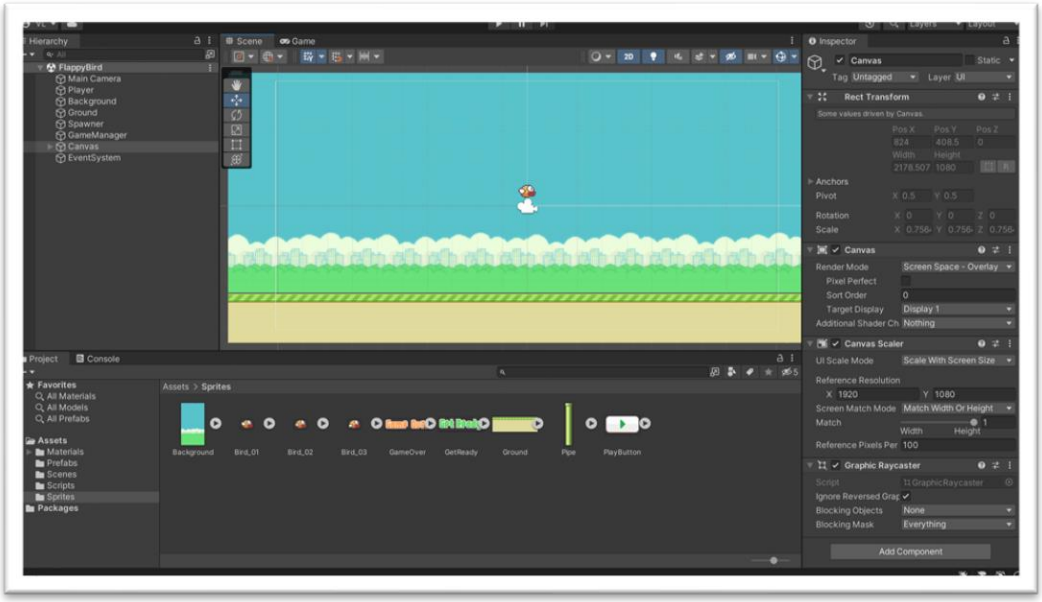
Sprite Renderer: Thành phần này kết xuất sprite và điều khiển nó trông như thế nào trong mỗi bối cảnh

Là 3 phép biến đổi tịnh tiến, quay theo các trục, và phóng to thu nhỏ một đối tượng

## 3.2. Làm quen với môi trường Unity

### Các thành phần và bố trí

- Bố trí mặc định của Unity.




Hình 3.3: Bố trí mặc định

### 3.2.1. Scenes

- Phần màu đỏ số 1.
- Phần này phần hiển thị các đối tượng trong scenes một cách trực quan, có thể lựa chọn các đối tượng, kéo thả, phóng to, thu nhỏ, xoay các đối tượng ...
- Phần này có để thiết lập một số thông số như hiển thị ánh sáng, âm anh, cách nhìn 2D hay 3D .

### 3.2.2. Game

- Phần màu đỏ số 1
- Phần này hiển thị game khi thực thi, một tab bên cạnh của tab Scenes.

### 3.2.3. Inspector

- Phần số 2 màu vàng
- Phần này hiển thị các component của một Game Object và các thông số của các component.

### 3.2.4. Project

- Phần số 3 màu xanh lá cây
- Phần này hiển thị thư mục Assets, chứa tất cả các tài nguyên của dự án game.
- Ở phần này, bên cạnh tab khác, có phần Console để hiển thị các log trong quá trình debug.

### 3.2.5. Hierarchy

- Phần đóng khung màu tím
- Phần này quản lý tất cả các đối tượng trong scenes, có thể chọn lựa, đổi tên, xoá các đối tượng ra khỏi game.

### 3.2.6. Top bar

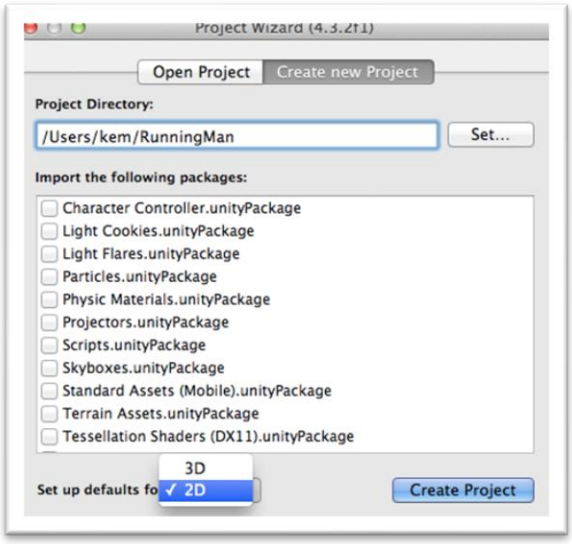
- Phần đóng khung màu đà
- Phần này chứa các nút chuyên dụng:
    - Bốn nút bên trái: (1) tuỳ chọn cho phép dùng chuột kéo toàn bộ scenes, phóng to, thu nhỏ, lựa chọn các đối tượng trong scenes... (2) cho phép dùng chuột di chuyển các đối tượng trong scenes, (3) cho phép dùng chuột quay các đối tượng (4) cho phép dùng chuột phóng to, thu nhỏ các đối tượng

- Ba nút ở giữa: (1) cho phép chạy demo game, (2) cho phép dừng game tại một frame nào đó, (3) cho phép chạy từng frame.
- Hai drop list bên phải: (1) cho phép tuỳ chọn hiển thị các layer, (2) cho phép chọn và lưu các bố trí do người dùng thiết lập.

## 3.3. **Tạo và cấu hình với dự án 2D**

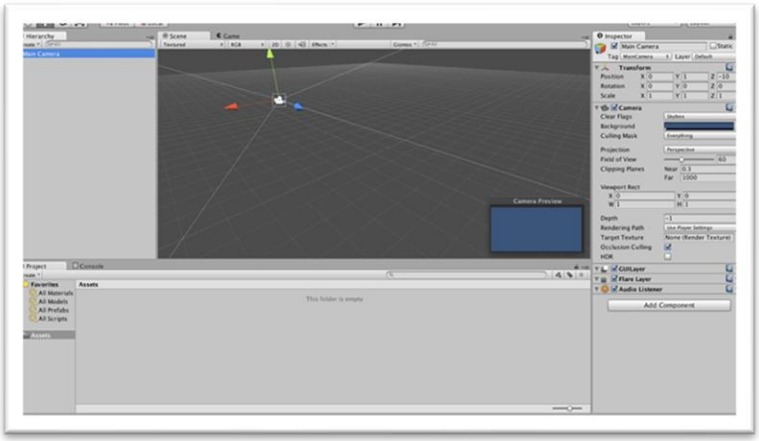
### **3.3.1. Tạo dự án**

Bật Unity lên, vào menu File/New Project




Hình 3.4: New Project

Chọn loại project là 2D sau đó gõ tên project xong nhấn Create Project. Kết quả như sau:



Hình 3.5: First look

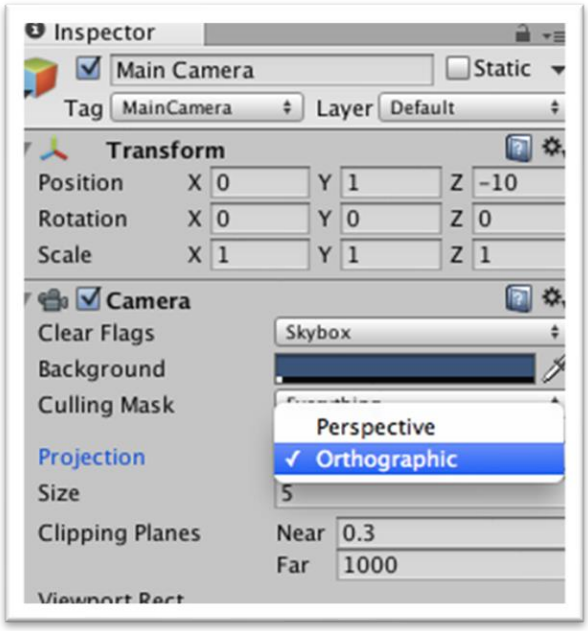
### 3.3.2. Cấu hình dự án 2D

Sau khi tạo dự án xong, ở Hierarchy sẽ có một đối tượng là Main Camera.

Điều chỉnh kiểu hiển thị Scene là 2D


Hình 3.6: Scene

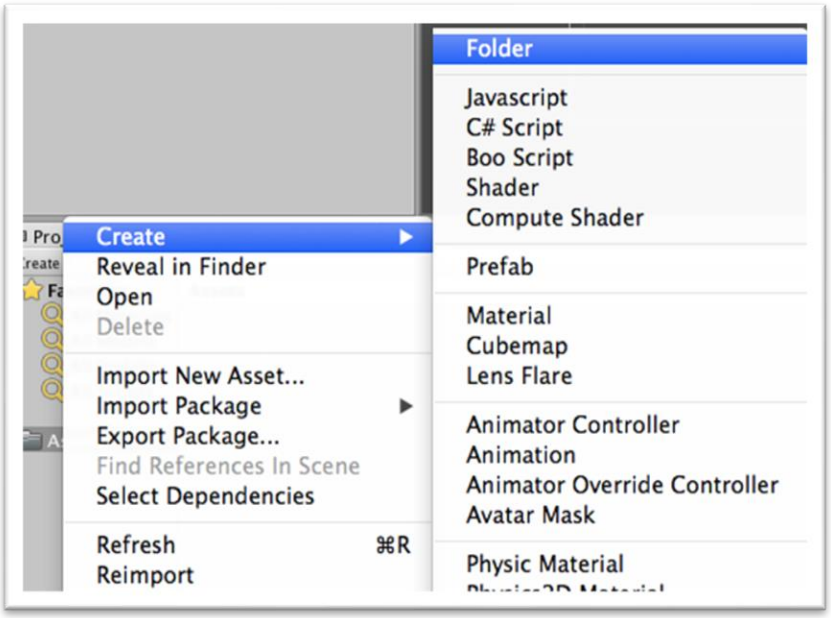
Ở cửa sổ Hierarchy chọn Main Camera. Ở cửa sổ Inspector sẽ hiển thị các thông số của camera, chọn lại giá trị Projection hay phép chiếu là phép chiếu vuông góc thay cho phép chiếu phối cảnh.




Hình 3.7: Inspector

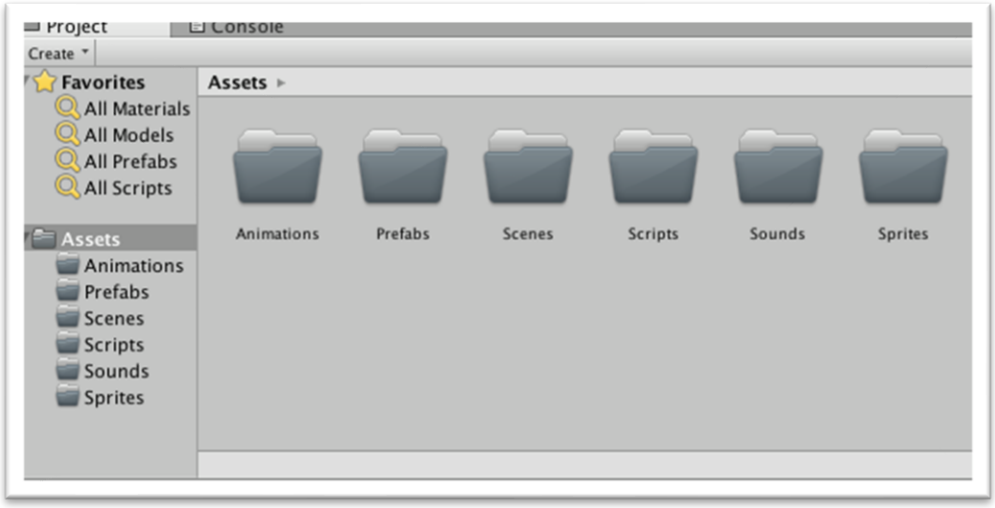
Tiếp theo, ở cửa sổ Project, sẽ tạo sẵn các thư mục để chứa tất cả các tài nguyên có sử dụng trong game sau này lần lượt là: Animations, Prefabs, Scripts, Sprites, Sounds, Scenes.

R-Click vào thư mục Assets ở cửa sổ Project, chọn Create, chọn Folder.




Hình 3.8: Tạo folder

Kết quả như hình sau:



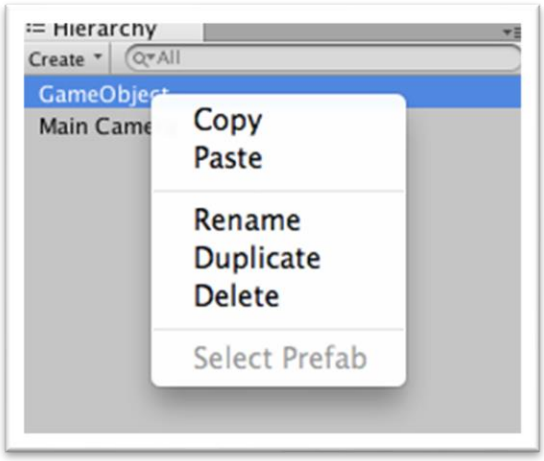

Hình 3.9: Các folder trong assets

## 3.4. **Tạo các đối tượng cơ bản**

### 3.4.1. Game Object

#### *3.4.1.1. Empty Object*

Ở menu chọn GameObject, chọn Create Empty.




Hình 3.10: New object

Có thể đổi tên, sao chép hay xoá các đối tượng thông qua cửa sổ này.

Empty game object là một đối tượng đơn giản nhất, khi mới tạo ra, chỉ chứa các thông số biến đổi (transform) ngoài ra nó sẽ không chứa bất kỳ một component nào cả.

Có thể thêm các component hoặc để nhóm các đối tượng khác lại với nhau thành một nhóm, hoặc sử dụng các empty object cho các mục đích khác (sẽ ứng dụng sau)...

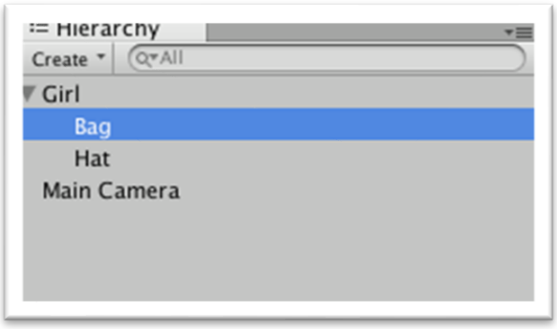
*3.4.1.2. Parent object và Child object*

Parent object: là một đối tượng chứa các đối tượng con khác, gắn liền với nhau.

Child object: là một đối tượng nằm trong một đối tượng khác, gắn liền với nhau.

Để tạo các đối tượng parent và child, chỉ việc kéo thả một đối tượng có sẵn vào trong đối tượng đã có trong cửa sổ Hierarchy. Khi thay đổi các đối tượng con thì chỉ có tác động trên đối tượng con đó.

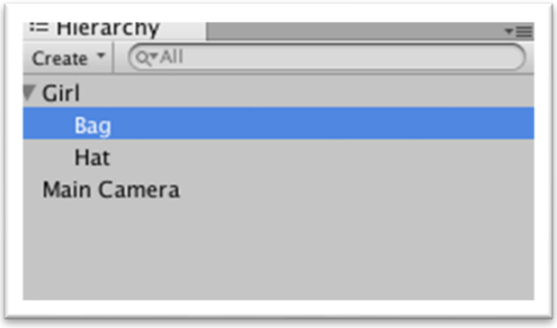
Khi thay đổi đối tượng cha thì các đối tượng con sẽ thay đổi theo.



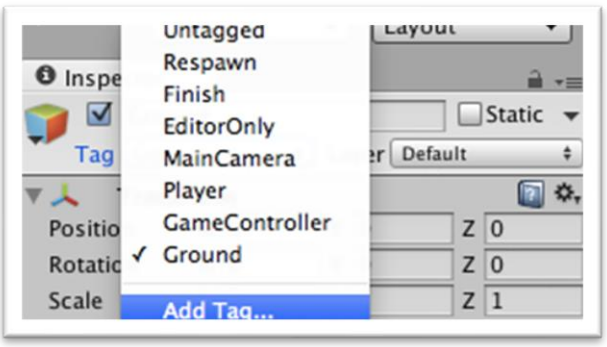

Hình 3.11: Đối tượng Girl có các đối tượng con là Bag, Hat.

*3.4.1.3. Tag*

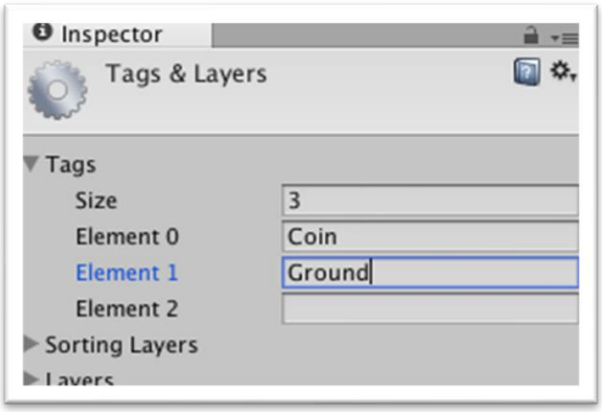
Tag là một thuộc tính của của một game object. Sử dụng thuộc tính này mục đích xác định và phân biệt các đối tượng với nhau, khi xử lý sự kiện hay bất kể vấn đề gì cần.




Hình 3.12: Tag




Hình 3.13: Thêm tag mới




Hình 3.14: Danh sách các tag của người dùng

### 3.4.2. Sprite

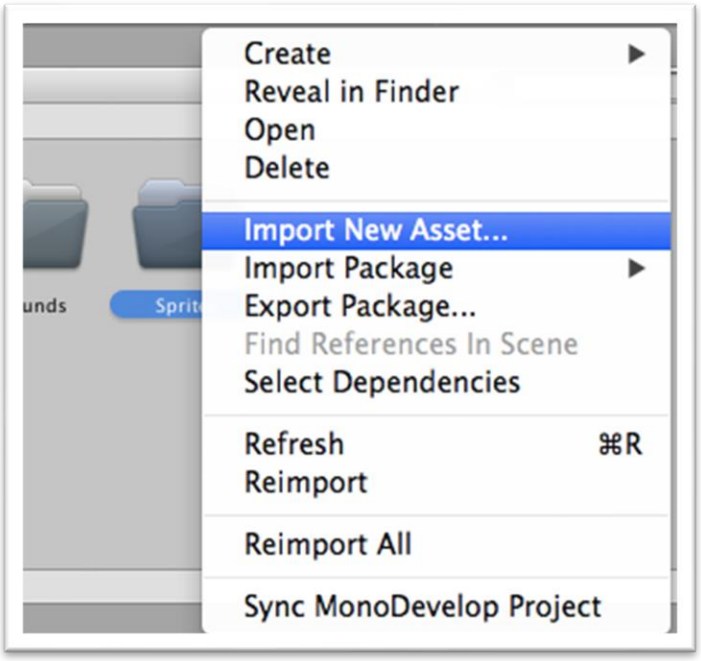
Có 2 loại sprite là Single sprite và Multiple sprite.

Hình 3.15: Single sprite

Hình 3.16: Multiple sprite

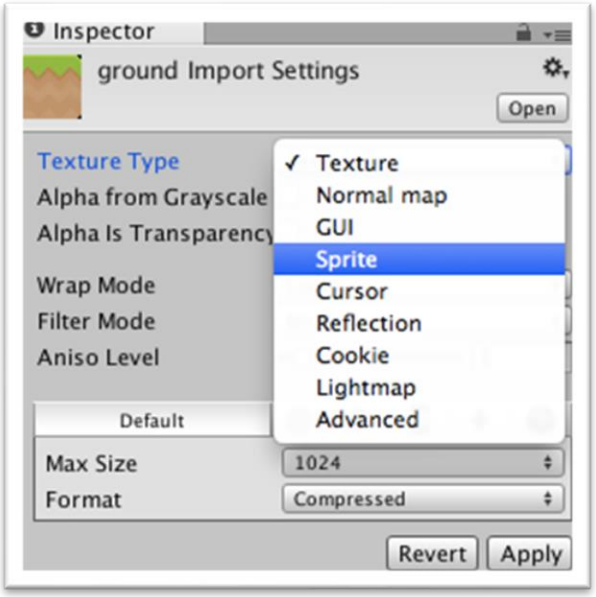
*3.4.2.1. Single sprite*

Ở cửa sổ Project, chọn thư mục Assets, R-Click vào thư mục Sprites, chọn Import New Assets, sau đó tìm đến một hình ảnh nào đó.




Hình 3.17: Import New Assets

Ở cửa sổ Inspector, chọn Texture Type là Sprite, Sprite Model là single, sau đó nhấn Apply.




Hình 3.18: Chọn texture type là Sprite

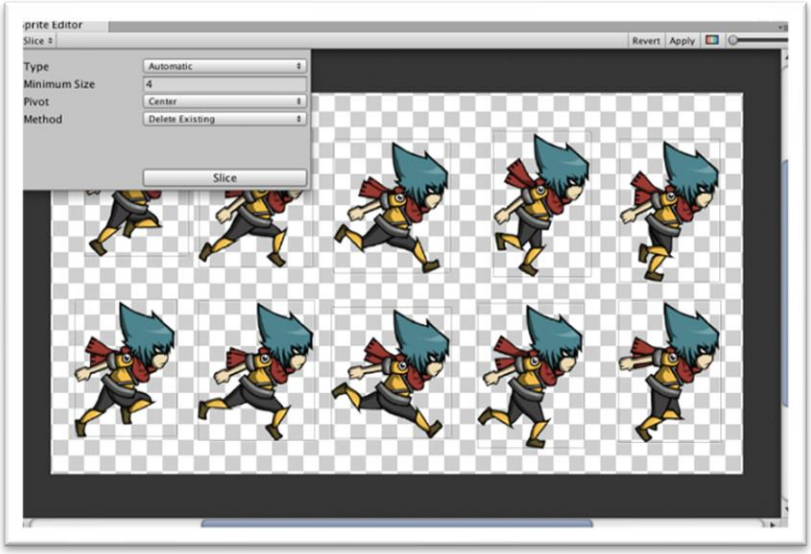
*3.4.2.2. Multiple sprite*

Tiến hành tương tự, Import New Assets, chọn đến một tập tin ảnh chứa nhiều Sprite như ở ví dụ trên, sau đó ở cửa sổ Inspector chọn Texture Type là Sprite, Sprite Model là Multiple. Xong nhấn Apply.


Hình 3.19: Multiple sprite

Tiếp theo cần xác định các vùng bao, để xác định các sprite con thuộc Multiple sprite vừa import vào đó bằng cách ở bảng Inspector chọn Sprite Editor.

Ở cửa sổ Sprite Editor, chọn Slice, sau đó để mặc định các thông số, chọn nút Slice, Unity sẽ tự động xác định vùng bao nhỏ nhất cho các sprite. Hoặc có thể tự dùng chuột để xác định các vùng bao này.



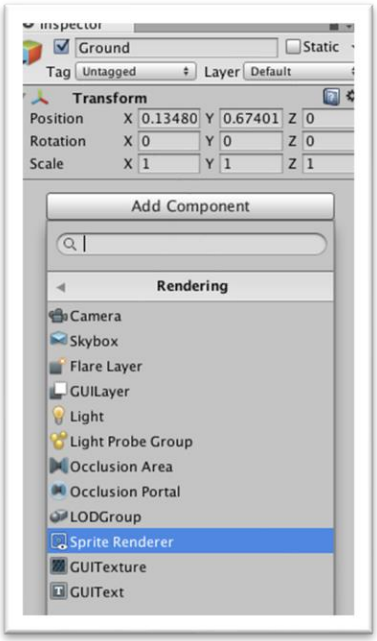

Hình 3.20: Sprite editor

Sau đó nhấn nút Apply ở cửa sổ Sprite Editor để áp dụng.

Vậy là đã tạo xong hai loại sprite bằng Unity.

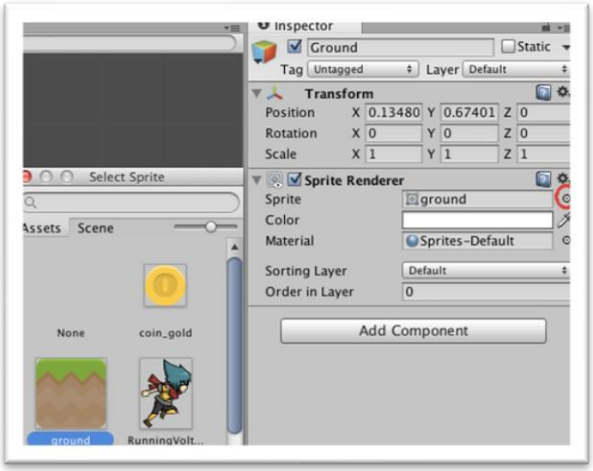
*3.4.2.3. Thêm Sprite render cho Empty GameObject*

Tạo một Empty GameObject, đặt tên là Ground. Sau đó chọn đối tượng này, ở cửa sổ Inspector chọn Add Component sau đó chọn Rendering, chọn Sprite Render.




Hình 3.21: Inspector

Tiếp theo, ở mục Sprite Render ở cửa sổ Inspector, chọn Sprite để vẽ (Nút khoan tròn đỏ)




Hình 3.22: Chọn Sprite

Mách nước: có thể kéo thả trực tiếp từ thư mục sprite vào cửa sổ Scenes.

### 3.4.3. Animation

Một animation là một hình ảnh động mô tả một đối tượng nào đó trong game.

Ví dụ: có thể là một chiếc xe đang chạy, hay một nhân vật đang đi....

Một animation trong Unity có thể bao gồm nhiều hành động, một hành động như vậy gọi là một clip.

Ví dụ: một nhân vật có thể có các hành động đi, đứng, nhảy....

Có hai kỹ thuật để tạo animation (cả 2D và 3D): đó là kỹ thuật key frame và kỹ thuật skeletal hay spine.

#### *3.4.3.1. Kỹ thuật tạo animation*

*** Kỹ thuật key frame**

Đối với kỹ thuật key frame, sử dụng một sprite cho một key frame của hành động.

Hình 3.23: Mỗi sprite là một keyframe

Để tạo ra chuyển động, sẽ vẽ một key frame tại thời điểm đầu và thay đổi tuần tự các key frame sau, sẽ có được một animation.

Đây là phương pháp đơn giản nhất để tạo chuyển động, nhưng lại tốn kém về bộ nhớ, phải tốn nhiều sprite cho nhiều chuyển động khác nhau.

*** Kỹ thuật skeletal hay spine hay bộ xương**

Đối với kỹ thuật này, chia đối tượng ra thành nhiều sprite, mỗi sprite là một bộ phận của đối tượng (giống như 1 khúc xương của bộ xương). Để tạo ra một key frame mới, sẽ thay đổi các sprite về vị trí, độ lớn, xoay của các sprite thành phần có liên quan đến chuyển động. Sau đó kết hợp các key frame lại với nhau như kỹ thuật key frame để tạo thành các animation.

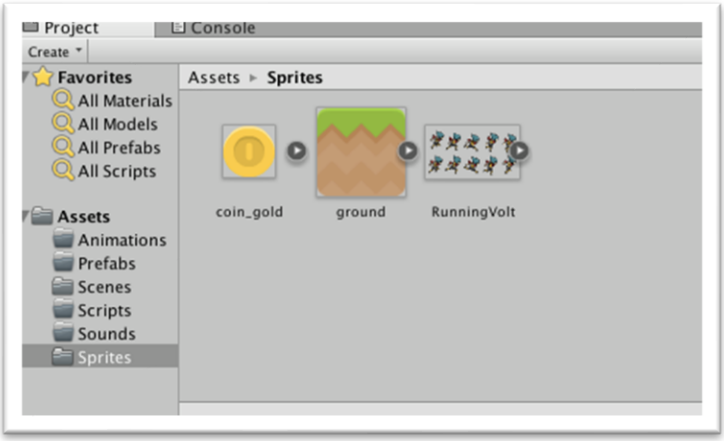
Hình 3.24: 2D Spine

Cách này có vẻ tốn thời gian hơn, nhưng lại rất là hiệu quả, đặc biệt là tiết kiệm được nhiều bộ nhớ.

*3.4.3.2. Tạo animation bằng Unity*

Do sự giới hạn về tài nguyên và thời gian nên trong bài viết này mình chỉ hướng dẫn cách tạo animation theo kỹ thuật Key Frame. Cách tạo animation theo skeletal cũng tương tự.

Từ phần trước  đã tạo được những Sprite cơ bản như sau:




Hình 3.25: Folder Sprites của Assets

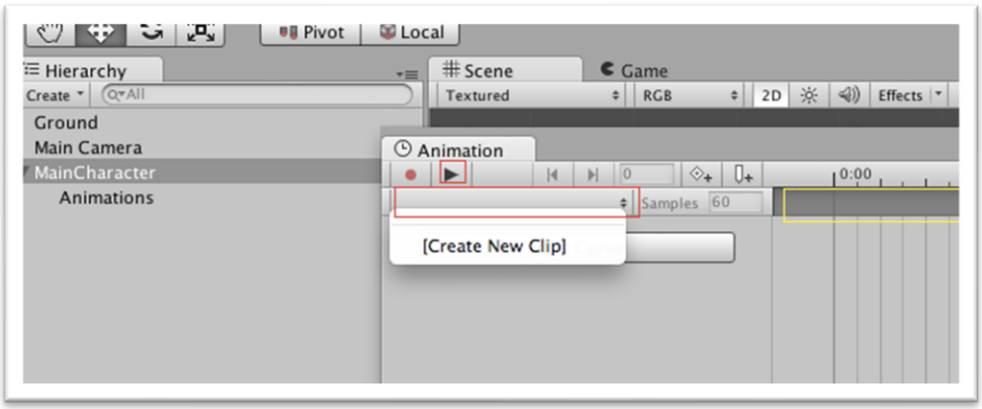
**Bước 1:** Tạo một Empty GameObject đặt tên là MainCharacter (Parent Object)

**Bước 2:** Tạo một đối tượng Empty GameObject nữa, đặt tên là Animations, là đối tượng con của MainCharacter. (Đối tượng con nên đặt ở vị trí 0,0,0).

**Bước 3:** Thêm Sprite Render cho đối tượng con Animations vừa tạo. Rồi chọn sprite hiển thị mặc định cho Animation này. Kết quả như sau:

Hình 3.26: Kết quả Animation

**Bước 4:** Chọn đối tượng MainCharacter ở cửa sổ Hierarchy, rồi chọn Menu -> Window -> Animation




Hình 3.27: Một cửa sổ Animation editor hiện ra như sau

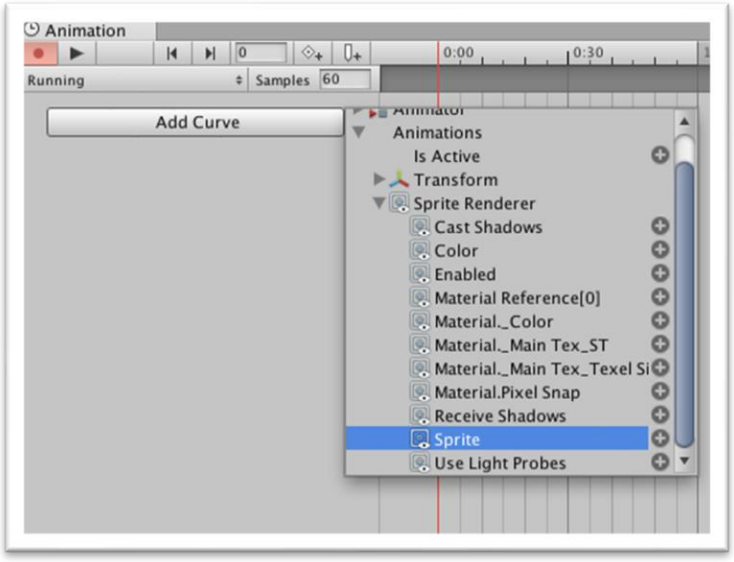
* Hình chữ nhật đỏ nhỏ: nút play để xem trước animation

* Hình chữ nhật đỏ lớn: danh sách các clip hiện thời của aniamtion

* Hình chữ nhật vàng: thanh key frame

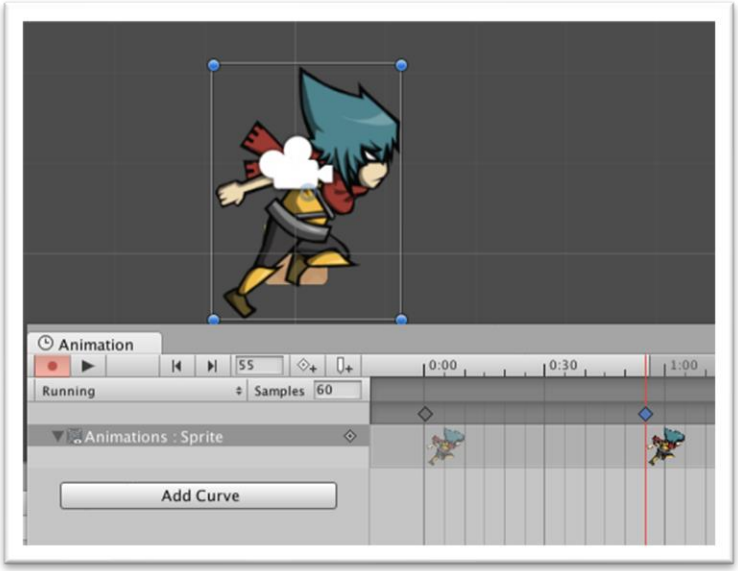
Đầu tiên sẽ click vào danh sách clip rồi chọn Create New Clip, đặt tên clip là Running, rồi save lại ở thư mục Animations của Assets.

**Bước 5:** Ở cửa sổ Animation Editor, chọn Add Curve, chọn Animations (Đối tượng con của đối tượng MainCharacter) chọn Sprite Render, chọn Sprite.



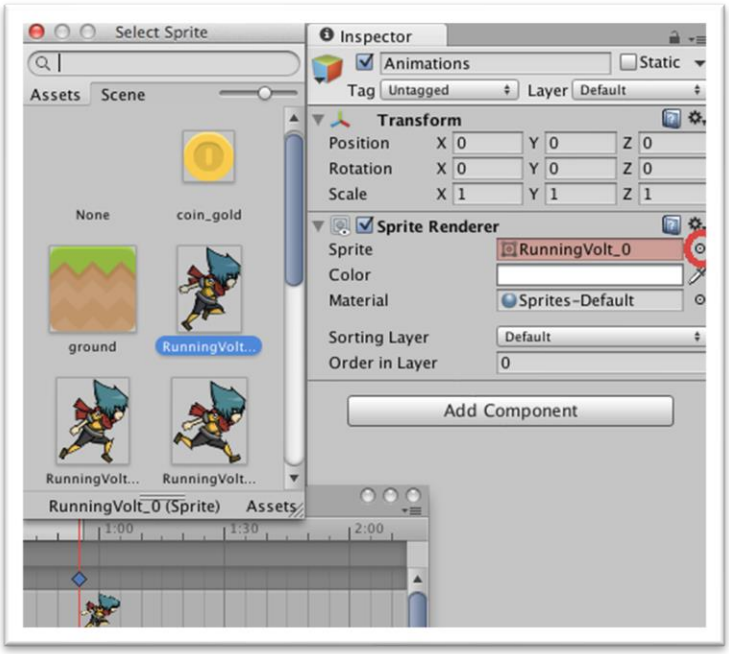

Hình 3.28: 2 Key Frame

Kết quả như sau, mặc định sẽ tạo ra tối thiểu là 2 Key Frame.



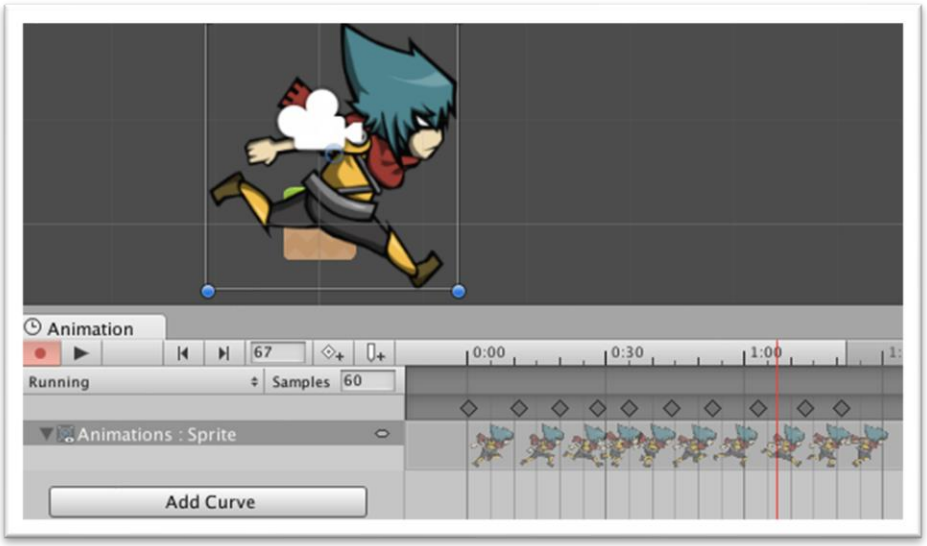

Hình 3.29: 2 Key Frame

Tiếp theo chọn key frame thứ 2, ở cửa sổ Inspector, ở component Sprite Render, tiến hành đổi sprite khác (RunningVolt1 thay vì RunningVolt0). Chọn nút được bao quanh bởi ô tròn đỏ, rồi chọn Sprite khác từ cửa sổ mới hiện ra.



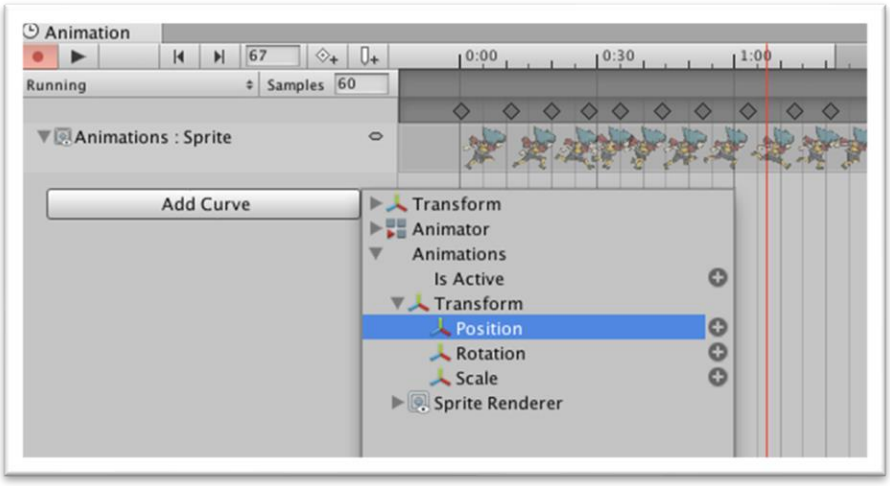

Hình 3.30: 2 Key Frame

Bây giờ, chỉ cần click đúp vào thanh Key Frame (Thanh có ô vuông màu vàng ở hình trước) để thêm các key frame và kéo thả các key frame sao cho thời gian phù hợp để có được chuyển động cần thiết.



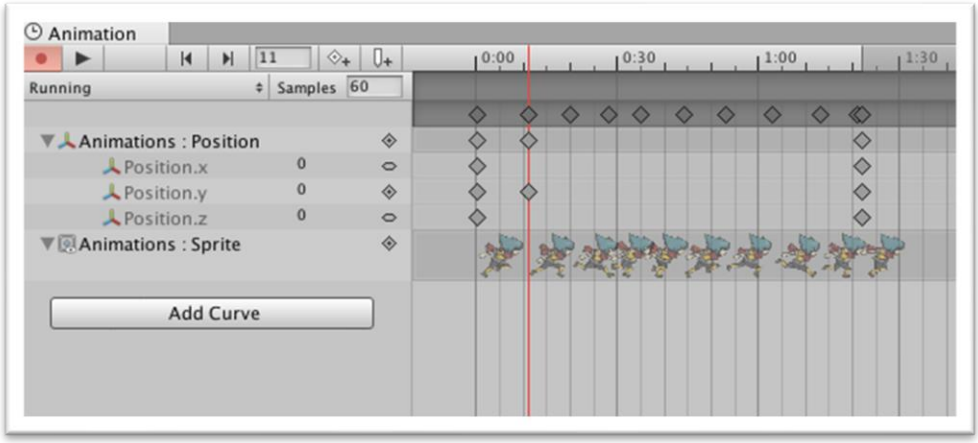

Hình 3.31: 2 Key Frame

Ngoài thay đổi sprite có thể thay đổi Transform (Translate, Scale, Rotation) cho sprite tại mỗi key frame, bằng cách thêm Curve Transform cho đối tượng Animation như hình:




Hình 3.32: Transform

Sau đó chỉ việc chọn các key frame, rồi đặt các giá trị transform cho phù hợp theo ba trục x, y, z.




Hình 3.33: Transform

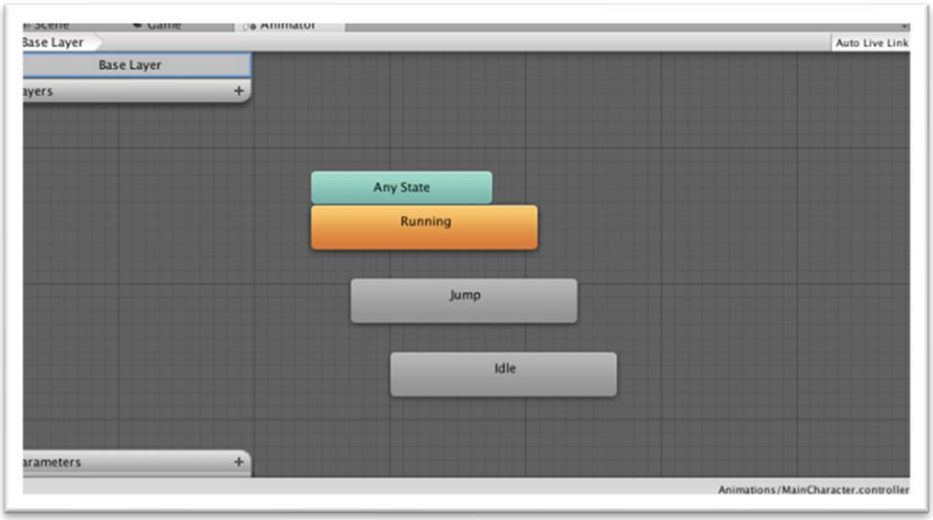
*3.4.3.3. Điều khiển các hành động nhân vật - Animator*

Ở phần trên  đã tìm hiểu cách tạo các clip hay các hành động của một animation. Với một animation như vậy sẽ có một Controller (MainCharacter.controller) đi kèm theo.


Hình 3.34: Animator

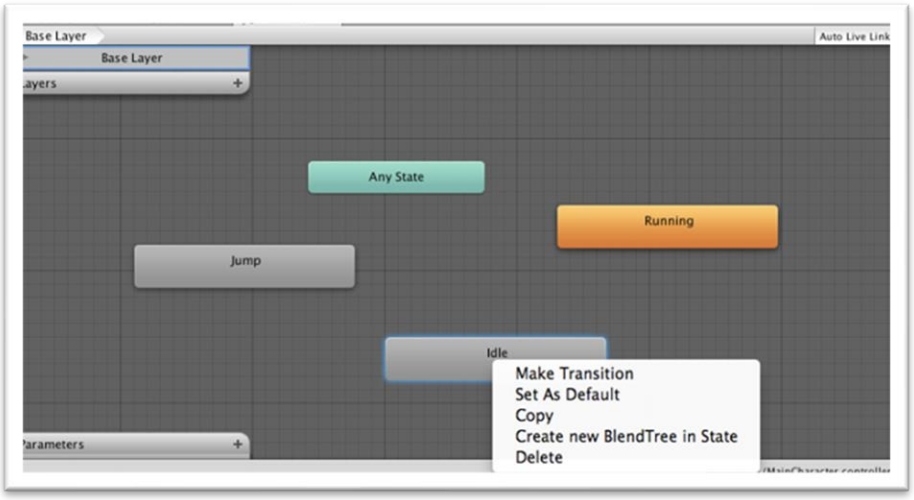
Ở cửa sổ Hierarchy chọn đối tượng MainCharacter, chọn Menu, chọn Window, chọn Animator, cửa sổ Animator sẽ xuất hiện như sau:

(Hoặc  có thể click đúp vào MainCharacter.controller cũng có kết quả tương tự.)



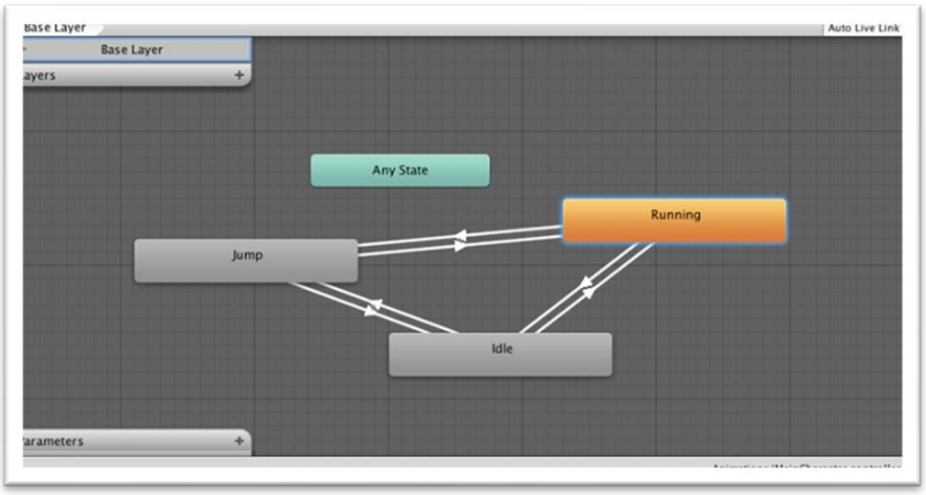

Hình 3.35: Danh sách các state, mỗi state tương ứng với một clip

Click chuột phải vào state Idle, chọn Set Default để thiết lập state mặc định cho đối tượng.




Hình 3.36: State mặc định

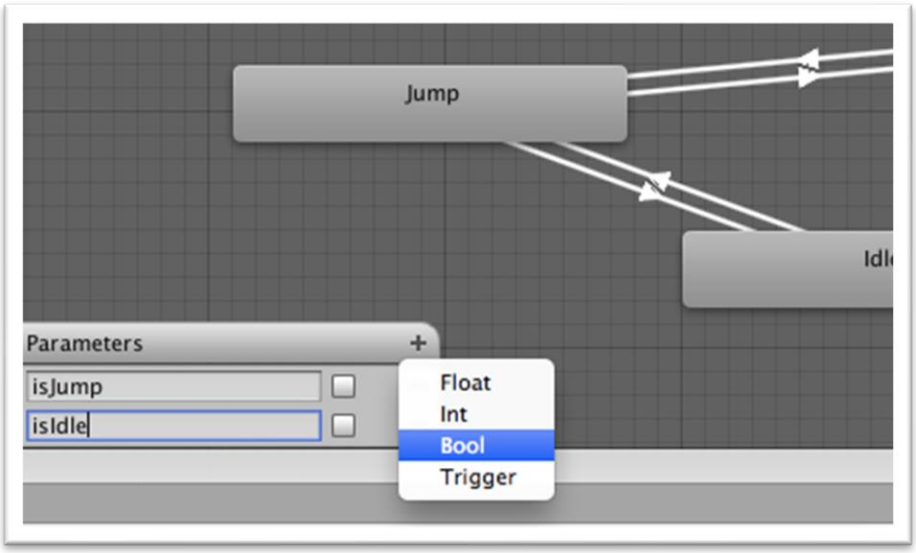
Chọn Make Transition, sau đó đưa chuột đến trạng thái đích. Với mỗi transition vừa tạo, có nghĩa rằng nhân vật từ trạng thái hiện tại có thể chuyển đổi trực tiếp qua trạng thái đích. Kết quả thu được gọi là máy trạng thái hay sơ đồ chuyển đổi trạng thái.




Hình 3.37: Sơ đồ chuyển đổi trạng thái

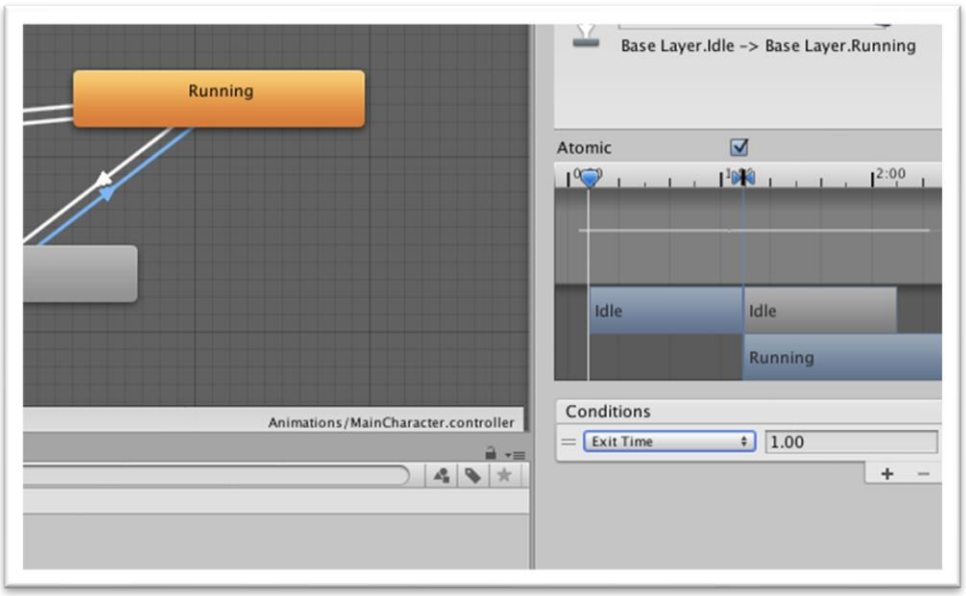
Khi đối tượng được load lên, trạng thái mặc định sẽ được thiết lập, vậy lúc nào thì sẽ chuyển qua trạng thái khác ??  sẽ tạo thêm các tham số, và dựa vào giá trị các tham số này để chuyển đổi các trạng thái.

Sẽ tạo ra 3 tham số kiểu bool là isJump, isIdle, isRunning để điều khiển.



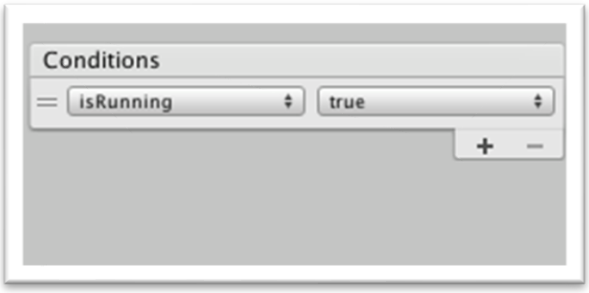

Hình 3.38: Parameters

Để thiết lập điều kiện cho một transition, click chọn transition đó (transition được chọn chuyển qua màu xanh), ở cửa sổ Inspector, mục thuộc tính Conditions (điều kiện) sẽ thiết lập giá trị của các tham số, để xác định lúc nào thì chuyển trạng thái.




Hình 3.39: Chuyển trạng thái

Ở đây, chọn isRunning = true.

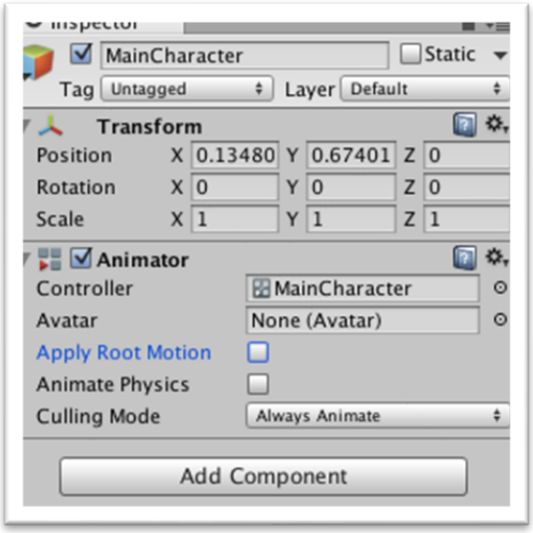



Hình 3.40: Conditions

Có thể thêm các điều kiện khác (trường hợp chuyển đổi phụ thuộc nhiều điều kiện) bằng cách nhấn dấu cộng hoặc dấu trừ để bỏ bớt một điều kiện.

Vậy mỗi khi set giá trị tham số isRunning = true, trạng thái nhân vật sẽ chuyển sang Running. Tương tự thiết lập isIdle, isJump cho các transition khác.

Cuối cùng, ở cửa sổ Hierarchy, chọn MainCharacter, ở cửa sổ Inspector, mục Animator, bỏ chọn Apply Root Motion.

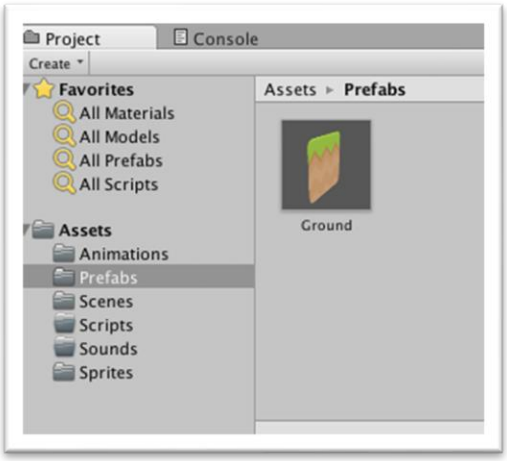



Hình 3.41: Inspector

### 3.4.4. Prefab

Prefab cho tạo ra các bản sao nhanh của một đối tượng mà không cần thiết lập lại các giá trị khởi tạo của một đối tượng nào đó ngoài trừ các giá trị transform (vị trí, tỉ lệ, quay).

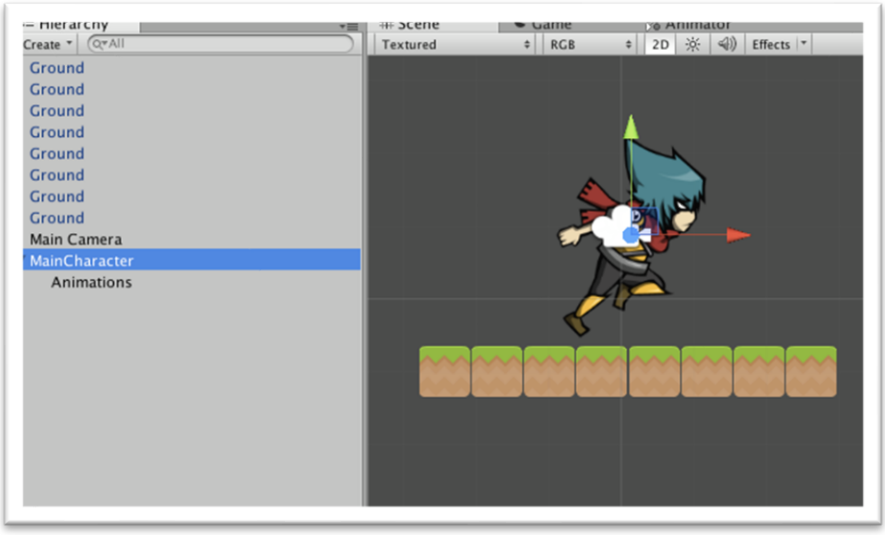
Để tạo một Prefab cho một đối tượng nào đó, chỉ cần kéo thả đối tượng đó ở cửa sổ Hierarchy xuống thư mục Prefab (nếu chưa có thì tạo mới) trong cửa sổ Project của . Sau này muốn sử dụng chỉ việc kéo các Prefab này trở lại cửa sổ Scene.




Hình 3.42: Prefab cho đối tượng Ground

Bây giờ sẽ kéo thả Prefabs này tạo thành một đường thẳng, và đặt lại vị trí đối tượng MainCharacter ở vị trí nằm trên đường thẳng này như hình vẽ.

**Chú ý:** là đặt vị trí của đối tượng MainCharacter, chứ không phải đối tượng con Animation của MainCharacter. Vị trí của đối tượng Animation bây giờ vẫn là (0, 0) so với đối tượng Maincharacter (vì là đối tượng con) nghĩa là đối tượng Animation này nằm tại tâm của đối tượng MainCharacter.




Hình 3.43: Bố cục game đơn giản

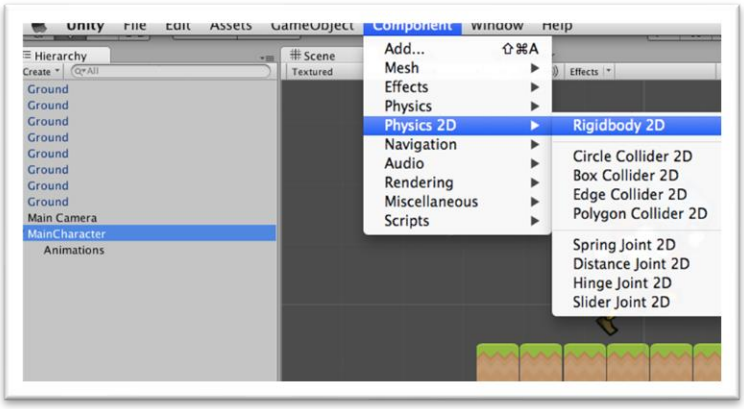
Tip: Để thiết lập giá trị y của các Ground bằng nhau, chọn tất cả rồi thiết lập giá trị y ở cửa sổ Inspector.

### 3.4.5. Thành phần vật lý và xử lý va chạm

*3.4.5.1. Thêm thành phần vật lý (Physics 2D)*

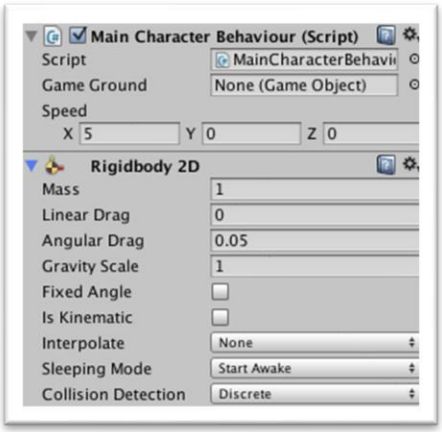
Thành phần vật lý hỗ trợ các thao tác về vật lý như: tác dụng lực, trọng lực trái đất, ma sát...

Để thêm thành phần vật lý làm như sau: ở Hierarchy, chọn đối tượng MainCharacter (đối tượng cần thêm) / Menu /Component / Physics 2D / Rigidbody 2D




Hình 3.44: Ridbody 2D

Sẽ thấy ở cửa sổ Inspector khi chọn đối tượng MainCharacter sẽ thêm một Component nữa là Rigidbody 2D như sau:




Hình 3.45: Add Component

Một số giá trị cần lưu ý sau:

Mass: là khối lượng của đối tượng

Linear Drag: Hệ số ma sát của vật đối với chuyển động kéo.

Angular Drag: Hệ số ma sát của vật đối với chuyển động quay

Gravity scale: giống như hệ số G trong vật lý (~ 9.81), chỉ sự ảnh hưởng của lực hút trái đất. Có thể đặt = 0, tức là không ảnh hưởng bởi lực hút trái đất.

Is Kinematic: loại bỏ tác dụng vật lý ra khỏi đối tượng, thường sử dụng với các đối tượng như tường, nền ...

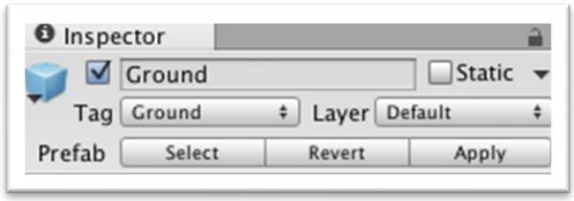
Fixed Angle: Đối tượng luôn nằm một góc cố định. Không thay đổi khi tương tác vật lý rigidbody2D.AddForce(Vector2 f)

Sau khi thêm thành phần vật lý vào, nhấn nút play để xem demo, sẽ thấy đối tượng sẽ từ từ rơi xuống (do Gravity scale > 0).

Tương tự, sẽ thêm thành phần vật lý cho đối tượng Ground, và đặt thuộc tính cho nó là Is Kinematic để làm nền.

**Chú ý:**

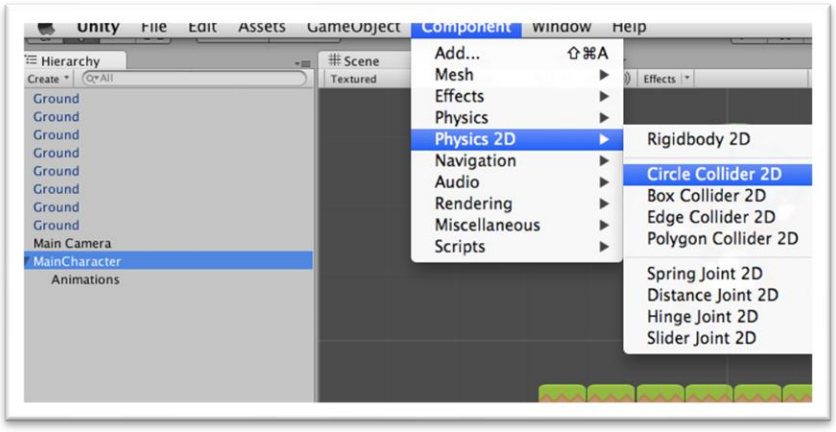
Để có tác dụng cho tất cả các Prefab, sau khi thêm hoặc thay đổi bất cứ thuộc tính nào, nhấn Apply.




Hình 3.46: Inspector

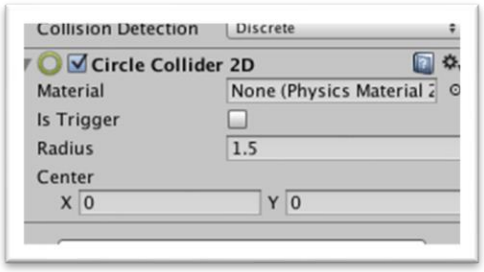
*3.4.5.2. Thành phần xử lý va chạm*

Ở Hierarchy, chọn đối tượng MainCharacter (đối tượng cần thêm) / Menu /Component / Physics 2D / Circle Collider 2D




Hình 3.47: Xử lý va chạm

Đối tượng sẽ có thêm thuộc tính Collider:

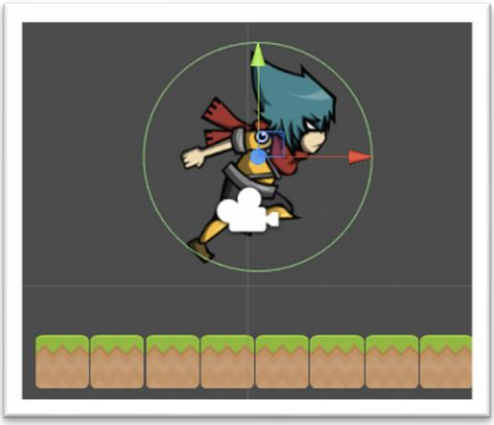



Hình 3.48: Collider

Sẽ chọn tâm và bán kính để xác định vùng xử lý va chạm.

Thuộc tính: Is Trigger: nếu chọn thì đối tượng chỉ dùng để xác định va chạm mà không ảnh hưởng bởi tác động vật lý.

Kết quả sẽ thấy như sau:

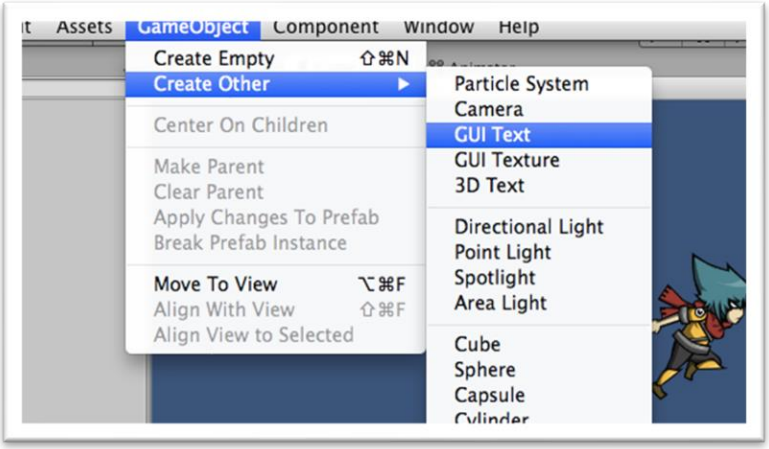


Hình 3.49: Kết quả

Tiếp theo tiến hành thêm thành phần Collider cho đối tượng Ground, lần này sẽ chọn Box Collider 2D thay vì Circle Collier 2D. Các thuộc tính cũng tương tự, có thể chỉnh sửa hình chữ nhật để xác định vùng va chạm.

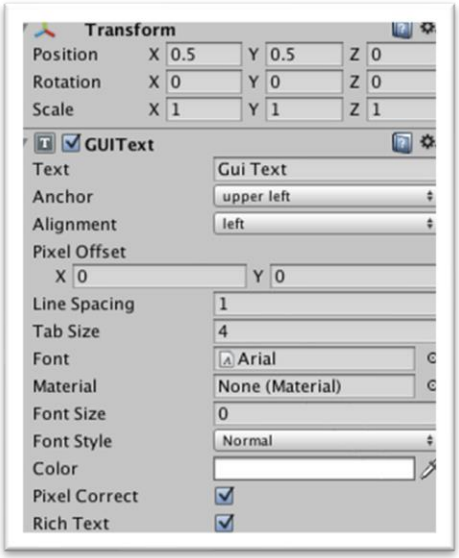
### 3.4.6. Sử dụng text

Từ chọn Menu/Game Object/Create Other/Gui Text



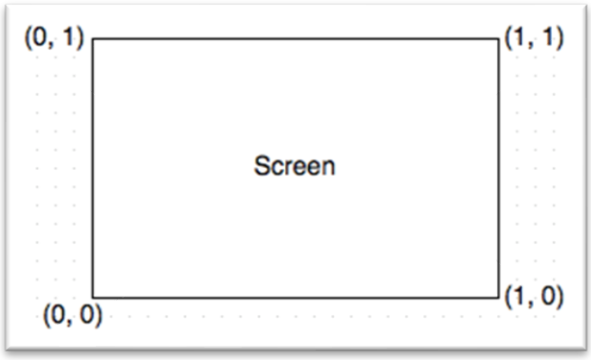

Hình 3.50: Add Text

Sẽ thấy ở cửa sổ Hierarchy thêm một đối tượng tên là GUI Text, chọn đối tượng này sẽ thấy ở cửa sổ Inspector có các thuộc tính như sau:




Hình 3.51: Thuộc tính GUI TEXT

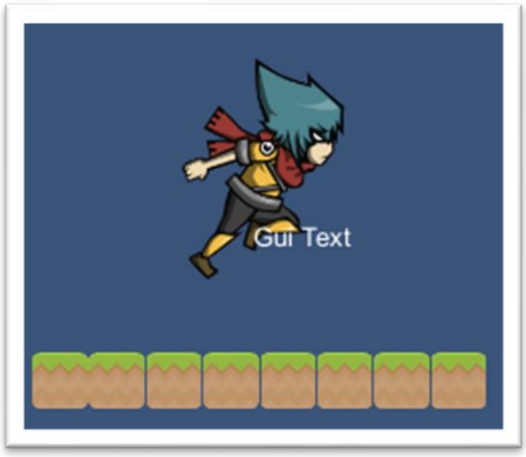
Nhìn vào các thuộc tính có thể dễ dàng đoán được chức năng của từng thuộc tính, một lưu ý quan trọng là giá trị position của GUI Text chỉ nhìn thấy nếu nằm trong đoạn [0,1]. Với quy ước góc như hình vẽ:




Hình 3.52: Position

Có nghĩa rằng nếu đặt text ở vị trí (0.5, 0.5) thì lúc chạy game lên, đoạn text này lúc nào cũng nằm ở giữa màn hình.

Để xem hiển thị trực quan  chuyển qua tab Game, sẽ thấy text được hiển thị đầy đủ và đúng như mô tả ở hình trên.




Hình 3.53: Kết quả

# CHƯƠNG 4: THỰC NGHIỆM VÀ TRIỂN KHAI

## 4.1. **Giới thiệu hệ thống game**

Có 2 loại đối tượng: “chú chim” và các “ống khói”. Nếu không tác động gì, chú chim sẽ rơi tự do. Người chơi cần chạm vào màn hình (trên điện thoại thông minh hoặc máy tính bảng) hoặc gõ phím “khoảng trắng” (space) (trên web hoặc máy tính để bàn) để làm cho chú chim nhảy lên. Mỗi lần chạm hoặc gõ phím “khoảng trắng” thì chú chim sẽ nhảy 1 lần. Trò chơi kết thúc khi chú chim chạm đất hoặc chạm phải bất kì ống khói nào. Mỗi khi vượt qua cặp ống khói trên và dưới thì tích lũy thêm 1 điểm.

## 4.2. **Mô tả hệ thống game**

Flappy Bird thành các đối tượng sau:

1. Ground: Nền đất.

2. Background: Hình nền, là cảnh quan trời mây bãi cỏ phía xa.

3. Bird: Chú chim nhảy

4. Pipe: Ống khói.

5. PlayButton: Nút ấn cho phép bắt đầu trò chơi.

6. GameOver: Dòng chữ thông báo kết thúc trò chơi.

7. Score: Phần hiển thị điểm số.

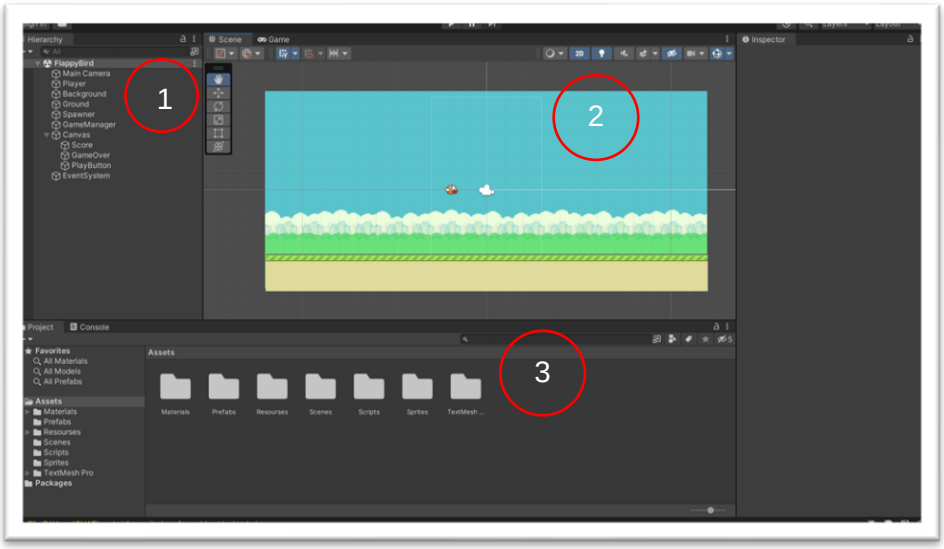
Khi đó, trò chơi Flappy Bird chính là sự chuyển động, tương tác giữa các đối tượng trên.

Các bài toán lập trình chủ yếu bao gồm:

- Bắt và xử lí sự kiện “gõ phím” hoặc “nhấp chuột” để người chơi có thể can thiệp vào chuyển động của đối tượng Bird - chú chim.
- Rơi tự do. Khi không có tác động nào, chú chim - Bird sẽ rơi tự do. Vậy sẽ cần một phương thức C# tương ứng với nhiệm vụ đó. Về phương ngang, thực ra chú chim không thay đổi tọa độ. Chú chim chỉ chuyển động theo phương đứng.

- Sinh ngẫu nhiên các ống khói - Pipe. Đây là phần thử thách của trò chơi, độ khó của trò chơi cũng phụ thuộc vào các thông số (khoảng cách giữa các cặp ống khói, chênh lệch giữa các khe hở trước và sau...) sinh ngẫu nhiên các ống khói - Pipe như thế này.
- Dịch chuyển đối tượng theo thời gian. Cả nền đất - Ground, hình nền -Background, các ống khói - Pipe đều cần chuyển động sang trái theo thời gian với tốc độ khác nhau.

### 4.2.1. Scene




Hình 4.1: Scene

Không gian trò chơi có thể rất lớn, nhưng ngay khi chạy chương trình thì người chơi chỉ nhìn thấy một phần của nó. Unity gọi đó là Scene. Một trò chơi có thể có nhiều Scene.

| Mô tả chi tiết hình 4.1 | | |
|---|---|---|
| **Ký hiệu** | **Chức năng/Nội dung** | **Ý nghĩa** |
| 1 | Hierachy | Cây thư mục phân cấp game |
| 2 | Screen | Màn hình chính hiển thị game |
| 3 | Assets | Các thư mục chưa code cho game |

Bảng 4.1: Bảng mô tả chi tiết Scene

### 4.2.2. Sprite

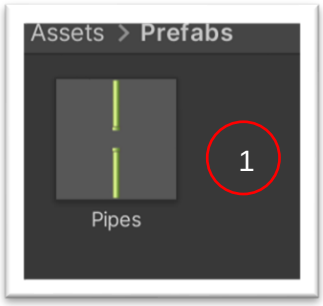

Hình 4.2: Các hình png dùng làm sprite cho các đối tượng trong FlappyBird

| Mô tả chi tiết hình 4.2 | | |
|---|---|---|
| **Ký hiệu** | **Chức năng/Nội dung** | **Ý nghĩa** |
| 1 | Sprite | Nơi chứa các hình ảnh của Game Object |

Bảng 4.2: Bảng mô tả chi tiết Sprite

- Một đối tượng đồ họa 2D sẽ tương ứng với 1 Sprite. Unity hỗ trợ thiết kế các đối tượng đồ họa đó, hoặc có thể thiết kế trên phần mềm khác và kết xuất (export) thành các tập tin (file) hình ảnh và bỏ vào đúng chỗ trong Project.
- Score không có đồ họa tương ứng, do đó sẽ không có Sprite tương ứng với nó. Sẽ biểu diễn Score bằng Text trên Canvas của Unity.
- Chú chim - Bird sẽ có 3 Sprite tương ứng. Khi liên tiếp hiển thị chuỗi 3 hình này, sẽ sẽ tạo ra chuyển động vỗ cánh, làm trò chơi thêm sinh động.

### 4.2.3. Prefabs




Hình 4.3: Prefab

| Mô tả chi tiết hình 4.3 | | |
|---|---|---|
| **Ký hiệu** | **Chức năng/Nội dung** | **Ý nghĩa** |
| 1 | Sprite | Nơi chứa các hình ảnh của Game Object |

Bảng 4.3: Bảng mô tả chi tiết Prefabs

Một số đồ họa phức tạp gồm nhiều đồ họa đơn giản hợp thành thường đi  nhau thì nên gom vào một nhóm, gọi là Prefab. Trong Flappy Bird, thấy các ống khói - Pipe thường đi thành cặp, 1 ống ở dưới, 1 ống ở trên, ở giữa có khe hở. Chính vì vậy, sẽ gom cụm và tạo thành 1 Prefab tương ứng cho nó.

### 4.2.4. Material


Hình 4.4: Material

| Mô tả chi tiết hình 4.4 | | |
|---|---|---|
| **Ký hiệu** | **Chức năng/Nội dung** | **Ý nghĩa** |
| 1 | Materials | Các đồ họa này sẽ được nhân bản, nối tiếp nhau, cuộn tròn lại. |

Bảng 4.4: Bảng mô tả chi tiết Material

Để ý thấy rằng nền đất - Ground và hình nền Background chỉ có kích thước hữu hạn, nhưng lại dịch trái liên tục trong suốt trò chơi.

Unity hỗ trợ các chuyển động đồ họa kiểu này bằng Texture trong Material. Có thể hiểu nôm na là các đồ họa này sẽ được nhân bản, nối tiếp nhau, cuộn tròn lại. Quá trình dịch trái sẽ biến thành quá trình xoay sang trái và chiếu lên Scene. Vì thế, thực chất, đối tượng nền đất - Ground và hình nền - Background của Flappy Bird sẽ tương ứng với Material Ground và Material Background.

### 4.2.5. Canvas và Button, Image, Text

Canvas là là một cấu phần giao diện (UI component) mà có thể đặt các cấu phần giao diện khác có chung nhiều thông số cài đặt (setting) lên. Cũng có thể hiểu Canvas

là một cách gom nhóm nhiều cấu phần giao diện nhỏ lẻ. Trong Flappy Bird, cần Canvas để đặt:

- Cấu hình giao diện Button ứng với PlayButton
- Cấu hình giao diện Image ứng với GameOver
- Cấu hình giao diện Text ứng với Score

Đến đây có bảng tổng hợp sau:

| Đối tượng trong FlappyBird | Đồ Họa | Các cấu phần tương ứng trong Unity |
|---|---|---|
| Ground<br>Background | Ground.png<br>Background.png | Material<br>Material |
| Bird | Bird_01.png<br>Bird_02.png<br>Bird_03.png | Sprite |
| Pipe | Pipe.png | Prefab |
| PlayButton | PlayButton.png | Button trên Canvas |
| GameOver | GameOver.png | Image trên Canvas |
| Score | | Text trên Canvas |

Bảng 4.5: Các đối tượng trong Flappy Bird và cấu phần tương ứng trong Unity

### 4.2.6. Các cấu phần vật lí trong Unity

Nếu các cấu phần giao diện đồ họa là cái đẹp đẽ trưng ra trước mặt người chơi,thì khối hình vật lí dùng trong các logic tính toán va chạm, tác động, rơi tự do, bay lượn... lại thường là một cấu phần vật lí (Physics component) được khai báo kèm theo cấu phần giao diện đó. Ở đây, cấu phần vật lí đó trong Unity là Rigidbody. Các điều khiển vật lí trên Unity sẽ gắn với Rigidbody.

- Ứng với Bird là một Circle Collider 2D và một Rigidbody 2D với thuộc tính Body Type là Kinematic.
- Ứng với Ground là một Box Collider 2D
- Ứng với mỗi Pipe là một Box Collider 2D. Một cặp Pipe sẽ có 1 cặp BoxCollider 2D tương ứng. Khoảng trống ở giữa cặp Pipe cũng có một BoxCollider 2D. Sẽ cần

Collider này để kích hoạt quá trình tính điểm khi Circle Collider 2D của Bird vượt qua.

| Đối tượng trong FlappyBird | Đồ họa | Các cấu phần tương ứng trong Unity |
|---|---|---|
| Ground | Ground.png | MaterialBox Collider 2D |
| Background | Background.png | Material |
| Bird | Bird_01.png<br>Bird_02.png<br>Bird_03.png | Sprite<br>Rigidbody<br>2D Circle Collider 2D |
| Pipe | Pipe.png | Prefab3Box Collider 2D |
| PlayButton | PlayButton.png | Button trên Canvas |
| GameOver | GameOver.png | Image trên Canvas |
| Score | | Text trên Canvas |

Bảng 4.6: Các đối tượng trong Flappy Bird và cấu phần tương ứng trong Unity

### 4.2.7. Điều khiển đối tượng bằng C#

Trong Flappy Bird

- Ứng với Bird sẽ có Player.cs
- Ứng với Ground sẽ có Ground.cs
- Ứng với Background sẽ có Parallax.cs
- Ứng với mỗi cặp ống khói - Pipe sẽ có Pipes.cs

Ngoài ra:

- Để điều khiển việc sinh các cặp Pipe mới ngẫu nhiên và hủy các cặp Pipecũ (khi đã vượt ra ngoài màn hình bên trái) có thêm Spawner.cs.
- Mỗi trò chơi sẽ có một tập tin C# đặc biệt gọi là GameManager đóng vai tròkhởi tạo và điều khiển các quá trình tổng thể như PlayButton,GameOver,Score

| Đối tượng trong FlappyBird | Đồ họa | Các cấu phần tương ứng trong Unity | Mã C# điều khiển tương ứng |
|---|---|---|---|
| Ground | Ground.png | MaterialBox | Ground.cs |

| | | | |
|---|---|---|---|
| | | Collider 2D | |
| Background | Background.png | Material | Parallax.cs |
| Bird | Bird_01.png<br>Bird_02.png<br>Bird_03.png | Sprite<br>Rigidbody<br>2D Circle Collider 2D | Player.cs |
| Pipe | Pipe.png | Prefab3Box Collider 2D | Pipes.cs<br>Spawner.cs |
| PlayButton | PlayButton.png | Button trên Canvas | GameManager.cs |
| GameOver | GameOver.png | Image trên Canvas | GameManager.cs |
| Score | | Text trên Canvas | GameManager.cs |

Bảng 4.7: Các đối tượng trong Flappy Bird và cấu phần tương ứng trong Unity

### 4.2.8. Màn hình chính


Hình 4.5: Màn hình chính

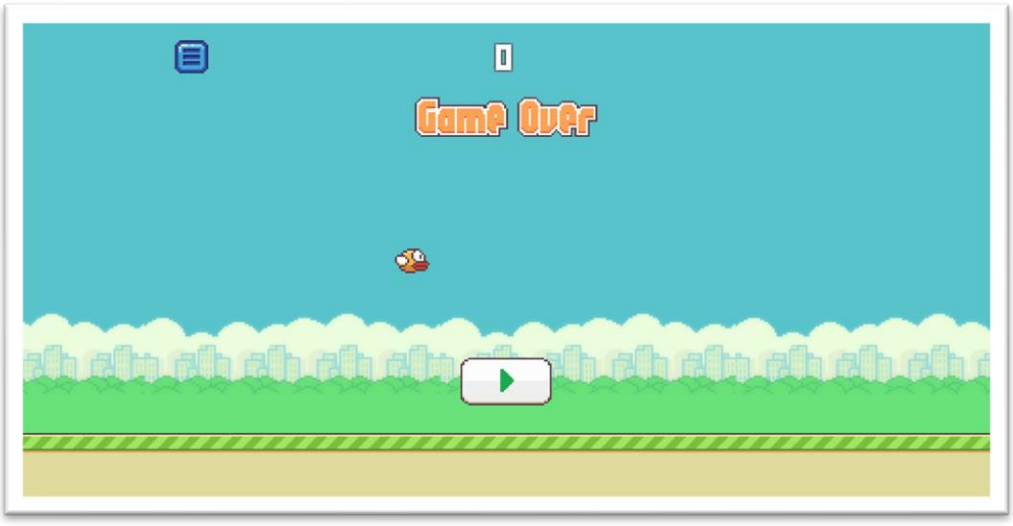
Đây là giao diện khi mới vào game, sẽ có 2 nút button cho việc vào Scene tiếp theo hoặc thoát game khi người chơi muốn thoát.

### 4.2.9. Màn hình lựa chọn màn chơi


Hình 4.6: Màn hình lựa chọn màn chơi

Khi nhấn Start của màn hình chính sẽ hiện ra Scene lựa chọn màn chơi level 1 và level 2 tương ứng với mức độ trò chơi và nút Back khi nhấn vào sẽ quay trở lại màn hình chính.

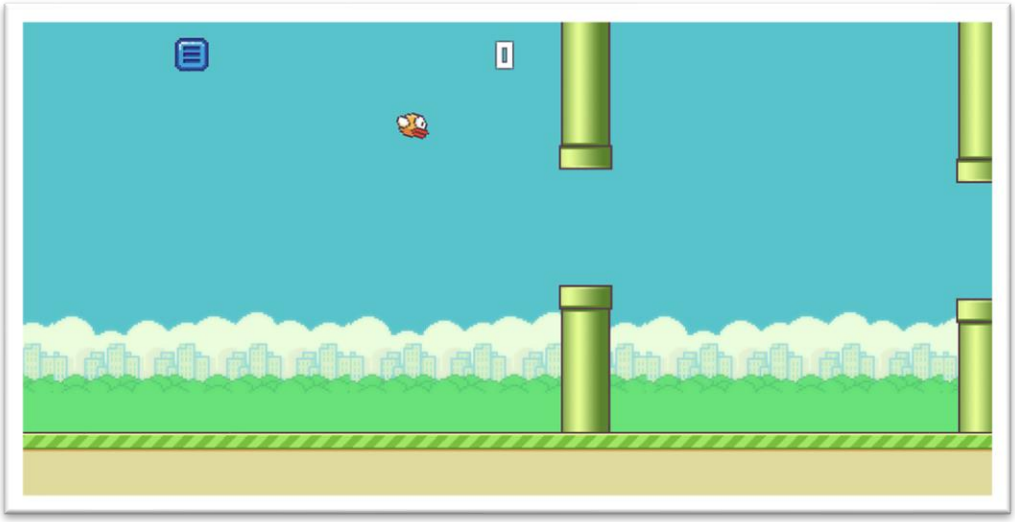
### 4.2.10. Màn hình game level 1




Hình 4.7: Màn hình game level 1

Đây là giao diện khi mới bắt đầu vào game và khi người chơi bấm vào nút button trên màn hình thì game sẽ bắt đầu và còn nút button màu xanh khi nhấn vào sẽ quay trở lại màn hình lựa chọn màn chơi.

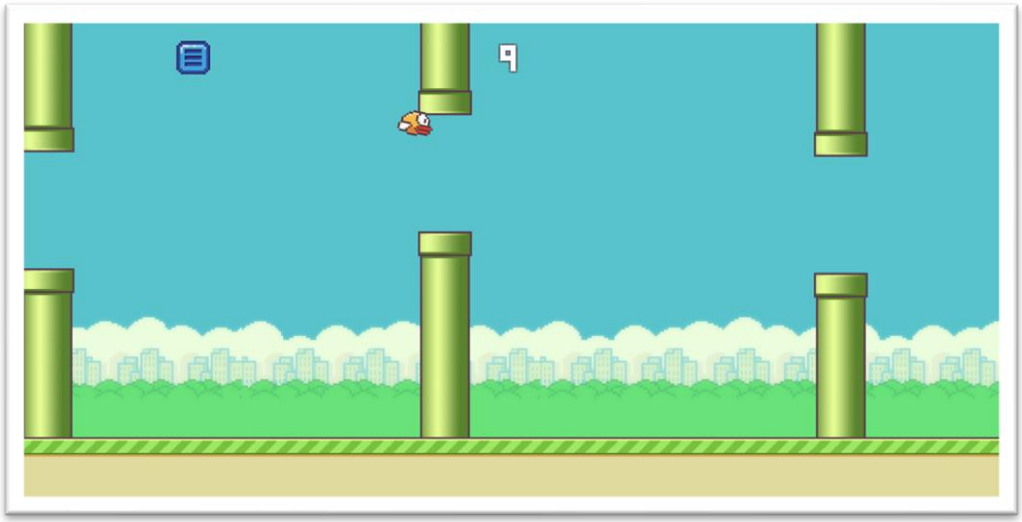
### 4.2.11. Play Screen level 1



Hình 4.8: Tap Screen

Chức năng chính của game là khi người dùng thao tác chạm vào màn hình, sẽ kích hoạt khiến chú chim của  nhảy lên, vượt qua các chướng ngại vật 1 cách khéo léo để đạt điểm cao nhất

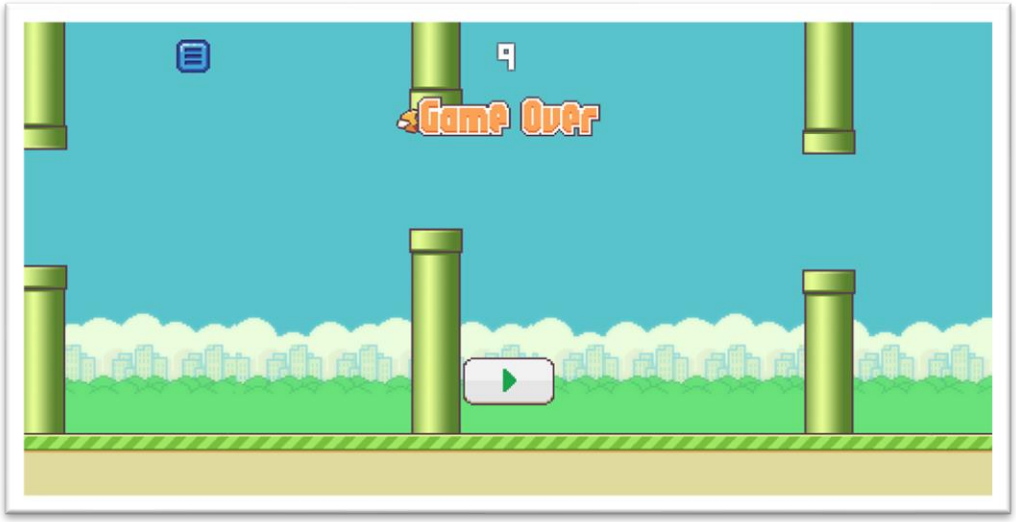
### 4.2.12. Score level 1



Hình 4.9: Score

Điểm được hiển thị ở giữa phần trên cùng của game để người dùng tiện theo dõi. Và sau khi game over thì sẽ tự động trở lại màn hình chính

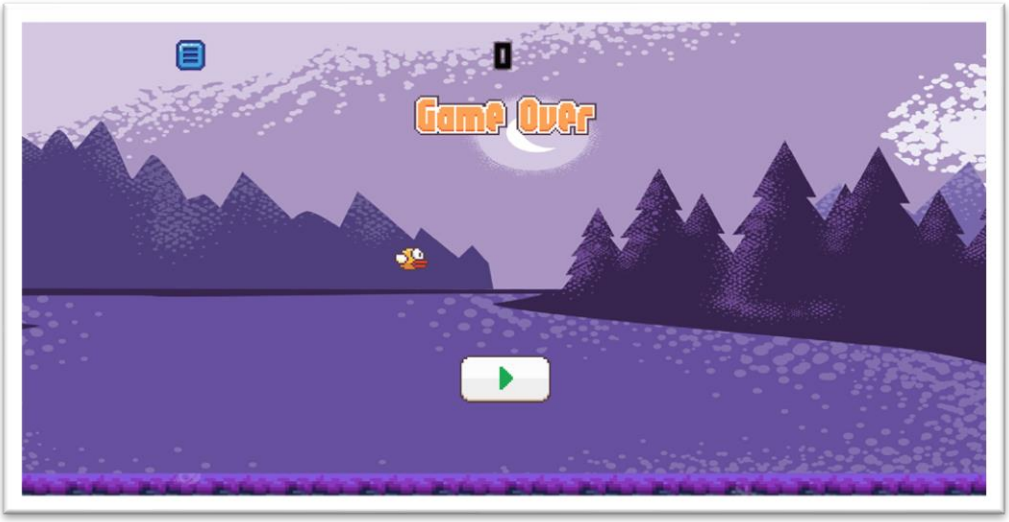
### 4.2.13. Game Over level 1




Hình 4.10: Game Over

Khi chạm vào ống khói hoặc dưới đất thì trò chơi sẽ kết thúc.

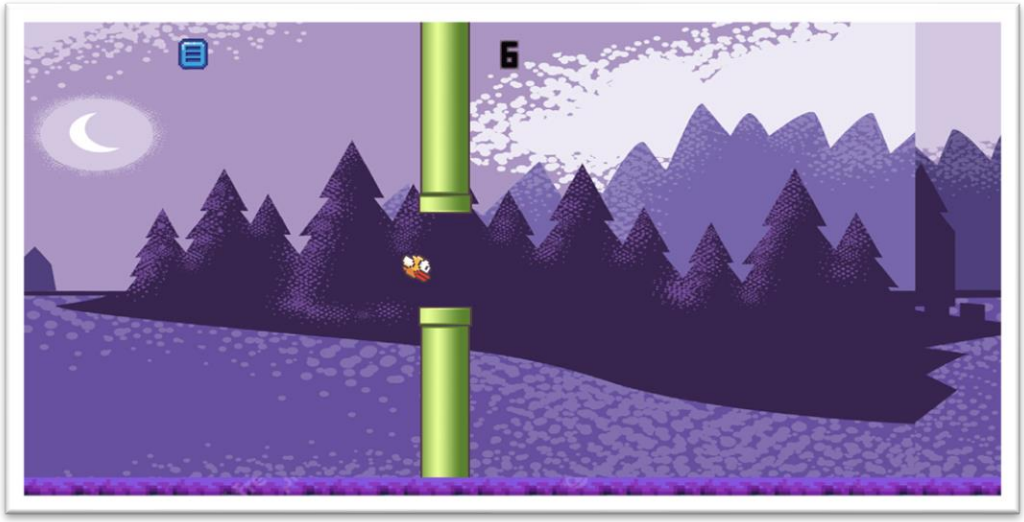
### 4.2.14. Màn hình game level 2




Hình 4.11: Màn hình game level 2

Đến với level 2 sẽ thay đổi background và ground.

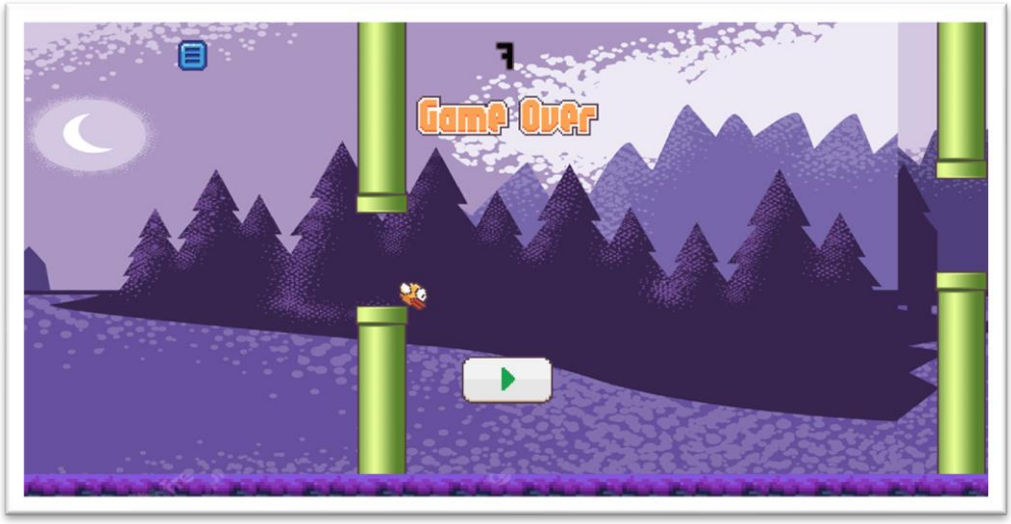
### 4.2.15. Play Sceen level 2




Hình 4.12: Play Sceen level 2

Có sự thay đổi giữa độ khó giữa 2 màn chơi. Khoảng cách 2 ống khói thu hẹp lại làm cho nhân vật sẽ khó tích điểm, ngoài ra tốc độ game tăng lên so với level 1.

### 4.2.16. Score level 2




Hình 4.13: Score level 2

Điểm số cũng đã thay đổi màu sắc để phù hợp với background.

## 4.3. **Triển khai**

### 4.3.1. Cấu hình máy

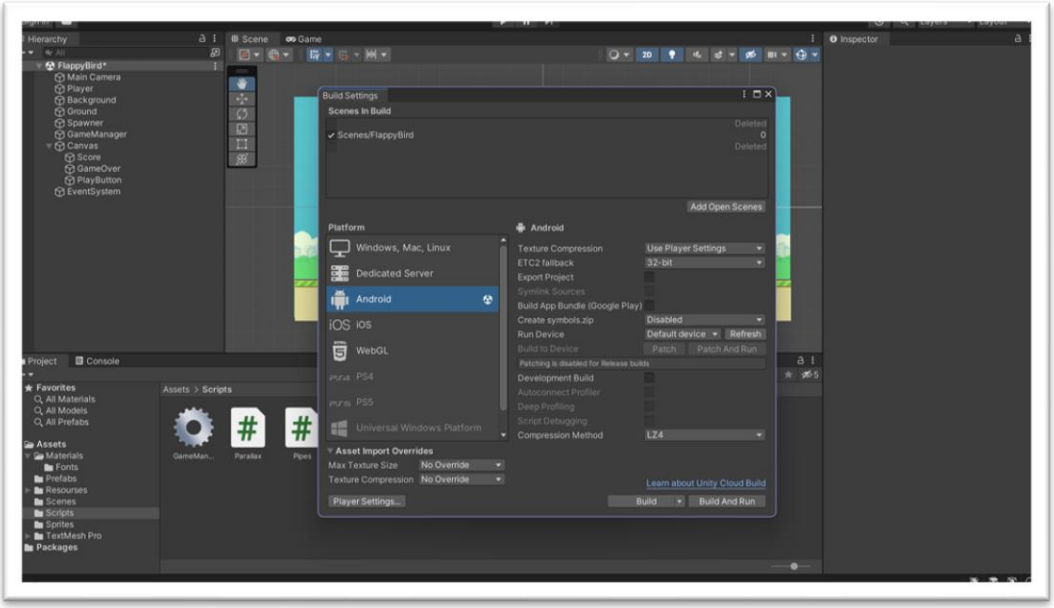
Cấu hình máy phù hợp cho tất cả dòng máy hiện nay, bao gồm cả Android và IOS, người dùng có thể chơi được trên cả PC hoặc Laptop, game rất nhẹ chỉ có 20 MB.

### 4.3.2. Hệ điều hành

Game có thể xuất ra file apk nên hiện tại Andoid 4 trở lên là đã có thể cài đặt

### 4.3.3. Cách thức cài đặt (Ví sụ: Cài đặt trên PC, Laptop, Mobile, ....)

Nhấn vào File -> Buiding Setting, chọn vào những Scene mà muốn xuất và tiếp theo chọn Platform nào mà muốn cài đặt




Hình 4.14: Build Setting

# KẾT LUẬN

**Kết quả đạt được**

Game Flappy Bird 2D được xây dựng bằng công nghệ Unity sẽ giúp cho người lập trình dễ dàng thao tác và hiện tại công nghệ đang ở dạng Open Scoures. Game Flappy Bird 2D được tải cũng như sử dụng mượt mà.

Game Flappy Bird 2D này gồm có các chức năng như sau:

- Di chuyển: Chức năng này giúp cho các nhân vật trong game di chuyển một cách linh hoạt.
- Tích điểm: Cộng thêm điểm khi vượt qua vật cản trở.
- Bắt đầu game: Khi nhấn vào sẽ bắt đầu vào trò chơi và khi kết thúc sẽ xuất hiện nút replay nếu nhấn vào sẽ bắt đầu lại trò chơi.
- Phân cảnh: Tạo ra Scene chạy liên tục sang bên phải.
- Tạo vật cản: Khi chạm vào vật cả sẽ kết thúc game hoặc khi vượt qua sẽ tích điểm
- Animation cho nhân vật trong game: Chú chim sẽ đập cánh liên tục khi di chuyển.

**Hướng phát triển**

- Tiếp tục thiết kế một bản thông báo Best Score khi trò chơi kết thúc
- Âm thanh trong game..
- 1 Scene màn hình đăng nhập trước khi vào game.

# TÀI LIỆU THAM KHẢO

[1] https://vtc.edu.vn/lap-trinh-game-unity/ ngày tham khảo 17/06/2023

[2] https://www.facebook.com/notes/hội-lập-trình-viên-game-đà-nẵng/làm-game-2d-bằng-unity-phần-1-các-khái-niệm-cơ-bản/232376550267948 ngày tham khảo 30/7/2023

[3] https://www.studocu.com/vn/document/truong-dai-hoc-da-lat/quan-he-doi-ngoai-viet-nam/main-lap-trinh-flappy-bird-bang-c/57574864 ngày tham khảo 01/09/2023

# PHỤ LỤC

Link drive phần source code:

https://drive.google.com/drive/folders/1r_un1CZ5nXUOWK83Zhs3rh4BeHr4AWRN?usp=sharing