Lab 4 (4 tiết): Player and Camera

MỤC ĐÍCH

Bài lab này giúp sinh viên tìm hiểu và xây dựng một Player và vận hành một Camera di chuyển theo Player. Bài lab cũng giúp sinh viên thêm vào một Model3D cho Player.

YÊU CẦU

- Sinh viên đọc toàn bộ phần nội dung và thực hiện theo hướng dẫn.

- Thư mục đi kèm bài lab: Lesson 4

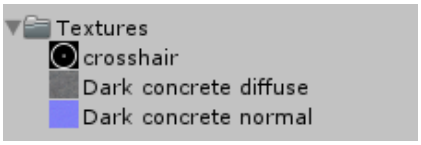
- Sau đó thực hiện các bài tập tương ứng.

NỘI DUNG

1. Giới thiệu

Trong bài lab này, sinh viên sẽ tìm hiểu phương pháp tạo ra một nhân vật (Player) với Capsule và gắn một Camera cho nhân vật đó.

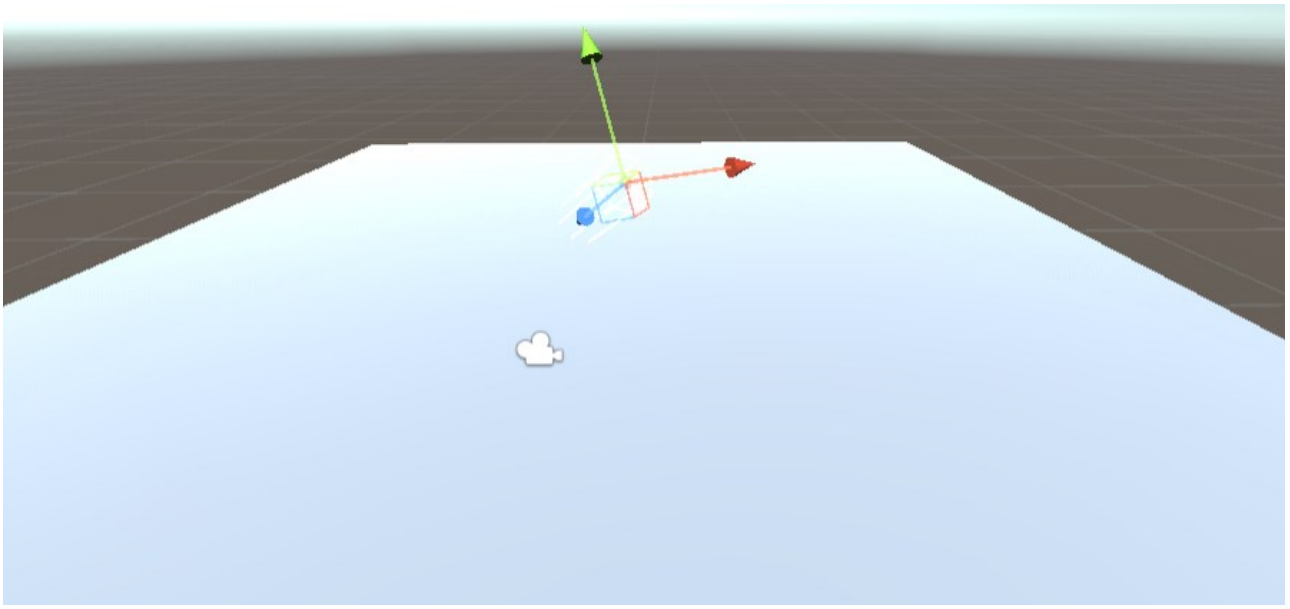
Tạo một dự án có tên là The Game, trong Project, tao một thư mục tên là Textures, đưa các tập tin có trong thư mục images đi kèm bài lab vào thư mục Textures vừa tạo.



2. Xây dựng mặt sàn

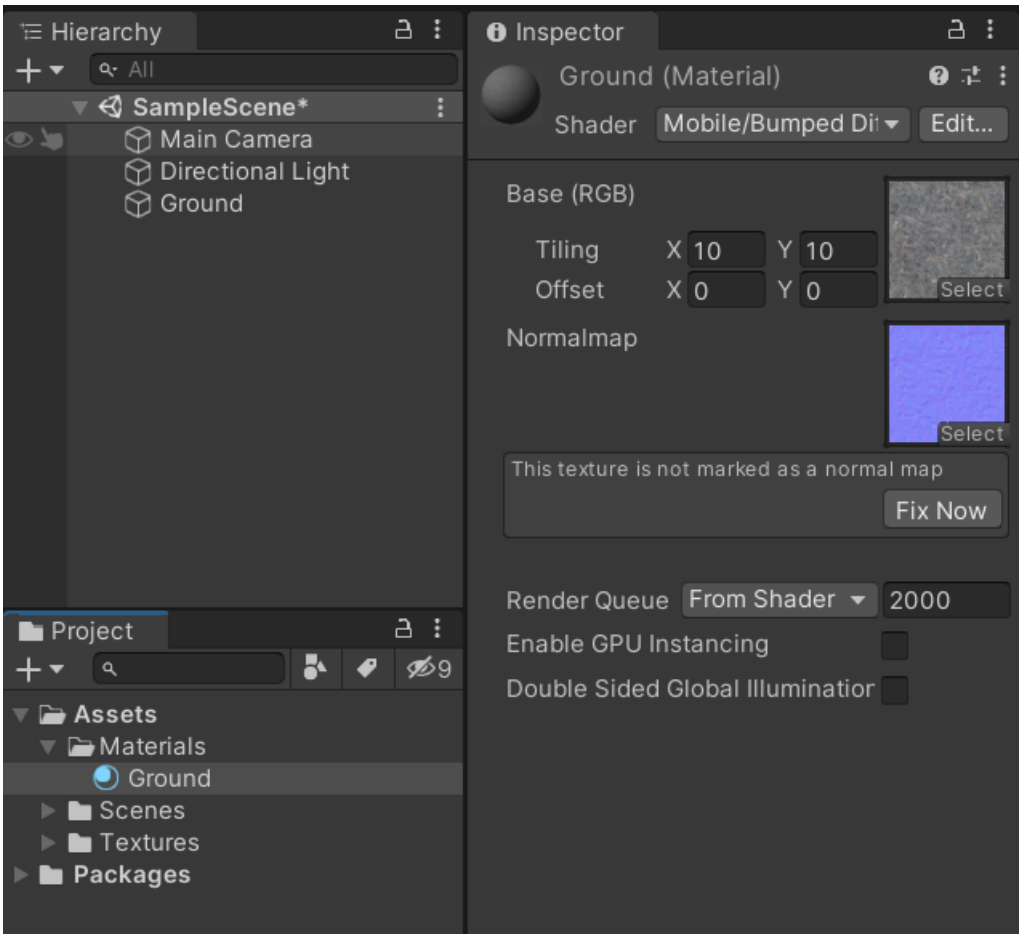
Tạo một khối hộp (Cube), đặt tên là Ground tại vị trí (0, 0, 0), thiết lập Scale là (50, 1, 50), bây giờ khối hộp trông như một mặt phẳng.

Điều chỉnh vị trí của Directional Light thành (30, -30, 0).



Trong Assets, tạo một thư mục Materials, trong thư mục đó tạo một Material và đặt tên là Ground.

Chọn material Ground vừa tạo, trên Inspector thiết lập thuộc tính Shader là Mobile/Bumped Diffuse, kéo 2 hình trong thư mục Textures (diffuse và nomal) vào 2 vị trí trên Inspector tương ứng, chỉnh tham số Tiling như hình dưới đây:

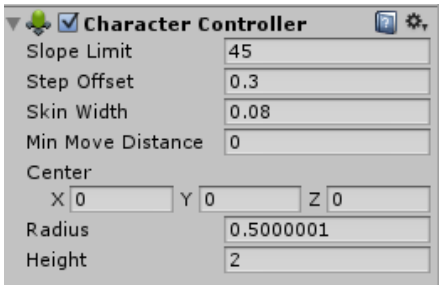


Gán Material Ground cho đối tượng Ground, ta được kết quả như sau:

Lưu ý: Sinh viên thay đổi các tham số Tiling X và Y để thấy sự ảnh hưởng của hình ảnh lên Ground.

Lưu màn vừa tạo vào thư mục Scenes với tên Main:

3. Character Controller



Trong Unity, để có thể điều khiển sự di chuyển của nhân vật, người ta sử dụng thành phần Character Controller. Thành phần này giống như một Collider, khi thêm vào đối tượng game, sẽ có các thuộc tính như:

- Slope Limit: độ dốc leo núi.

- Step Offset: độ dài bước đi của nhân vật.

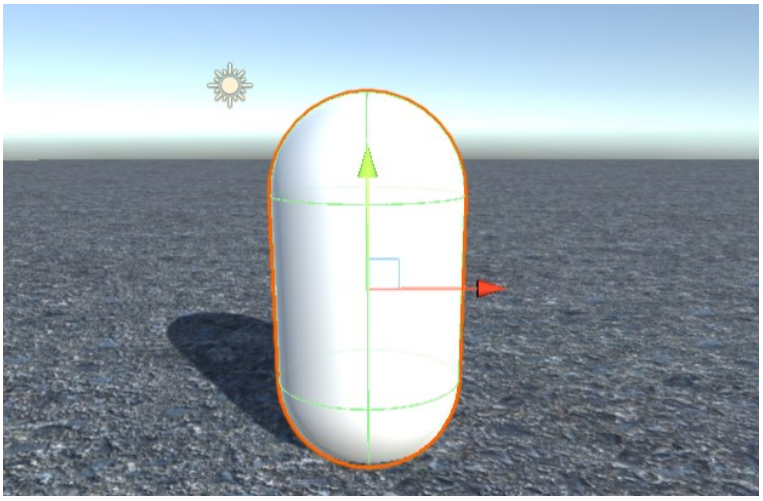
- Skin Width: độ dày của da, dùng trong xác định va chạm.

- Min move distance: Khoảng cách di chuyển tối thiểu.

- Center: tâm của Character Controller.

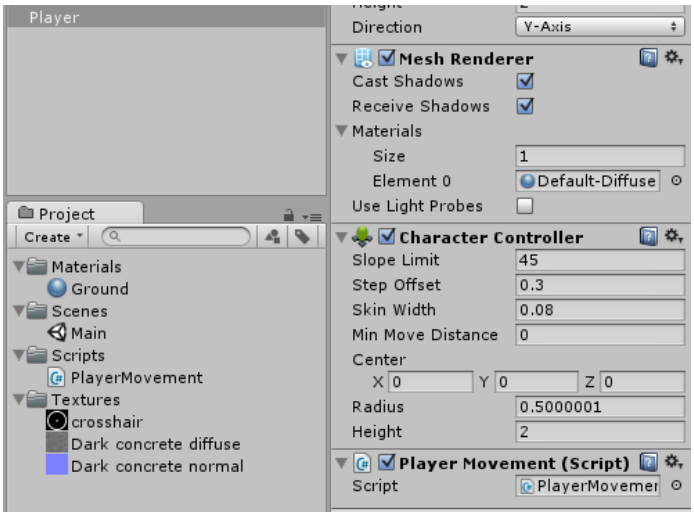
Ngoài ra còn một số thuộc tính cơ bản khác như bán kính, chiều cao, muốn điều khiển nhân vật, người ta thường dùng code để điều khiển.

Tạo một Capsule đặt tên là Player tại vị trí thích hợp và độ lớn thích hợp để camera có thể nhìn thấy. Thêm vào một thành phần Character Controller với các tham số mặc định.



4. Điều khiển nhân vật di chuyển

Tạo một thư mục có tên Scripts, trong thư mục này tạo tập tin *PlayerMovement.cs,* gán thành phần tập tin này cho đối tượng Player.



Thực hiện code cho nội dung tập tin như sau:

```
public class PlayerMovement : MonoBehaviour {

    // Khai báo đối tượng Character Controller
    CharacterController _characterController;

    // Tốc độ di chuyển của Player
    public float _moveSpeed = 5.0f;

    void Start () {
        // Lấy Character Controller từ Player
        _characterController = GetComponent<CharacterController>();
    }

    void Update () {
      // Khai báo một Vector hướng, lấy giá trị khi người dùng nhấn phím mũi tên
      Vector3 direction = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
        Vector3 velocity = direction * _moveSpeed * Time.deltaTime; // Nhân với tốc độ và deltaTime
        _characterController.Move(velocity); // nhân vật di chuyển
    }
}
```

Nhấn phím mũi tên trái, phải và lên xuống để xem kết quả.

5. Giúp nhân vật nhảy lên

Để giúp nhân vật nhảy lên khi nhấn phím SpaceBar, thực hiện thay đổi code như sau bởi các biến giá trị và trong hàm Update():

- Khai báo thêm các biến _jumpSpeed, _gravity, _yVelocity:

```
// Khai báo đối tượng Character Controller
CharacterController _characterController;
// Tốc độ di chuyển của Player
public float _moveSpeed = 5.0f;
// Độ cao của bước nhảy.
public float _jumpSpeed = 20.0f;
// Giá trị trọng lực
 float _gravity = 1.0f;
// giá trị theo chiều trục y
 float _yVelocity = 0.0f;

 void Start()
 {
     // Lấy Character Controller từ Player
     _characterController = GetComponent<CharacterController>();
 }
```

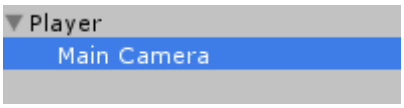
- Trong phương thức Update, thực hiện thay đổi code như sau (lưu ý chuyển Time.deltaTime vào trong hàm Move).

```
void Update () {
  // Khai báo một Vector hướng, lấy giá trị khi người dùng nhấn phím mũi tên
  Vector3 direction = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
  Vector3 velocity = direction * _moveSpeed; // Nhân với tốc độ và deltaTime

     if (_characterController.isGrounded) // Nếu đang ở mặt đất
     {
         if (Input.GetButtonDown("Jump"))  // mà người dùng nhấn phím SpaceBar
         {
             _yVelocity = _jumpSpeed;    // Thì gán giá trị trục y lên độ cao jumpSpeed;
         }
     }
     else _yVelocity -= _gravity; // Nếu không ở mặt đất thì giảm độ cao dần dần (rơi).
     velocity.y = _yVelocity;  // gán giá trị y của vector vận tốc.
     _characterController.Move(velocity*Time.deltaTime); // nhân vật di chuyển
}
```

Chạy và xem kết quả, nhấn phím mũi tên qua trái, phải, và nhấn phím Spacebar để xem nhân vật nhảy lên.

6. Điều chỉnh Camera

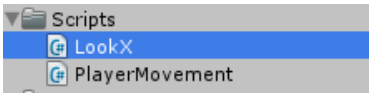
Để giúp camera luôn đi sau Player, thực hiện đưa đối tượng Camera thành con của đối tượng Player, đặt Camera tại vị trí (0, 0, -7) và đảm bảo góc quay của camera là (0, 0, 0).



Lưu lại, nhấn nút Play và quan sát sẽ thấy khi ta điều khiển Player di chuyển thì camera sẽ đi theo Player.

7. Quan sát bên trái, bên phải

Trong thư mục Scripts, tạo tập tin tên là LookX.cs



Thực hiện thêm code như sau trong class LookX.

```
// Biến cho biết sự thay đổi của trỏ chuột
public float _mouseX = 0.0f;

// Update is called once per frame
void Update () {
    _mouseX = Input.GetAxis("Mouse X"); // Tìm giá trị
}
```

Gắn script vào Player và chạy game, khi đó mỗi lần đưa chuột qua lại thì giá trị biến MouseX có sự thay đổi. Khi rê chuột qua trái thì giá trị âm, qua phải thì giá trị dương.

Bây giờ thay đổi code thành như sau trong phương thức Update():

```
// Update is called once per frame
void Update () {
    _mouseX = Input.GetAxis("Mouse X"); // Tìm giá trị
    // Khai báo giá trị quay theo góc Euler
     Vector3 rot = transform.localEulerAngles;
     //gán giá trị góc quay y theo chuột
     rot.y += _mouseX;
     transform.localEulerAngles = rot; // gán lại góc quay đó
}
```

Chạy game, rê chuột qua trái, phải để thấy khi đối tượng quay thì camera sẽ quay theo.

Lưu ý: có thể gia tăng độ nhạy bén của chuột bằng cách thêm vào một biến tùy chọn Sensitive như trong phần code sau:

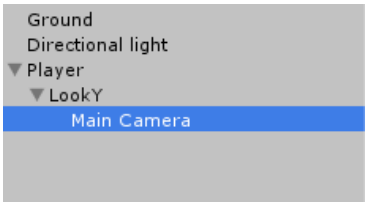
```
// Biến cho biết sự thay đổi của trỏ chuột
public float _mouseX = 0.0f;
// Độ nhạy
public float _sensitivity = 5.0f;
// Update is called once per frame
void Update () {
    _mouseX = Input.GetAxis("Mouse X"); // Tìm giá trị
    // Khai báo giá trị quay theo góc Euler
     Vector3 rot = transform.localEulerAngles;
     //gán giá trị góc quay y theo chuột
     rot.y += _mouseX*_sensitivity; // nhân với độ nhạy
     transform.localEulerAngles = rot; // gán lại góc quay đó
}
```

8. Quan sát trên, dưới.

Trong thư mục Scripts, tạo một tập tin có tên là LookY.cs, tiến hành viết code như sau:

```
// Biến cho biết sự thay đổi của trỏ chuột
public float _mouseY = 0.0f;
// Độ nhạy
public float _sensitivity = 5.0f;
// Update is called once per frame
void Update()
{
    _mouseY = -Input.GetAxis("Mouse Y"); // Tìm giá trị
    // Khai báo giá trị quay theo góc Euler
    Vector3 rot = transform.localEulerAngles;
    //gán giá trị góc quay y theo chuột
    rot.x += _mouseY * _sensitivity; // nhân với độ nhạy
    transform.localEulerAngles = rot; // gán lại góc quay đó
}
```

Tạo một GameObject mới, đặt tên là LookY và đưa vào thành con của Player, thiết lập vị trí của LookY về (0, 0, 0), phép quay là (0, 0, 0) và phép co dãn là (1, 1, 1), sau đó đưa Main Camera vào thành con của LookY.



Gán tập tin LookY.cs vào đối tượng LookY, chạy game và xem kết quả.

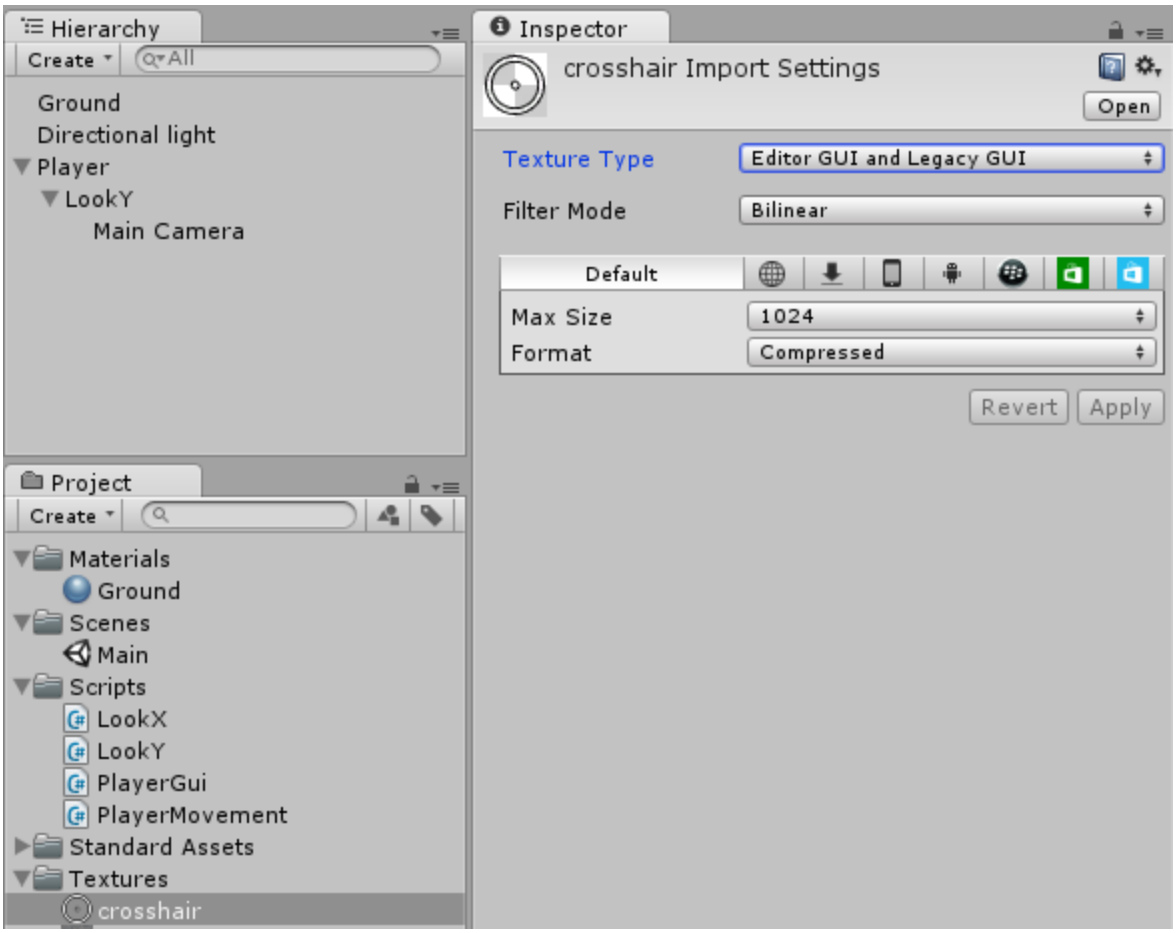
9. Tạo điểm ngắm của đối tượng.

Phần này chúng ta sẽ tạo ra một điểm ngắm bằng cách dùng một hình ảnh và vẽ nó lên giữa trung tâm của màn hình.

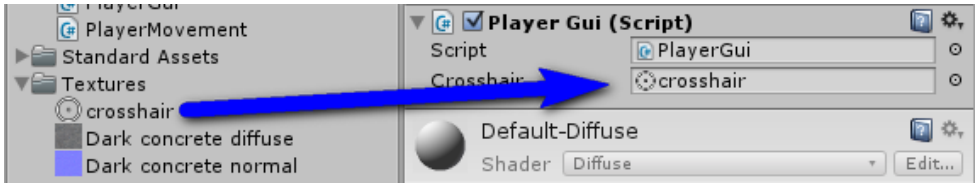
Tạo một lớp C#, đặt tên là "PlayerGui", viết nội dung như sau:

```
public class PlayerGui : MonoBehaviour {

    // Tạo một texture2D
    public Texture2D _crosshair;

    // Sử dụng hàm OnGui để vẽ lên màn hình khi vào game
    void OnGUI()
     {
        // Vẽ teture tại một vị trí trên màn hình
         GUI.DrawTexture(new Rect(0, 0, 30, 30), _crosshair);
     }
}
```

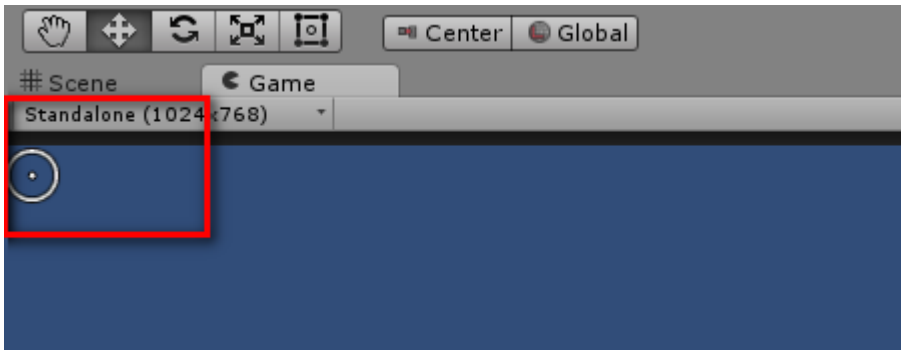
Trong thư mục Textures, click chọn tập tin crosshair, trên Inspector chọn thuộc tính TextureType là **Editor GUI and Legacy GUI**, nhấn Apply để chuyển texture từ nền đen qua nền trong suốt.



Gán script PlayerGui cho Player, kéo texture crosshair vào thuộc tính Crosshair.



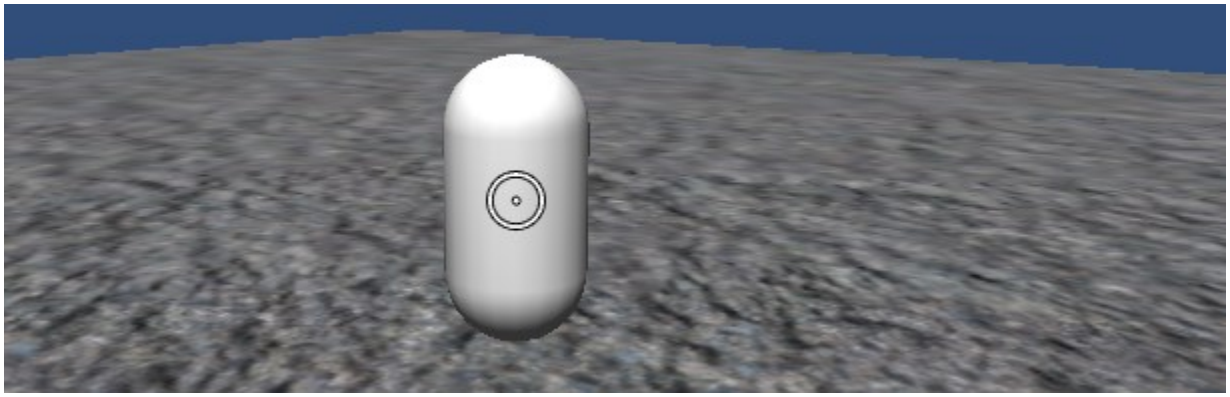
Chạy game và xem kết quả là hình vẽ được vẽ tại góc của màn hình:



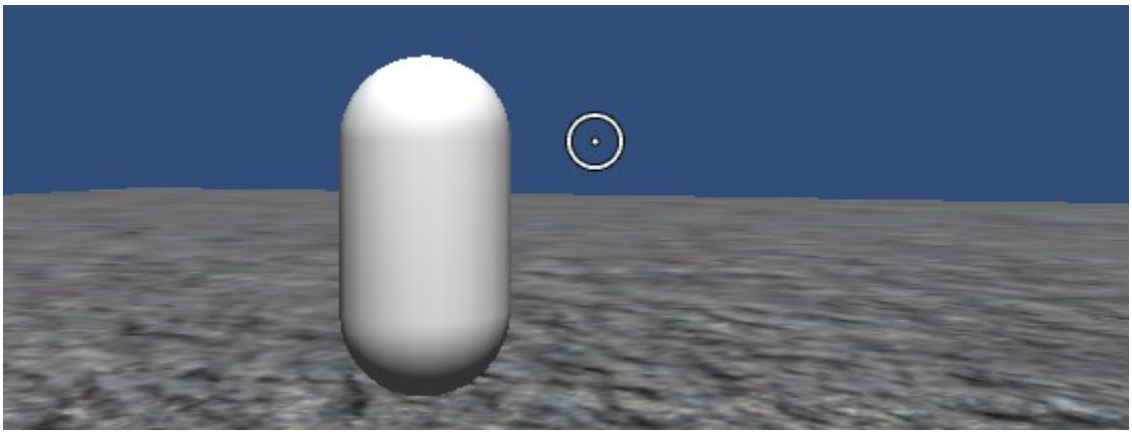
Để chuyển texture ra giữa màn hình, khai báo các tham số x, y như sau:

```
// Sử dụng hàm OnGui để vẽ lên màn hình khi vào game
void OnGUI()
{
    // Vị trí x, y nằm giữa màn hình
    float x = (Screen.width - _crosshair.width) / 2;
    float y = (Screen.height - _crosshair.height) / 2;
    // Vẽ teture tại một vị trí x, ytrên màn hình
    GUI.DrawTexture(new Rect(x, y, _crosshair.width, _crosshair.height), _crosshair);
}
```

Chạy và xem kết quả.



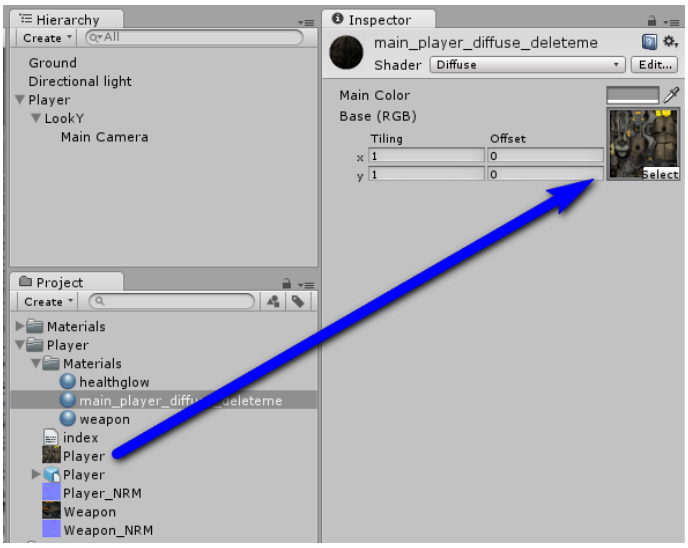
Để giúp điểm ngắm trông giống thực tế, ta chỉnh vị trí của đối tượng LookY về (1, 0.5, 1), kết quả như hình sau:



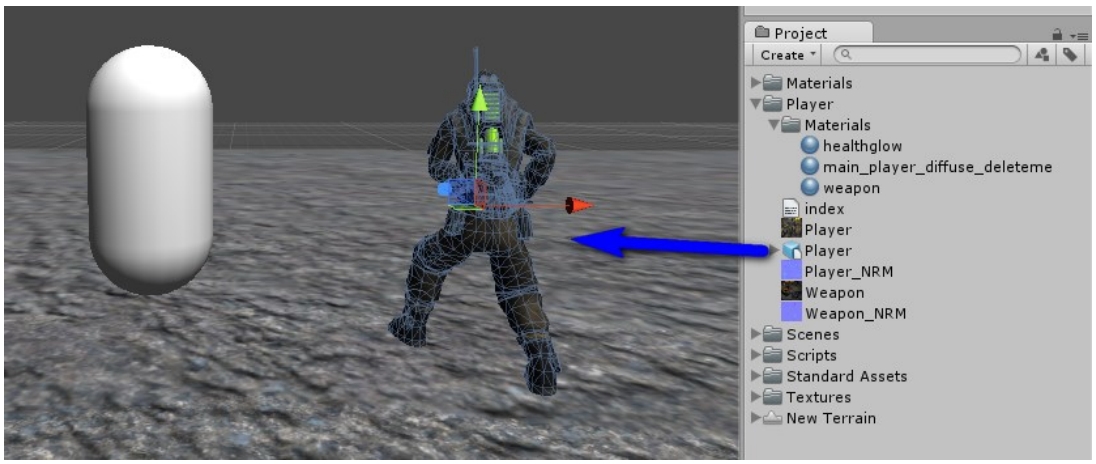
10. Gắn hình vào nhân vật

Trong thư mục đính kèm, kéo toàn bộ thư mục Player vào dự án.

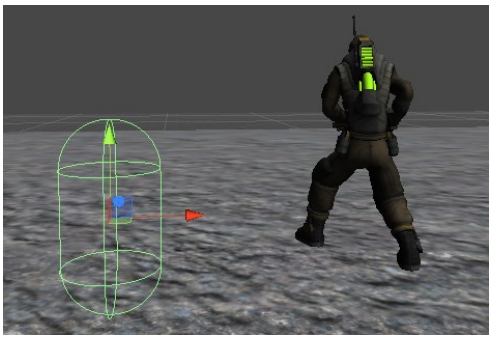
Tìm đến thư mục Materials, gán Texture Player vào Material thứ hai như hình sau:



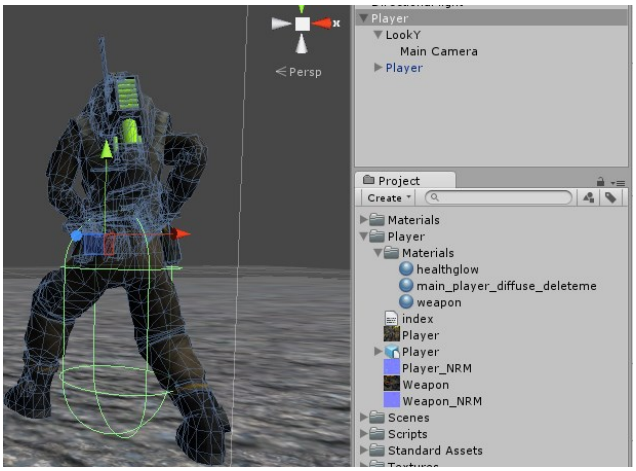
Thực hiện hành động tương tự cho Weapon. Kéo đối tượng Player vào Scene:



Thiết lập vị trí của đối tượng Player (Capsule) sao cho đối tượng nằm trên mặt đất, xóa thành phần Capsule (Mesh Fillter), bỏ chọn thành phần Mesh Renderer.



Trên Hierachy, kéo đối tượng Player (3D model) vào thành con của đối tượng Player (Capsule), thiết lập vị trí Player (3D) về (0, -1, 0) sao cho nhân vật đứng trên mặt đất.



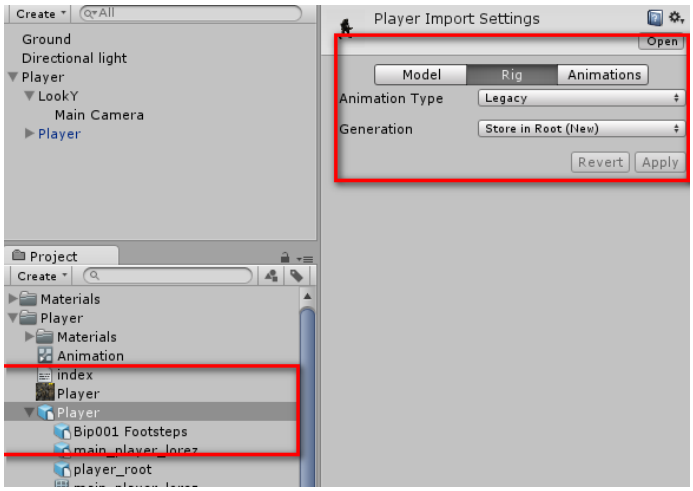
Lưu ý:

- Có thể thay đổi độ cao của Character Controller để Capsule bao hết nhân vật.

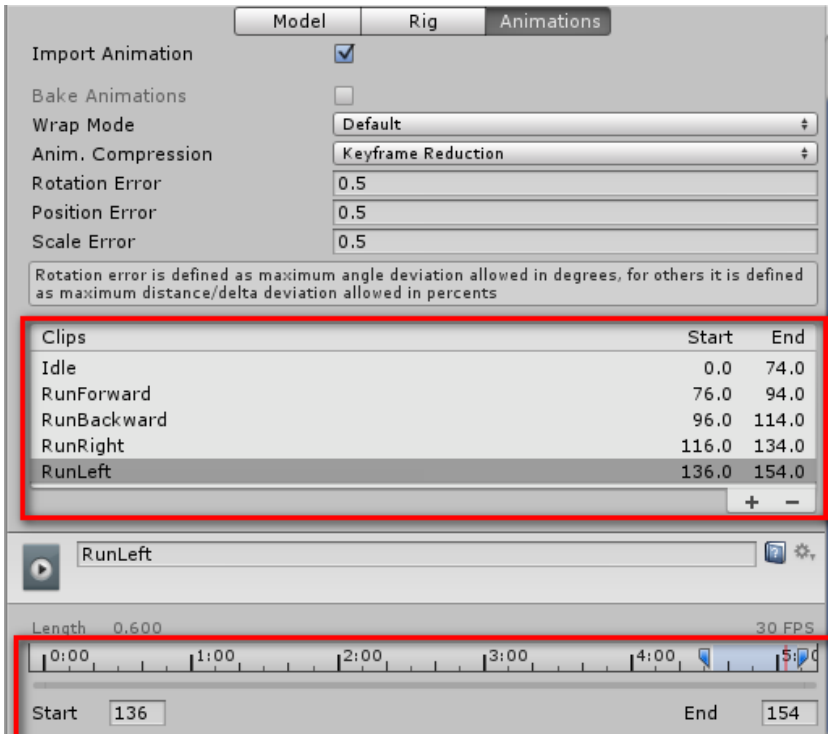
- Chỉnh ống ngắm (crosshair) lên cao bằng cách chọn đối tượng LookY và thay đổi vị trí thành (1, 1.5, 1.5).

11. Tạo hoạt hình cho nhân vật:

Chọn Player (3D) trong Project, trên Inspector, chọn thẻ Rig và thay đổi thuộc tính Animation Type thành Legacy, nhấn Apply.

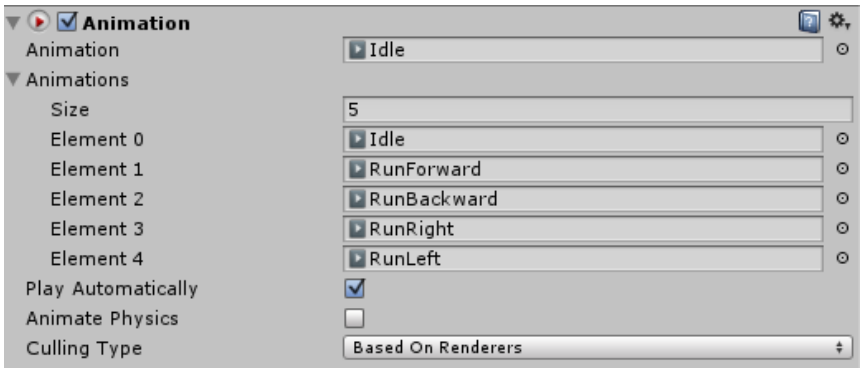


- Trong thẻ Animations, thiết lập các trạng thái của nhân vật bằng cách click vào nút **+** và thiết lập các frame bắt đầu (start) và kết thúc (end) như hình sau:



Lưu ý: Khi thiết kế nhân vật, các họa sĩ đã thiết kế sẵn bộ Animations cho trong nhân vật, ở đây chỉ hướng dẫn cách lấy các animation đó bằng cách xác định frame bắt đầu và kết thúc.

Thực hiện xong thì nhấn Apply. Khi nhấn Apply, chọn đối tượng Player (3D) trên Hierachy, thấy thuộc tính Animation tự động thêm vào, trong đó các thành phần được thêm vào như sau:



Tạo script PlayerAnimation.cs và gắn vào Player (3D), code như sau:

```
public class PlayerAnimation : MonoBehaviour {

    Animation _animation; // Biến Animation
    void Start () {
        // Lấy Animation từ Player
        _animation = GetComponent<Animation>();
    }

    void Update()
    {
        float v = Input.GetAxis("Vertical"); // Phím mũi tên lên, xuống
        if (v > 0) // Nhấn lên
        {
            _animation.Play("RunForward");
        }
        else if(v < 0) // Nhấn xuống
        {
            _animation.Play("RunBackward");
        }

        float h = Input.GetAxis("Horizontal"); // Phím mũi tên trái, phải

        if (h > 0) // Nhấn phải
        {
            _animation.Play("RunRight");
        }
        else if(h < 0) // Nhấn trái
        {
            _animation.Play("RunLeft");
        }
        if (v ==0 && h == 0) _animation.Play("Idle"); // Không phím nào nhấn
    }
}
```

Chạy và xem kết quả.

BÀI TẬP

Bài 1.

Thay đổi đối tượng Ground bằng cách dùng Terrain, thiết lập cây, đồi núi, đá nước cho Terrain để Player lúc này ở trong Terrain.

Thiết lập khung trời (sky box) cho game.

Bài 2.

Thêm vào viên đạn sao cho mỗi lần nhấn Enter thì có đạn bay ra khỏi nòng súng.

Bài 3.

Thiết lập điểm ngắm bắn tùy chọn bằng cách cho người dùng một số phím đặc biệt (Ví dụ: U,J,H,K) để dịch chuyển điểm ngắm lên(U), xuống(J), trái(H), phải(K).

Bài 4.

Xử lý trong lớp PlayerAnimation để khi nhấn cặp phím (lên - trái hoặc lên - phải …) thì chỉ thực hiện một Animation (*tham khảo phụ lục*).

Bài 5.

Thiết lập 2 phím G, X để khi người dùng nhấn phím G (gần) thì Camera sẽ là mắt người chơi, điểm ngắm phù hợp với mắt người chơi, khi nhấn phím X (xa) thì sẽ đưa về trạng thái mặc định (trạng thái camera như trong hướng dẫn).

Bài 6.

Sinh viên vào Asset Store, tìm hiểu một Model về thú (animal asset), thực hiện làm chuyển động con thú đó bằng code như trong mục 10 & 11 (phần hướng dẫn).

Bài 7.

Thực hiện xây dựng 1 game đơn giản như video sau đây (lưu ý, video này chỉ là phần 1, sinh viên thực hiện xong thì làm các phần tiếp theo để có game hoàn chỉnh):

https://www.youtube.com/watch?v=G9BdFZ2MCXc

PHỤ LỤC

Code tham khảo cho việc di chuyển Player khi nhấn cặp phím:

```
void Update()
    {
        float v = Input.GetAxisRaw("Vertical");
        float h = Input.GetAxisRaw("Horizontal");

        bool runningForward = v > 0;
        bool runningBackward = v < 0;
        bool notRunningForwardOrBackward = v == 0;
        bool runningForwardOrStanding = v >= 0;

        bool runningLeft = h < 0;
        bool runningRight = h > 0;
        bool notRunningLeftOrRight = h == 0;

        bool notRunningAtAll = notRunningForwardOrBackward &&
notRunningLeftOrRight;

        if (runningForward)
        {
            _animation.Blend("RunForward");
            _animation.Blend("Idle", 0);
            _animation.Blend("RunBackward", 0);
        }
        else if (runningBackward)
        {
            _animation.Blend("RunBackward");
            _animation.Blend("Idle", 0);
            _animation.Blend("RunForward", 0);
        }
        else if (notRunningForwardOrBackward)
        {
            _animation.Blend("RunForward", 0);
            _animation.Blend("RunBackward", 0);
        }

        if ((runningForwardOrStanding && runningRight) || (runningBackward &&
runningLeft))
        {
            _animation.Blend("RunRight");
            _animation.Blend("Idle", 0);
            _animation.Blend("RunLeft", 0);
        }
        else if ((runningForwardOrStanding && runningLeft) || (runningBackward
&& runningRight))
        {
            _animation.Blend("RunLeft");
```

```
            _animation.Blend("Idle", 0);
            _animation.Blend("RunRight", 0);
        }
        else if (notRunningLeftOrRight)
        {
            _animation.Blend("RunLeft", 0);
            _animation.Blend("RunRight", 0);
        }

        if (notRunningAtAll)
        {
            _animation.CrossFade("Idle");
        }
}
```