

# **Bài 3: Animation và điều khiển hành động nhân vật**

*Giảng viên:*

# MỤC TIÊU

- Cấu trúc Project
- A. Khởi tạo và cấu hình dự án Game 2D
- B. Tạo các đối tượng cơ bản
  - 1. *Game Object*
  - 2. *Sprite*
  - **3. *Animation và điều khiển hành động nhân vật***
  - 4. *Prefab*
  - 5. *Script và điều khiển máy trạng thái*
  - 6. *Thành phần vật lý và xử lý va chạm*
  - 7. *Sử dụng Text,*
  - 8. *Sử dụng Particle System*
  - 9. *Chuyển đổi màn chơi*
  - 10. *Sound*
  - 11. *Design Pattern trong Game*

# Nội dung

- Cấu trúc Project
- A. Khởi tạo và cấu hình dự án Game 2D
- B. Tạo các đối tượng cơ bản
  - 1. *Game Object*
  - 2. *Sprite*
  - **3. *Animation và điều khiển hành động nhân vật***
  - 4. *Prefab*
  - 5. *Script và điều khiển máy trạng thái*
  - 6. *Thành phần vật lý và xử lý va chạm*
  - 7. *Sử dụng Text,*
  - 8. *Sử dụng Particle System*
  - 9. *Chuyển đổi màn chơi*
  - 10. *Sound*
  - 11. *Design Pattern trong Game*

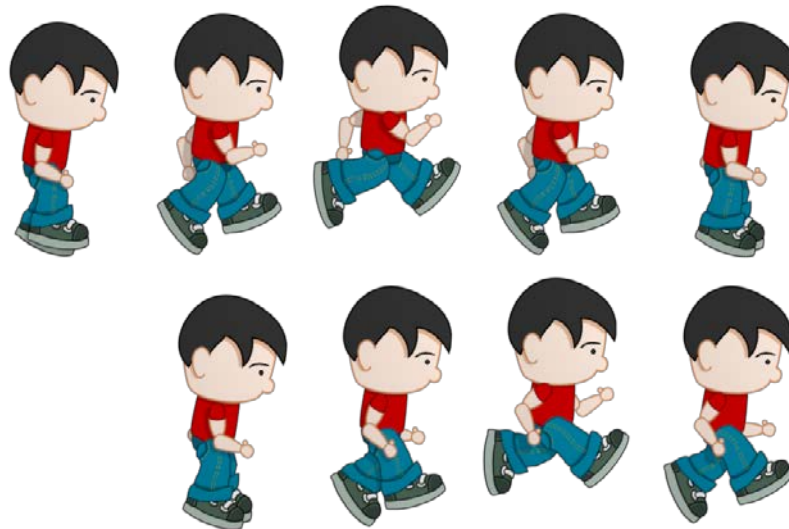
# Animation

- 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):
  - Kỹ thuật key frame.
  - Kỹ thuật skeletal hay spine.

# Animation

## Kỹ thuật tạo animation

- **Kỹ thuật key frame**
- Đối với kỹ thuật key frame, người ta sử dụng một sprite cho một key frame của hành động.



*Mỗi sprite là một keyframe*

# Animation

## Kỹ thuật tạo animation

- **Kỹ thuật key frame**
- Để tạo ra chuyển động, ta sẽ vẽ một key frame tại thời điểm đầu và thay đổi tuần tự các key frame sau, chúng ta sẽ có được một animation.
- Đây là phương pháp đơn giản nhất để tạo chuyển động, nhưng lại tốn kém về bộ nhớ, vì ta phải tốn nhiều sprite cho nhiều chuyển động khác nhau.

# Animation

## Kỹ thuật tạo animation

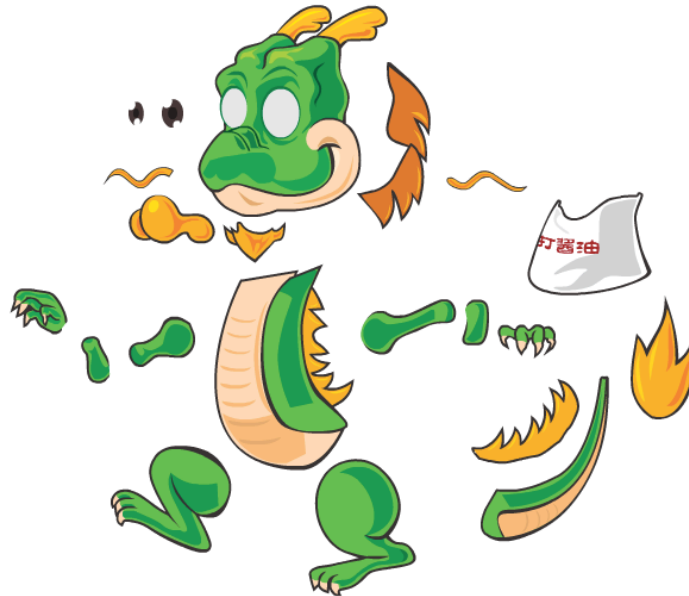
- \* **Kỹ thuật skeletal hay spine hay bộ xương**
- Đối với kỹ thuật này, người ta chia đối tượng ra thành nhiều sprite, mỗi sprite là một bộ phận của đối tượng (giống như 1 khúc xương của bộ xương).
- Để tạo ra một key frame mới, ta sẽ thay đổi các sprite về vị trí, độ lớn, xoay của các sprite thành 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.

# Animation

## Kỹ thuật tạo animation

- \* **Kỹ thuật skeletal hay spine hay bộ xương**
- Chúng ta có thể xem các sprite cấu tạo nên một key frame ở ảnh dưới.
- Cách này có vẻ tốn thời gian hơn, nhưng lại rất là hiệu quả, đặc biệt là tiết kiệm được nhiều bộ nhớ.

*Hình: 2D Spine*

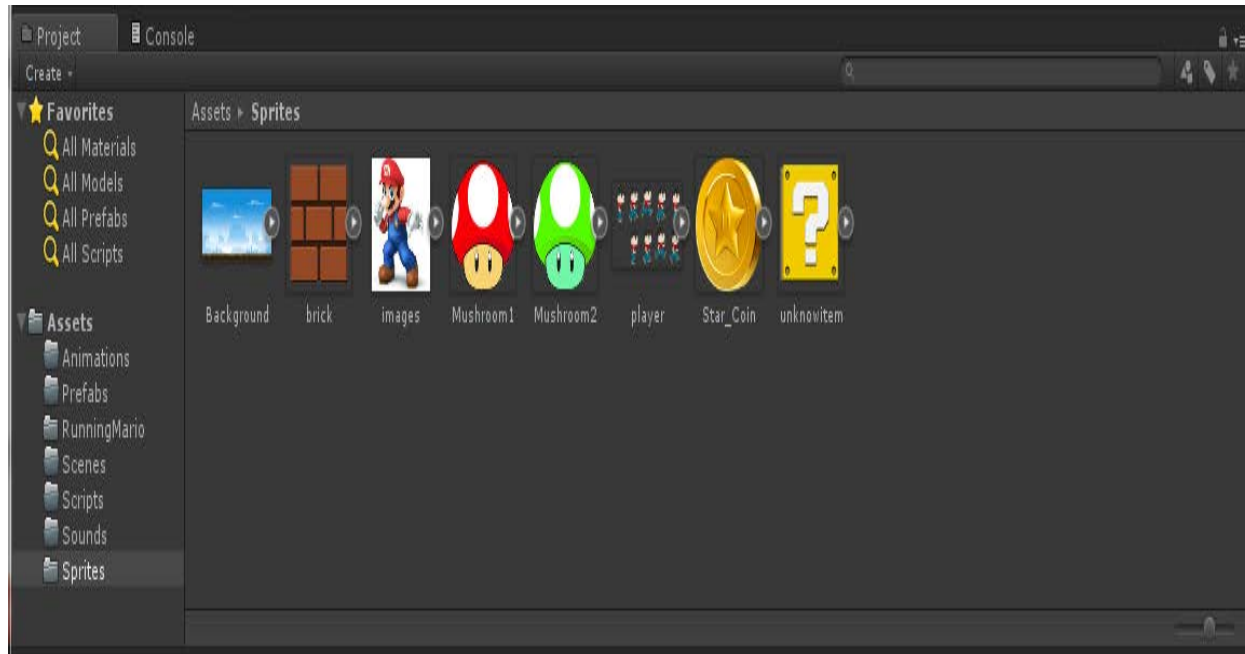




# Animation

## Tạo animation bằng Unity

- Tạo Animation theo kỹ thuật Key Frame.
- Cách tạo animation theo skeletal cũng tương tự.  
Từ bài trước chúng ta đã tạo được những Sprite cơ bản như sau:



# Animation

## Tạo animation bằng Unity


- **Bước 1:** Tạo một Empty GameObject đặt tên là MainCharacter (Parent Object)
- **Bước 2:** Tạo một đối tượng 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.





**DEMO**

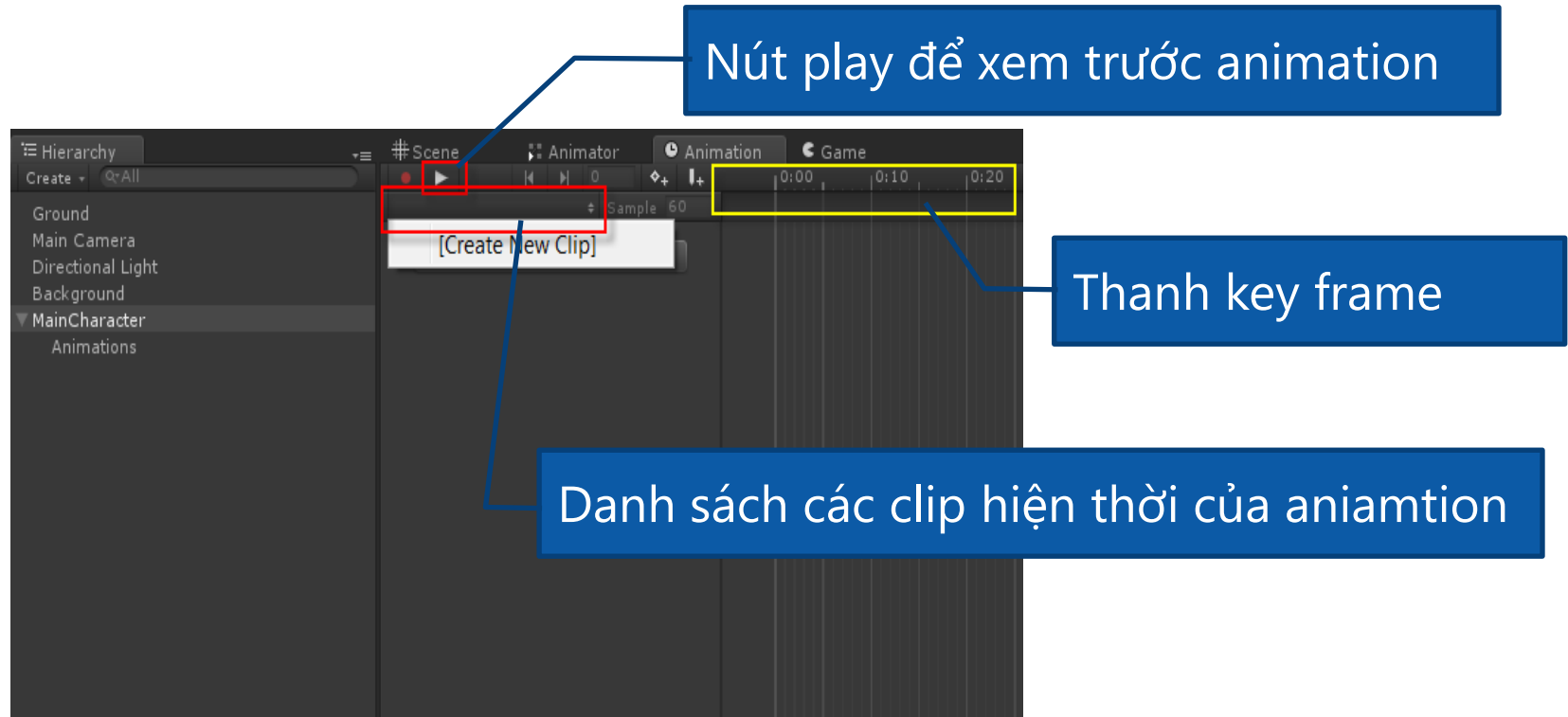
Bước 1, 2,3

A white hand cursor icon, resembling a computer mouse pointer, is positioned below the word 'DEMO'. The index finger is pointing upwards towards the letter 'O'.

# Animation

## Tạo animation bằng Unity

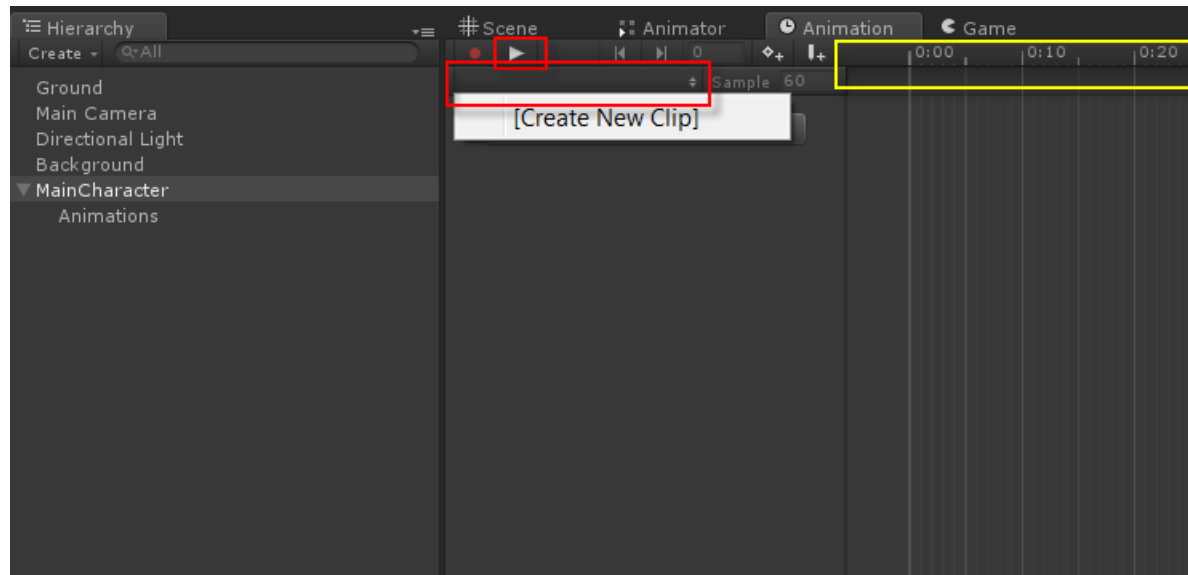
- **Bước 4:** Chọn đối tượng MainCharacter ở cửa sổ Hierarchy, rồi chọn Menu -> Window -> Animation.
- Một cửa sổ Animation editor hiện ra như sau:



# Animation

## Tạo animation bằng Unity


- Tạo một clip: Click vào danh sách clip rồi chọn Create New Clip, ta đặt tên clip là Running, rồi save lại ở thư mục Animations của Assets.





**DEMO**

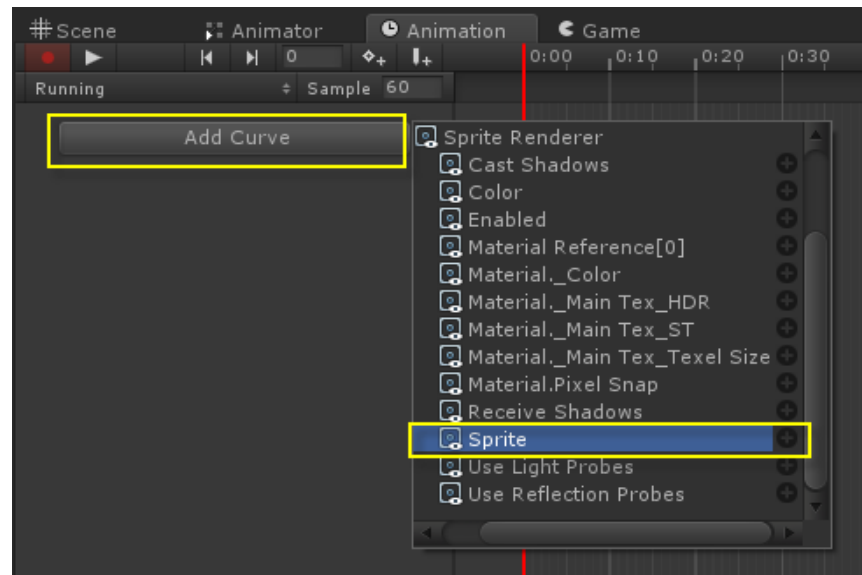
Bước 4

A large white hand cursor icon, resembling a computer mouse pointer, is positioned below the word 'DEMO'. The index finger of the hand is pointing upwards towards the letter 'O' in 'DEMO'.

# Animation

## Tạo animation bằng Unity

- **Bước 5:** Ở cửa sổ Animation Editor, chọn Add Curve, chọn Animations (Đối tượng con của đối tượng **MainCharacter**) chọn Sprite Render, chọn Sprite.
- Chú ý: Đối với một clip bất kỳ, bước này bắt buộc phải có.

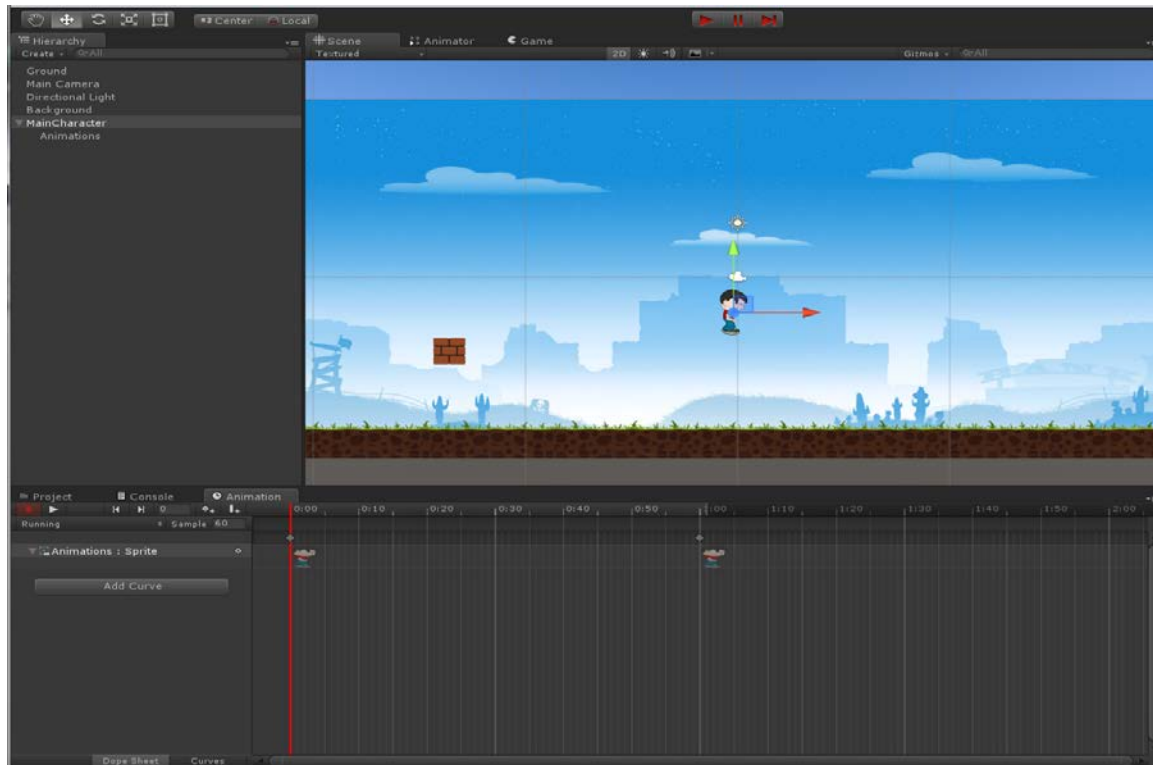


# Animation

## Tạo animation bằng Unity

### ■ Bước 5:

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

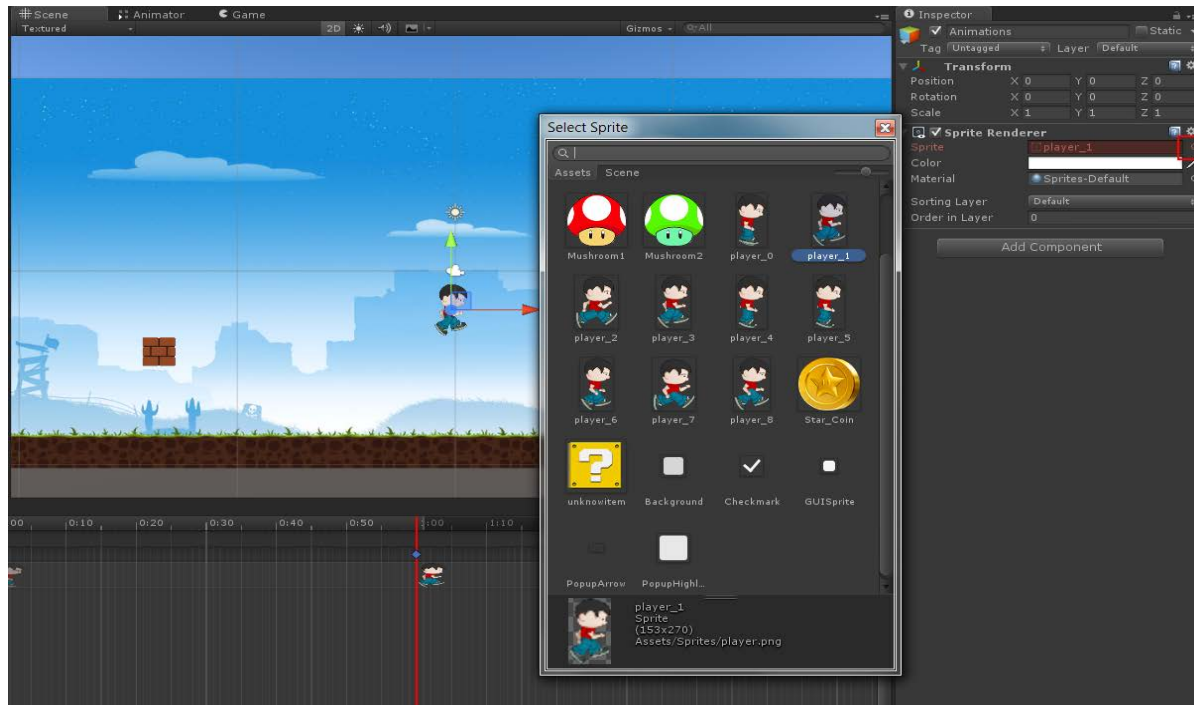




# Animation

## Tạo animation bằng Unity

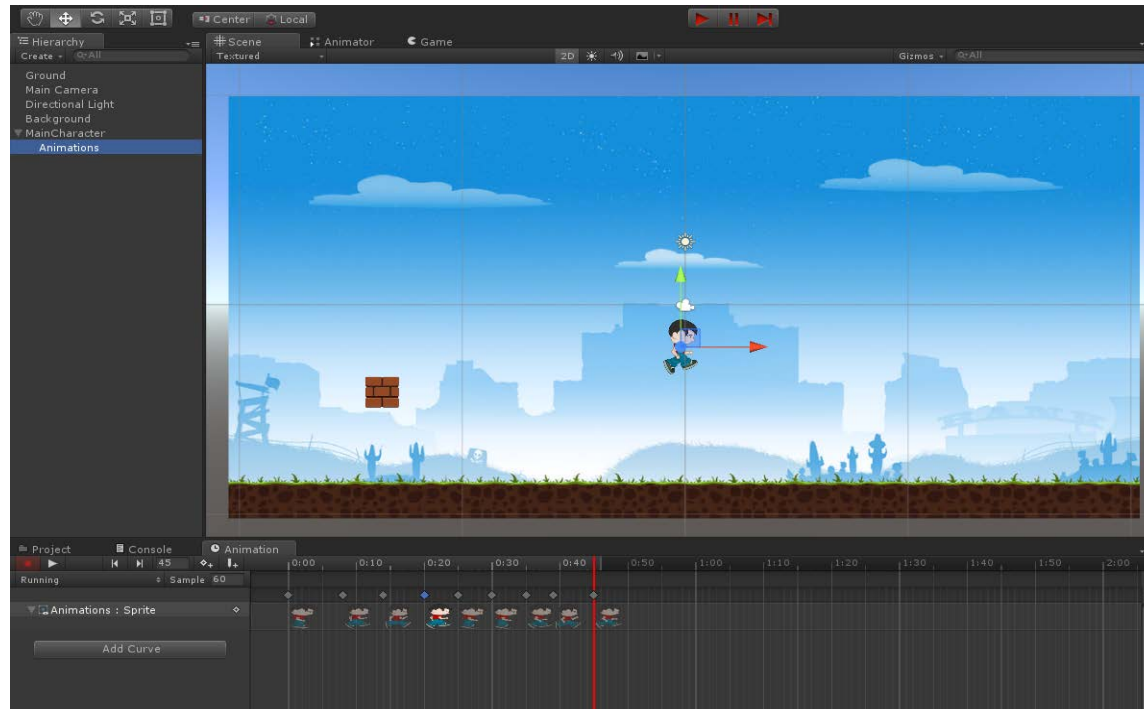
- Tiếp theo ta chọn key frame thứ 2, ở cửa sổ **Inspector**, ở component **Sprite Render**, ta tiến hành đổi sprite khác (player\_1 thay vì player\_0). Chọn nút được bao quanh bởi ô tròn đỏ, rồi chọn Sprite khác từ cửa sổ mới hiện ra.

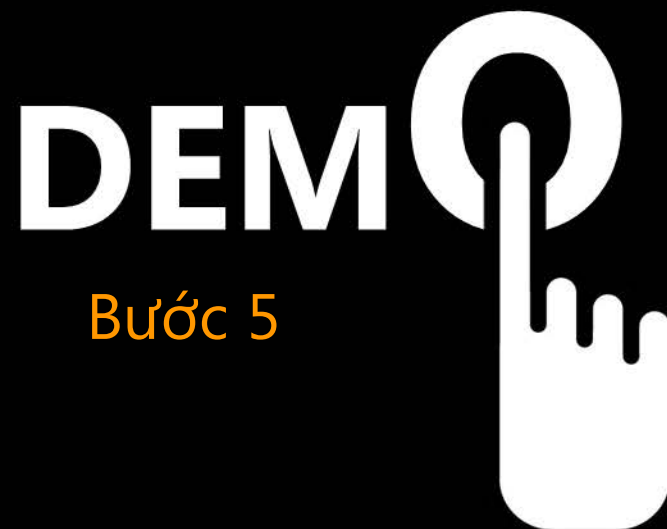


# Animation

## Tạo animation bằng Unity

- Bây giờ, chúng ta chỉ cần click đúp vào thanh Key Frame để thêm các key frame và kéo thả các key frame sao cho thời gian phù hợp để có được chuyển động cần thiết.

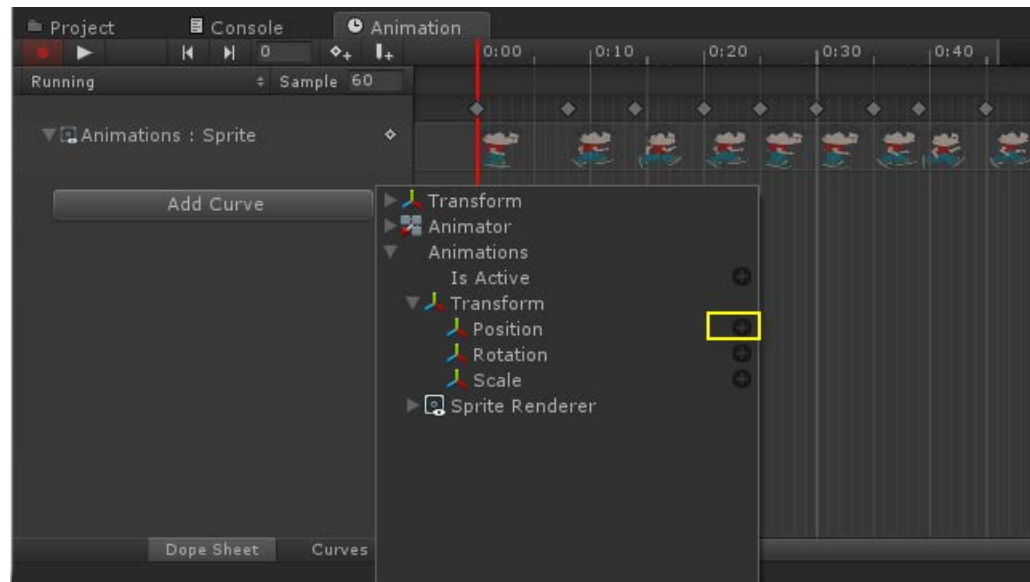




# Animation

## Tạo animation bằng Unity

- Ngoài thay đổi sprite chúng ta 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:



# Animation

## Tạo animation bằng Unity

- Sau đó ta chỉ việc chọn các key frame, rồi đặt các giá trị transform cho phù hợp theo ba trục x, y, z.
- Tương tự ta sẽ tạo các clip Jump (nhảy), Idle (trạng thái nghỉ) cho đối tượng.

# Animation

## Tạo animation bằng Unity

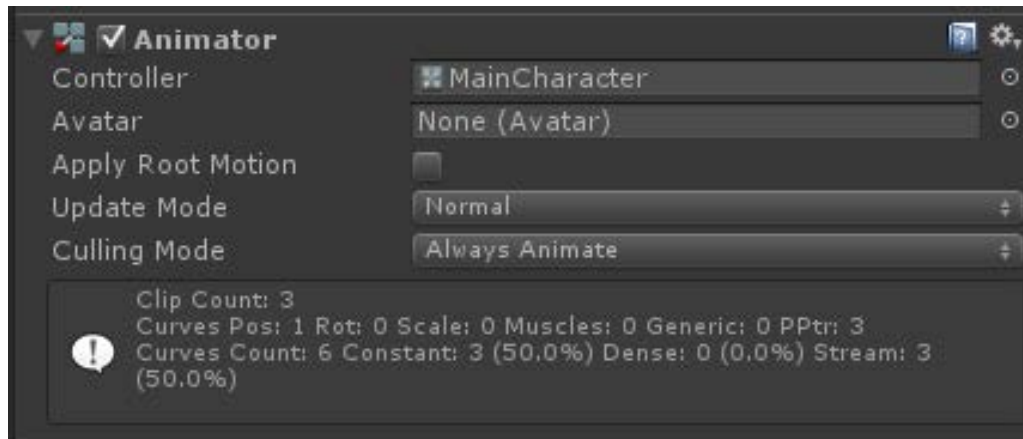
### ■ Chú ý:

- Thay đổi transform và sprite render của đối tượng con Animation chứ không thay đổi transform và sprite render của đối tượng cha là MainCharacter.
- Giải thích: Animation sẽ làm thay đổi transform của sprite, nên nếu thêm trực tiếp vào đối tượng cha, sau này thêm thành phần vật lý, hoặc áp dụng các phép transform vào sẽ bị sai, hoặc mất tác dụng.
- Đối với các hành động nhảy chúng ta sẽ không thay đổi vị trí các key frame, vì làm như vậy khi thêm thành phần vật lý vào đối tượng cha, lúc nhân vật nhảy, hình ảnh của nó nhảy lên nhưng vị trí của nó để tính vật lý (theo đối tượng cha) vẫn nằm ở dưới hoặc thấp hơn hình ảnh --> Không đúng thực tế.
- Đối với hành động nhảy này, ta chỉ cần chọn sprite đang ở tư thế nhảy mà thôi.



# Điều khiển hành động nhân vật (Animator)

- Để điều. khiển hiệu ứng(animation) trong unity chúng ta cần sử dụng Animator components
- Animator components có một số thuộc tính như sau(số lượng thuộc tính sẽ tùy thuộc vào đối tượng là 3D hay 2D):





# Điều khiển hành động nhân vật (Animator)

- Controller: chứa liên kết đến animator controller asset.
- Animator controller là assets tạo bởi Unity chứa một hoạt nhiều trạng thái( state machines) dùng để chỉ định animation nào đang được chạy khi sence đang chạy, những trạng thái (state machines) này có thể nằm trên nhiều layers và sử dụng nhiều kiểu biến khác nhau để điều khiển khi nào chúng dịch chuyển(transition) từ trạng thái này sang trạng thái kia.
- Ví dụ: có thể thay đổi thuộc tính của một animator controller từ script:

```
animator.SetTrigger("destroy");
```

```
//Biến ở đây là kiểu trigger
```

tương tự bạn có thể thay đổi trạng thái của gameObject animation sang các trạng thái khác nhau như đứng, chạy, nổ...

## Điều khiển hành động nhân vật (Animator)

- Avatar là một asset Unity tạo ra khi import một gameObject 3D hình người. Nó chứa định nghĩa dàn khung xương của nhân vật. Khi thêm đối tượng loại này vào scene của bạn, trường avatar sẽ tự động được cập nhật với một asset tạo ra cho nhân vật đó (nếu không sẽ là none).
- **Apply Root Motion:** là trường định nghĩa xem animation có ảnh hưởng đến Transform của gameObject hay không và thường được sử dụng với các nhân vật 3D hình người.

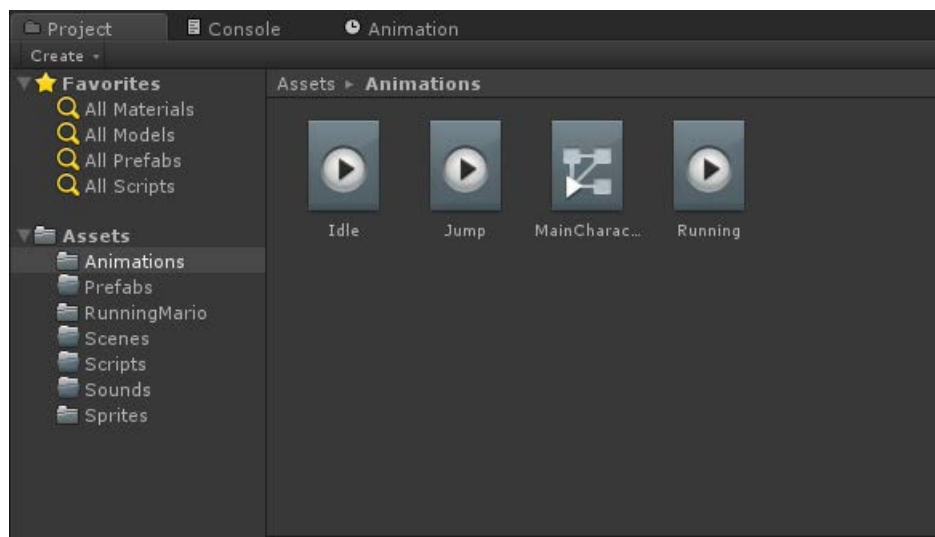
# Điều khiển hành động nhân vật (Animator)

- Animate Physics: Khi được check có nghĩa là animations sẽ được thực thi trong thời gian vật lý (thường thì thuộc tính này được dùng cho đối tượng có rigidbody).
- Culling Mode:
  - Base On Renderer chỉ định những animations chỉ được chạy khi chúng được render.
  - Always Animate có nghĩa là animations sẽ được chạy dù không render

# Điều khiển hành động nhân vật (Animator)

## Điều khiển các hành động nhân vật – Animator

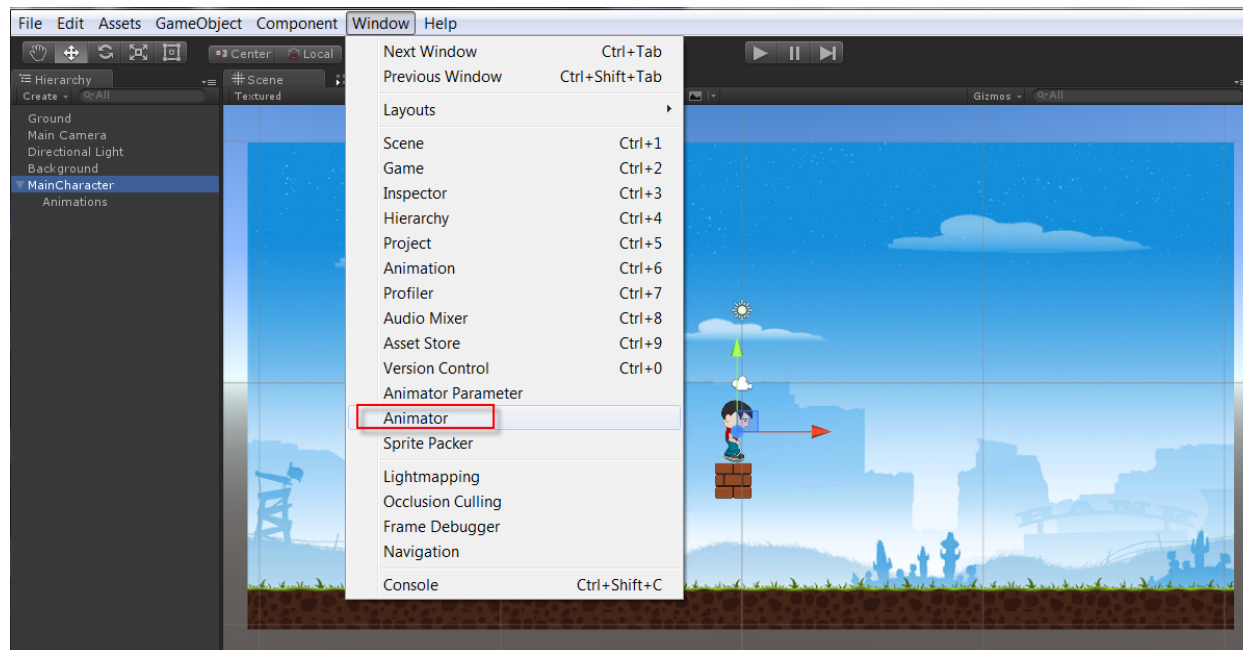
- Ở phần trước chúng ta đã tìm hiểu cách tạo các clip hay các hành động của một animation. Với một animation như vậy ta sẽ có một Controller (MainCharacter.controller) đi kèm theo.
- Phần này ta sẽ hướng dẫn cách chuyển qua lại giữa các hành động bằng máy trạng thái.



# Điều khiển hành động nhân vật (Animator)

## Điều khiển các hành động nhân vật – Animator

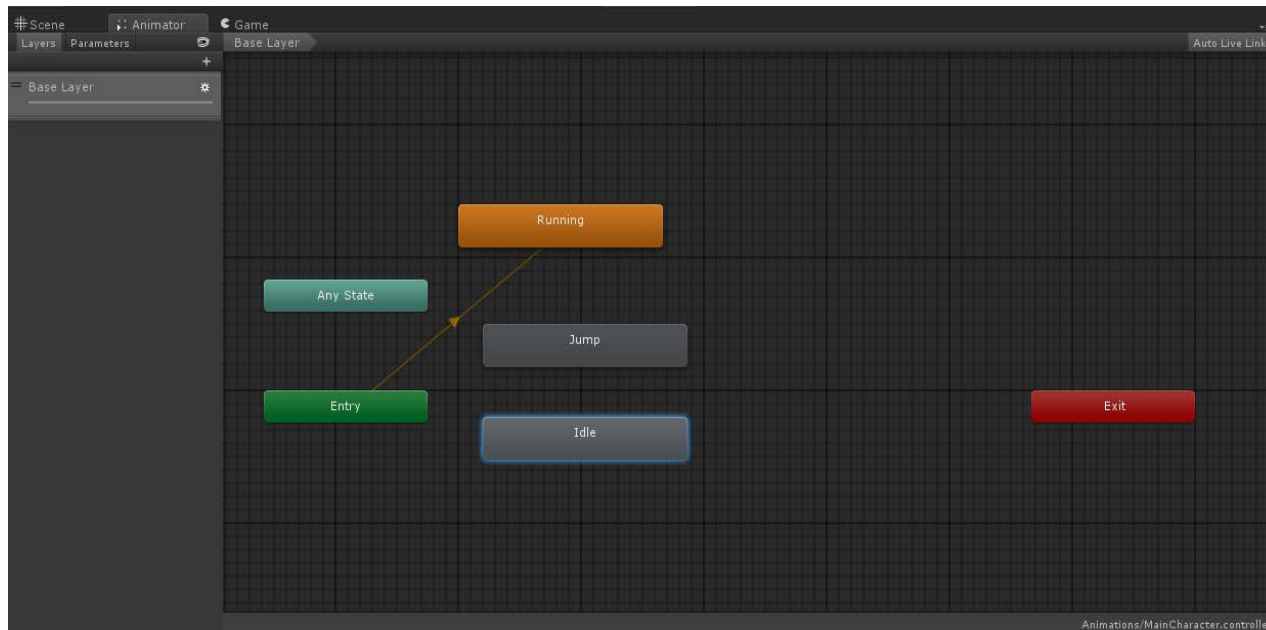
- Ở cửa sổ Hierarchy chọn đối tượng **MainCharacter**, chọn **Menu**, chọn **Window**, chọn **Animator**, cửa sổ Animator sẽ xuất hiện như sau:
- Hoặc có thể click đúp vào MainCharacter.controller.



# Điều khiển hành động nhân vật (Animator)

## Điều khiển các hành động nhân vật – Animator

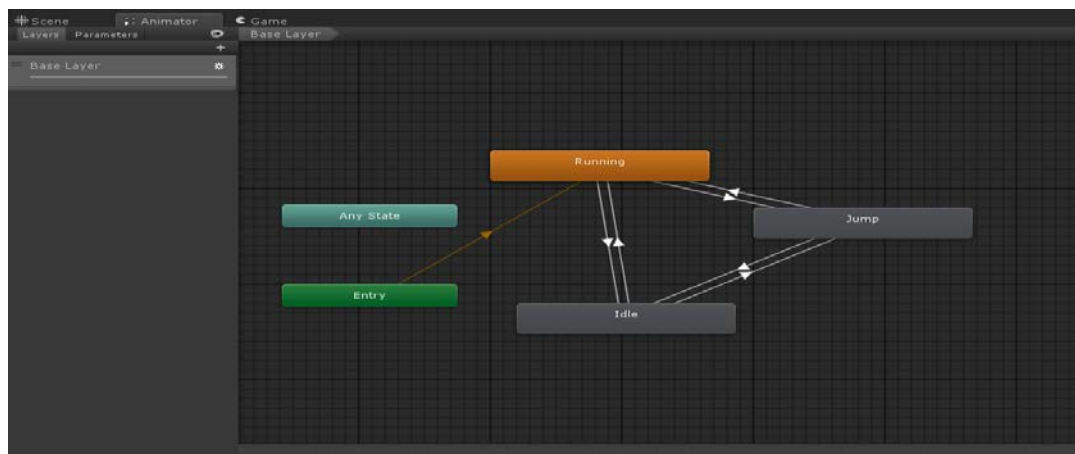
- Danh sách các state, mỗi state tương ứng với một clip



# Điều khiển hành động nhân vật (Animator)

## Điều khiển các hành động nhân vật – Animator

- Click chuột phải vào state Idle, chọn Set Default để thiết lập state mặc định cho đối tượng.
- 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 ta gọi là máy trạng thái hay sơ đồ chuyển đổi trạng thái.



# Điều khiển hành động nhân vật (Animator)

## Điều khiển các hành động nhân vật – Animator

- Khi đối tượng được load lên, trạng thái mặc định sẽ được thiết lập.
- Để chuyển qua trạng thái khác: ta tạo thêm các tham số, và dựa vào giá trị các tham số này để chuyển đổi các trạng thái.
- **Chú ý:** ở đây để cho đơn giản, ta thiết lập 3 trạng thái có thể chuyển qua lại trực tiếp với nhau.

3 tham số kiểu bool

- isJump,
- isIdle,
- isRunning



# Điều khiển hành động nhân vật (Animator)

## Điều khiển các hành động nhân vật – Animator

- Để thêm các tham số cho Animator controller bạn nhấp vào icon +
- Các tham số này có thể là kiểu float, ints, bool hoặc trigger.
- Các thông số này thường dùng trong việc thực hiện các animation hoặc chuyển đổi các animation.
- Ví dụ: giả sử bạn tạo một animation cho object di chuyển, có thể bạn sẽ sử dụng tham số kiểu float, và khi tham số float này đủ lớn, bạn có thể chuyển animation sang một animation khác thể hiện sự di chuyển nhanh hơn.



# DEMO

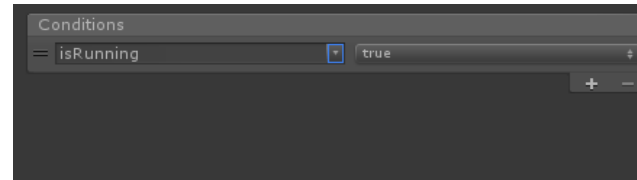
Tạo  
transition  
và state



# Điều khiển hành động nhân vật (Animator)

## Điều khiển các hành động nhân vật – Animator

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



# Điều khiển hành động nhân vật (Animator)

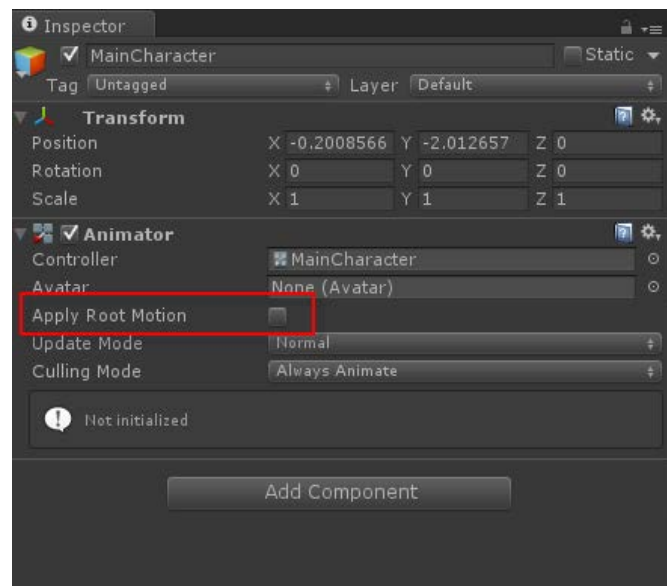
## Điều khiển các hành động nhân vật – Animator

- Ta 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 ta set giá trị tham số `isRunning = true`, trạng thái nhân vật sẽ chuyển sang `Running`.
- Tương tự ta thiết lập `isIdle`, `isJump` cho các transition khác.
- **Chú ý:** Phần Script ta sẽ đề cập rõ, cách thiết lập giá trị các tham số này.

# Điều khiển hành động nhân vật (Animator)

## Điều khiển các hành động nhân vật – Animator

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



- Đến đây, chúng ta đã biết cách xây dựng một animation cho các đối tượng trong game và điều khiển qua lại các đối tượng đó.



# DEMO

Thiết lập điều  
kiện cho  
transition



# Kết luận

- Cấu trúc Project
- A. Khởi tạo và cấu hình dự án Game 2D
- B. Tạo các đối tượng cơ bản
  - 1. *Game Object*
  - 2. *Sprite*
  - **3. *Animation và điều khiển hành động nhân vật***
  - 4. *Prefab*
  - 5. *Script và điều khiển máy trạng thái*
  - 6. *Thành phần vật lý và xử lý va chạm*
  - 7. *Sử dụng Text,*
  - 8. *Sử dụng Particle System*
  - 9. *Chuyển đổi màn chơi*
  - 10. *Sound*
  - 11. *Design Pattern trong Game*

# Chuẩn bị bài sau

- Cấu trúc Project
- A. Khởi tạo và cấu hình dự án Game 2D
- B. Tạo các đối tượng cơ bản
  - 1. *Game Object*
  - 2. *Sprite*
  - 3. *Animation và điều khiển hành động nhân vật*
  - **4. *Prefab***
  - **5. *Script và điều khiển máy trạng thái***
  - 6. *Thành phần vật lý và xử lý va chạm*
  - 7. *Sử dụng Text,*
  - 8. *Sử dụng Particle System*
  - 9. *Chuyển đổi màn chơi*
  - 10. *Sound*
  - 11. *Design Pattern trong Game*





**FPT POLYTECHNIC**

THANK YOU!

[www.poly.edu.vn](http://www.poly.edu.vn)