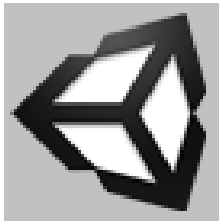
LẬP TRÌNH GAME VỚI UNITY

Bài 5: Interactions

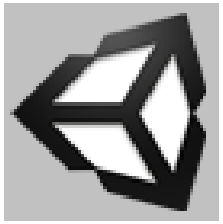
ThS. Thái Duy Quý

Đà Lạt, Tháng 03 năm 2016



Nội dung

- Rigidbody
- Collider
- Collision
- Example
- Raycasting
- Example

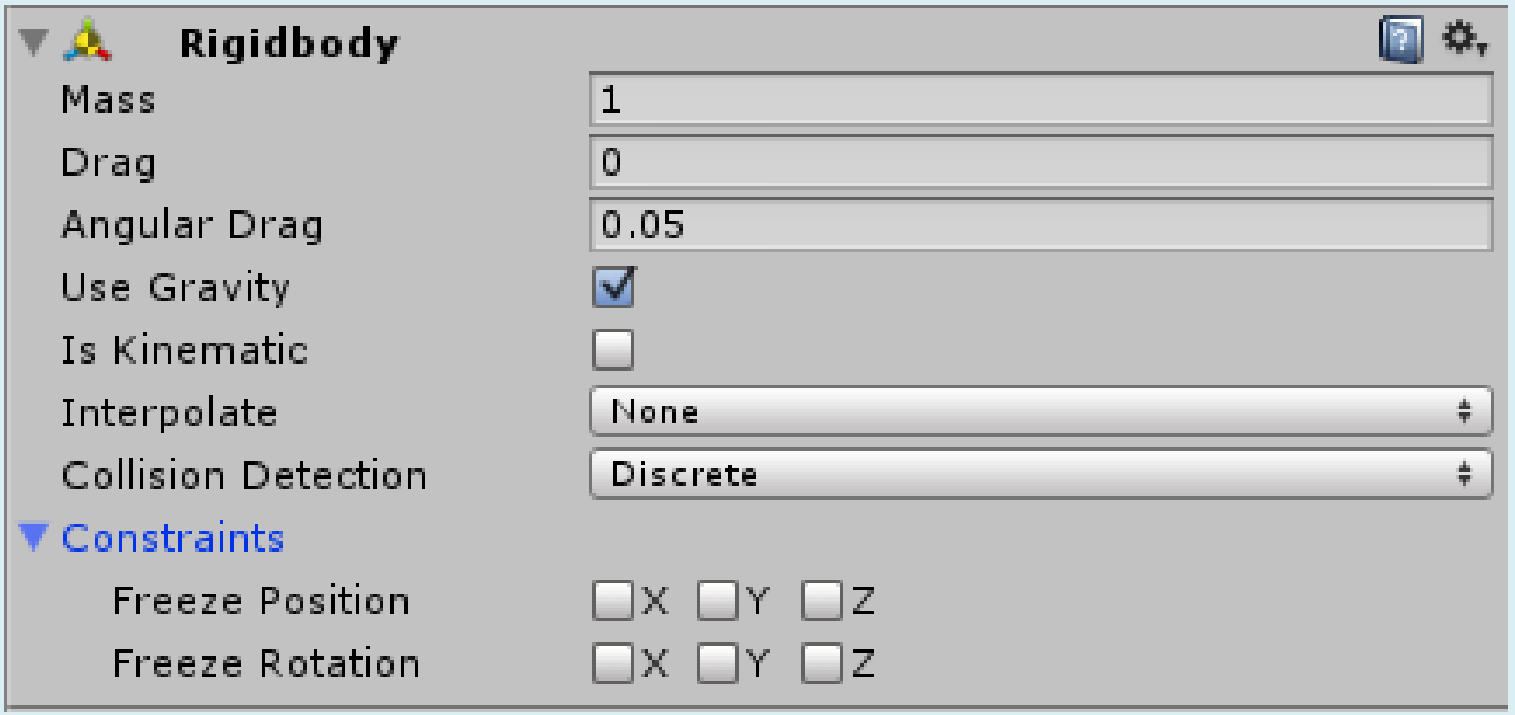


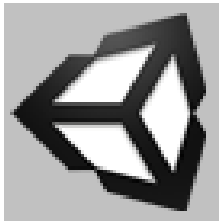
Giới thiệu

- Xác định va chạm:
  - Sử dụng thành phần Collider.
  - Là một lưới vô hình bao quanh Game Object, dùng để xác định va chạm với một Game Object khác.
- Điểm va chạm vô hình:
  - Dùng kỹ thuật Ray Casting.
  - Là kỹ thuật xác định điểm cuối của một tia vô hình trong không gian 3D.

Rigidbody

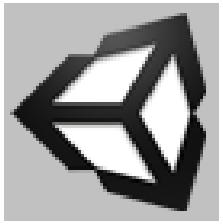
- Thành phần này cho phép Game Object hoạt động dưới ảnh hưởng của vật lý.





Rigidbody

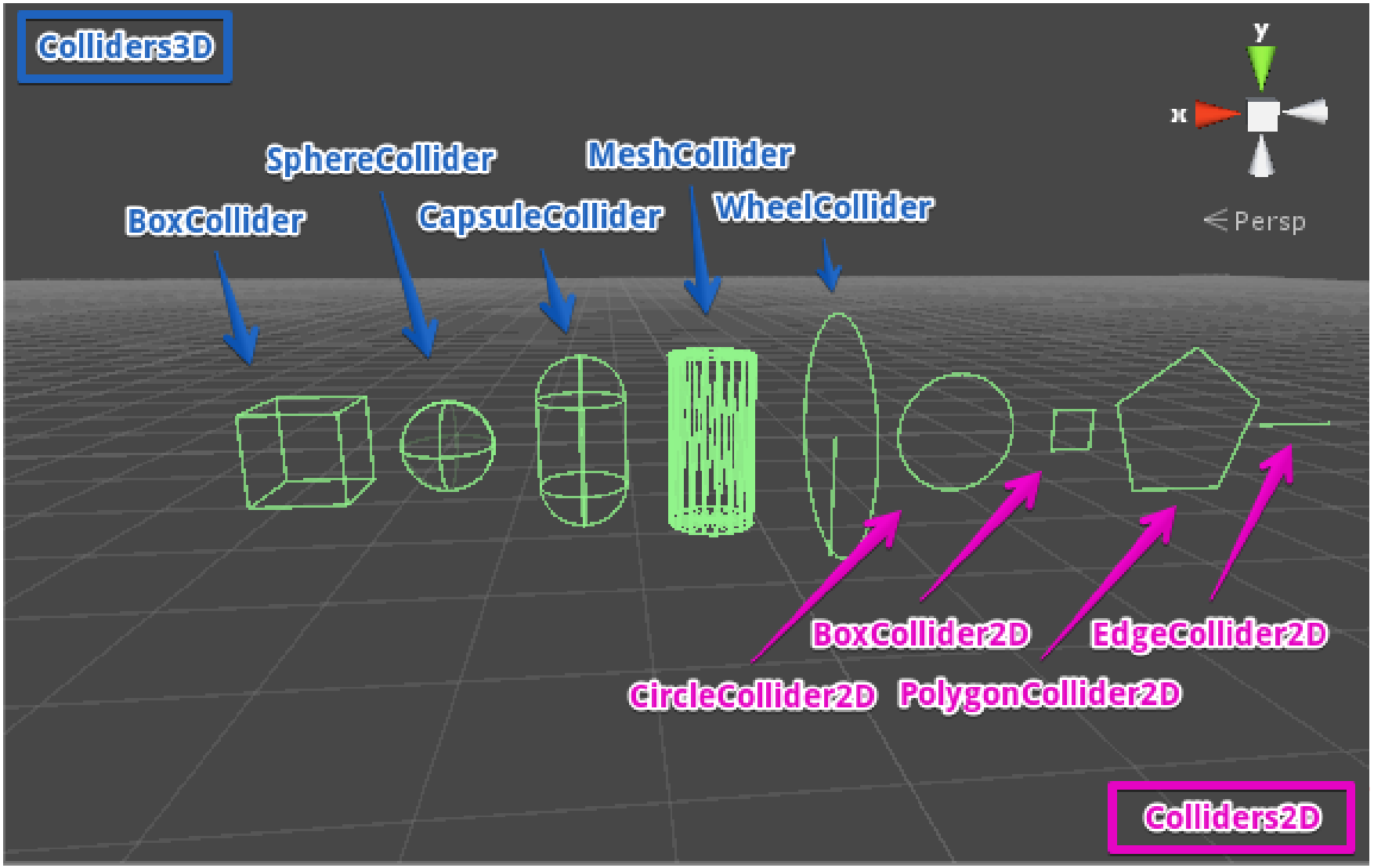
- Các thuộc tính đi kèm:
  - Mass: Khối lượng của vật thể.
  - Drag: Lực cản không khí, 0 là không có lực cản.
  - Angular Drag: Lực cản mô men quay.
  - Use Gravity: Có sử dụng trọng lực hay không.
  - Is Kinematic: Bỏ qua ảnh hưởng vật lý, dùng để điều khiển trong Code.
  - Interpolate: Mức độ giảm xóc của đối tượng:
  - Collision Detection: Cách thức va chạm
  - Constraints: Ràng buộc khi va chạm.

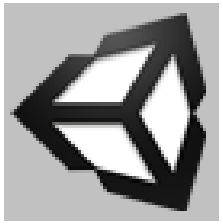


Collider

- Là một đối tượng vô hình bao quanh đối tượng game, nhằm xác định mức độ va chạm.
- Có các loại Collider như sau:
  - **Box Collider:** dành cho khối hộp.
  - **Capsule Collider:** dành cho capsule
  - **Mesh Collider:** Dành cho đối tượng phức tạp
  - **Sphere Collider:** Dành cho khối cầu
  - **Wheel Collider:** Dành cho bánh xe
  - **Terrain Collider:** Dành cho địa hình.

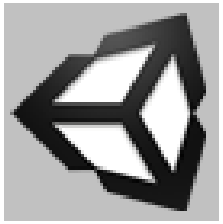
Các loại Collider





Collider

- Để sử dụng Collider, vào Components chọn Physics, chọn Collider tương ứng.
- Một số thuộc tính thông dụng:
  - **Is Trigger**: xác định là loại va chạm xuyên qua (trigger) hay va chạm không xuyên qua (collision).
  - **Material**: bề mặt va chạm hiệu ứng vật lý.
  - **Center**: vị trí tương đối của Collider so với object.
  - Ngoài ra, mỗi loại Collider sẽ có vài thông số riêng.
- Demo



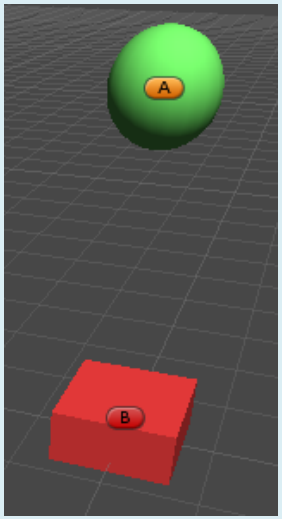
Các phương thức Collider

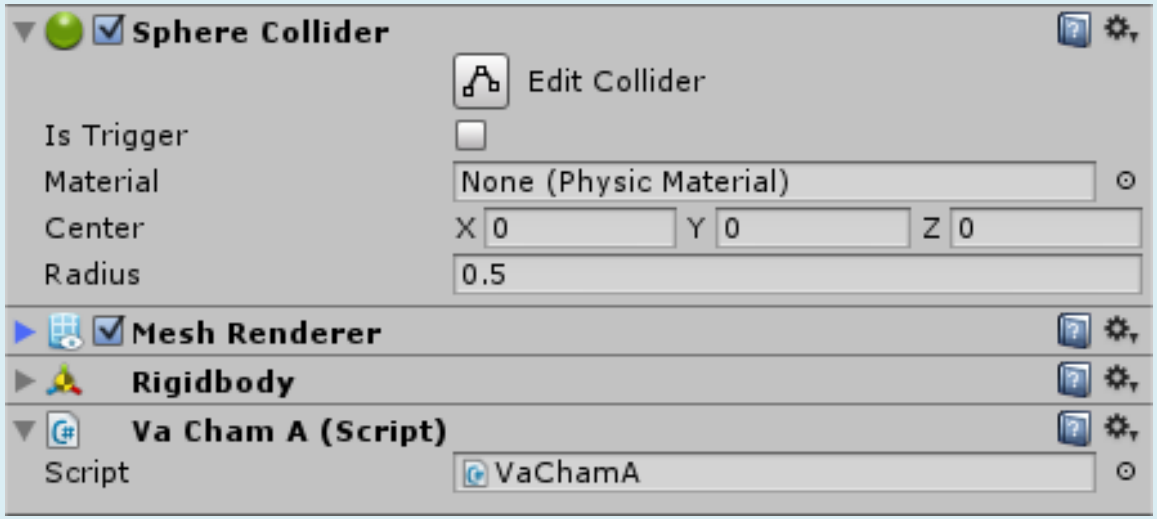
- **OnCollisionEnter**: Xác định va chạm khi hai đối tượng bắt đầu chạm nhau
- **OnTriggerEnter**: Ứng với va chạm loại trigger, là va chạm khi đối tượng này đi vào đối tượng kia
- **OnTriggerExit**: Sự kiện khi kết thúc va chạm

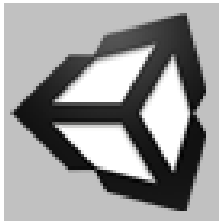
Lưu ý: Các hàm này chỉ hoạt động khi thêm vào thành phần Rigid Body

Ví dụ 1

- Hàm sự kiện va chạm được đặt trong đoạn script là component của object có Collider.
- Ở ví dụ này, ta tạo mới script *VaChamA* cho hình cầu A



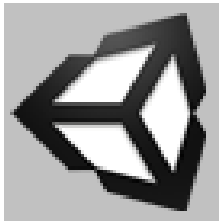




Viết mã cho Collision

```
public class VaChamA : MonoBehaviour {
    //Collider của object tiếp xúc với collider của object kia
    void OnCollisionEnter(Collision col1)
    {
        print(gameObject.name + " bắt đầu va chạm với " + col1.gameObject.name);
    }
    //Collider của object này đi vào collider của object kia
    void OnTriggerEnter(Collider col2)
    {
        print(gameObject.name + " đang va chạm với " + col2.gameObject.name);
    }
    //Collider của object này thoát khỏi collider của object kia
    void OnTriggerExit(Collider col2)
    {
        print(gameObject.name + " kết thúc va chạm với " + col2.gameObject.name);
    }
}
```

❖Chạy và xem kết quả



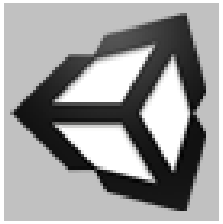
Ví dụ 1

- Cải tiến để khối cầu tự di chuyển và báo khi va chạm

```
public float _speed;
// Update is called once per frame
void Update () {
    // Tự động di chuyển
    transform.Translate(-_speed * Time.deltaTime, 0, 0);
}
// Bắt đầu xảy ra va chạm
void OnCollisionEnter(Collision col)
{
    print("Đối tượng "+gameObject.name + " bắt đầu va chạm với: " + col.gameObject.name);
}
// trong khi va chạm
void OnTriggerEnter(Collider col)
{
    print("Đối tượng " + gameObject.name + " đang va chạm với: " + col.gameObject.name);
}
// Thoát va chạm
void OnTriggerExit(Collider col)
{
    print("Đối tượng " + gameObject.name + " kết thúc va chạm với: " + col.gameObject.name);
}
```

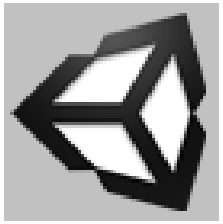
Ví dụ 2

- Thực hiện bắn viên đạn trong 2D



Đạn di chuyển

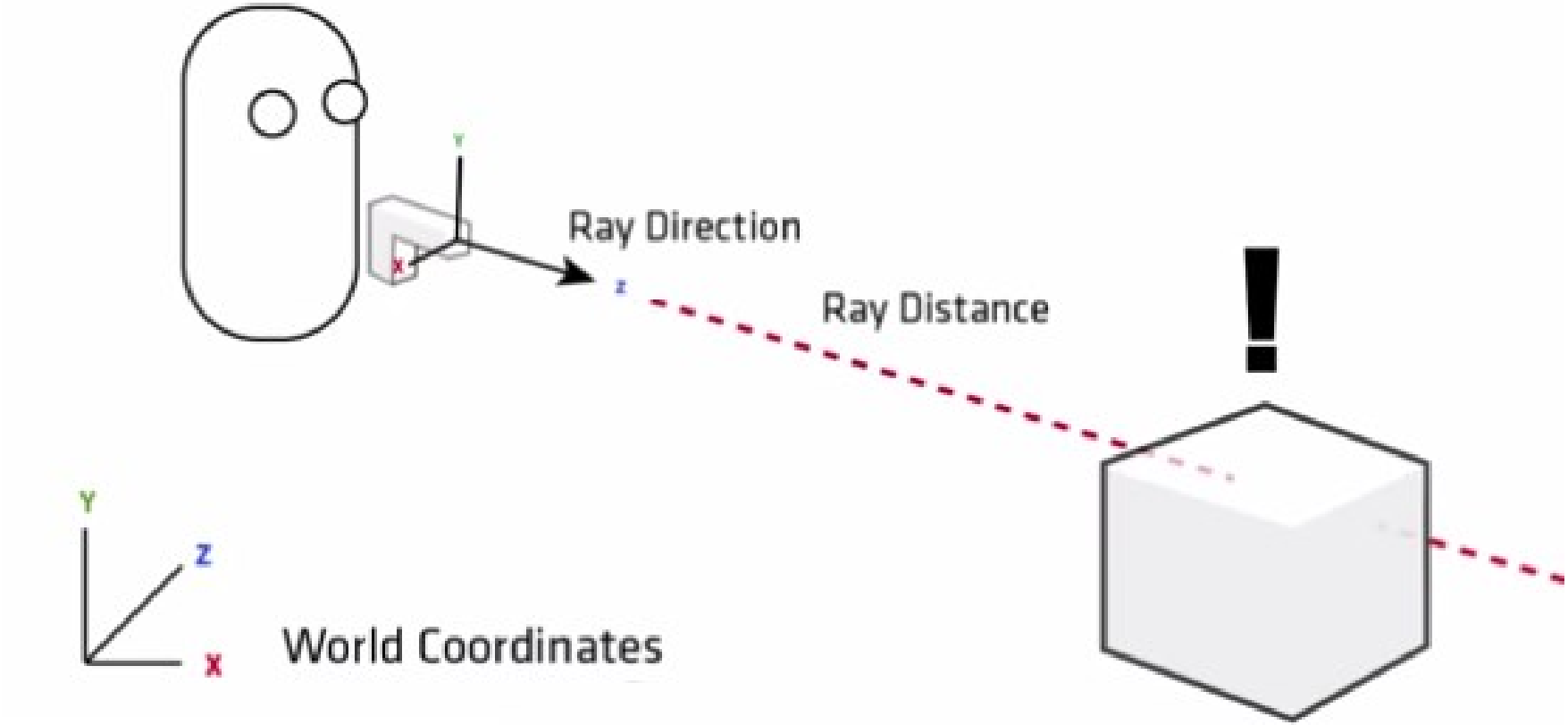
```
public class BulletMove : MonoBehaviour {

    float _speed = 5.0f;
    // Use this for initialization
    // Update is called once per frame
    void Update () {
        transform.Translate(_speed * Time.deltaTime, 0, 0);
    }

    void OnCollisionEnter2D(Collision2D col1)
    {
        print(gameObject.name + " OnCollisionEnter2D voi " + col1.gameObject.name);
    }

    void OnTriggerEnter2D(Collider2D col2)
    {
        print(gameObject.name + " OnTriggerEnter2D voi " + col2.gameObject.name);
    }

    void OnTriggerExit2D(Collider2D col2)
    {
        print(gameObject.name + " OnTriggerExit2D voi " + col2.gameObject.name);
    }
}
```

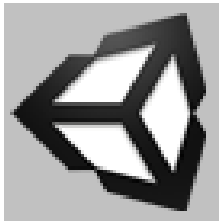


Súng di chuyển & bắn

```
public class BanDan : MonoBehaviour {

    public GameObject _obj;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        transform.Translate(-Input.GetAxis("Vertical") * Time.deltaTime * 5.0f,0, 0);
        if (Input.GetKeyUp(KeyCode.Space))
        {
            Instantiate(_obj, transform.position, Quaternion.identity);
        }
    }
}
```

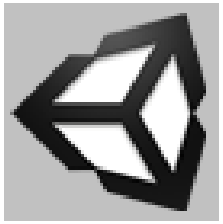
Kỹ thuật Ray casting





Ray casting

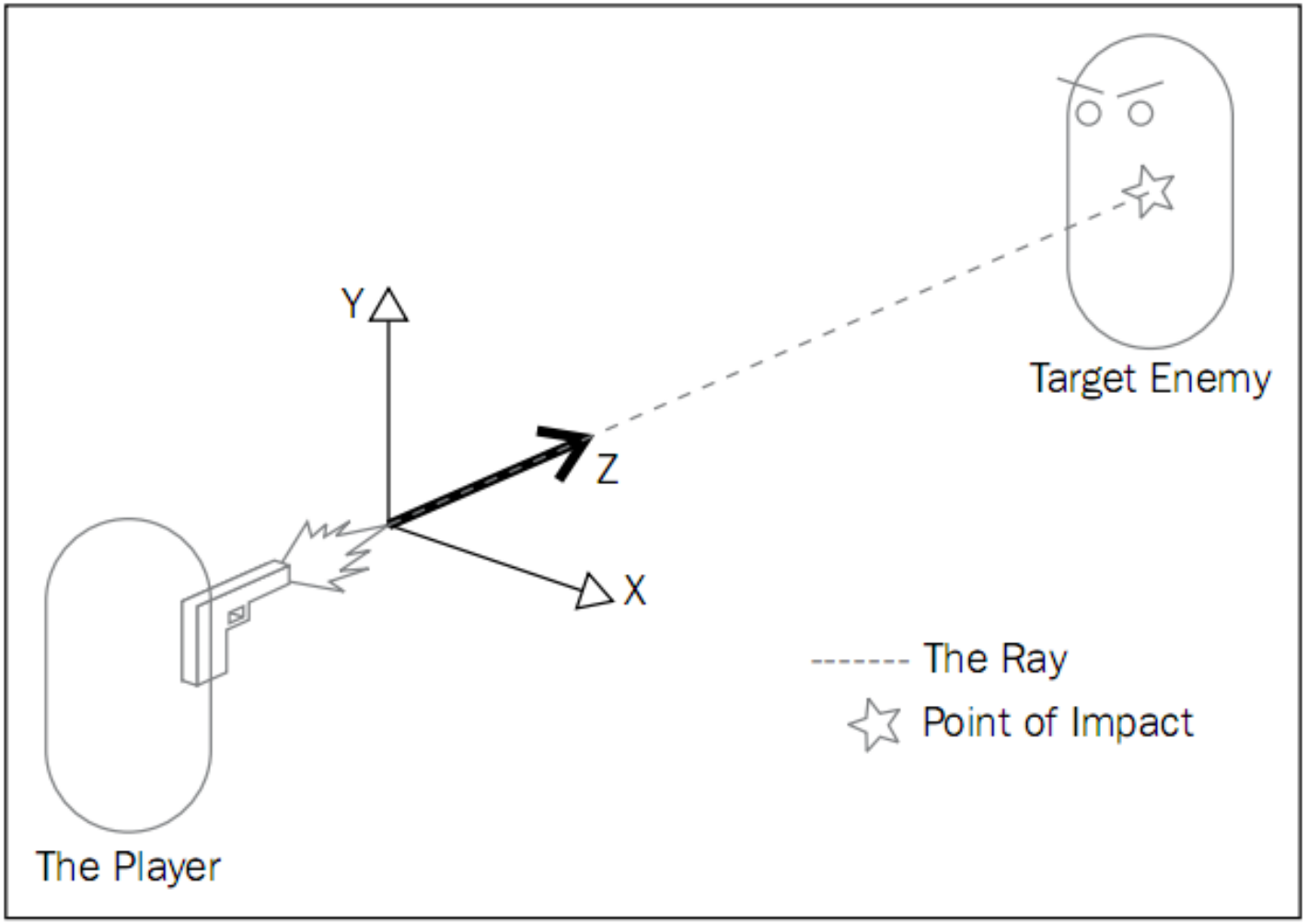
- Có thể xác định khi nào mở cửa bằng khi người chơi quay mặt về phía cửa.
- Khi đó chỉ cần 1 vector chỉ hướng và khoảng cách để xác định khi nào va chạm.
- Raycasting có nhiều ứng dụng trong việc tìm đích của viên đạn, xác định vật thể sắp va chạm khi rơi…
- Xem vấn đề sau đây khi xác định đường đi của viên đạn.

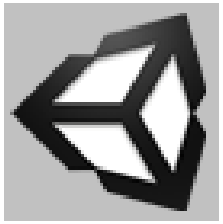


Predictive collision detection

- Thay vì kiểm tra va chạm với một Object thực tế, người ta sẽ dự đoán va chạm dựa trên 1 tia được bắn ra vô hình.
- Việc dùng tia để xác định va chạm từ trước có nhiều tác dụng như:
  - Lấy về thông tin Collider sẽ va chạm.
  - Đánh địa chỉ và đưa script cho phù hợp.
  - Thực hiện hiệu ứng tương ứng

Predictive collision detection

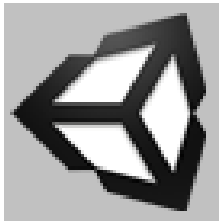




Example Ray casting

```
    // Update is called once per frame
void Update()
{

    var up = transform.TransformDirection(Vector3.up);
    //note the use of var as the type. This is because in c# you
    // can have lamda functions which open up the use of untyped variables
    //these variables can only live INSIDE a function.
    RaycastHit hit;
    Debug.DrawRay(transform.position, -up * 2, Color.green);

    if (Physics.Raycast(transform.position, -up, out hit, 2))
    {

        Debug.Log("HIT");

        if (hit.collider.gameObject.name == "floor")
        {
            Destroy(GetComponent("Rigidbody"));
        }
    }
}
```

Ví dụ về Raycasting

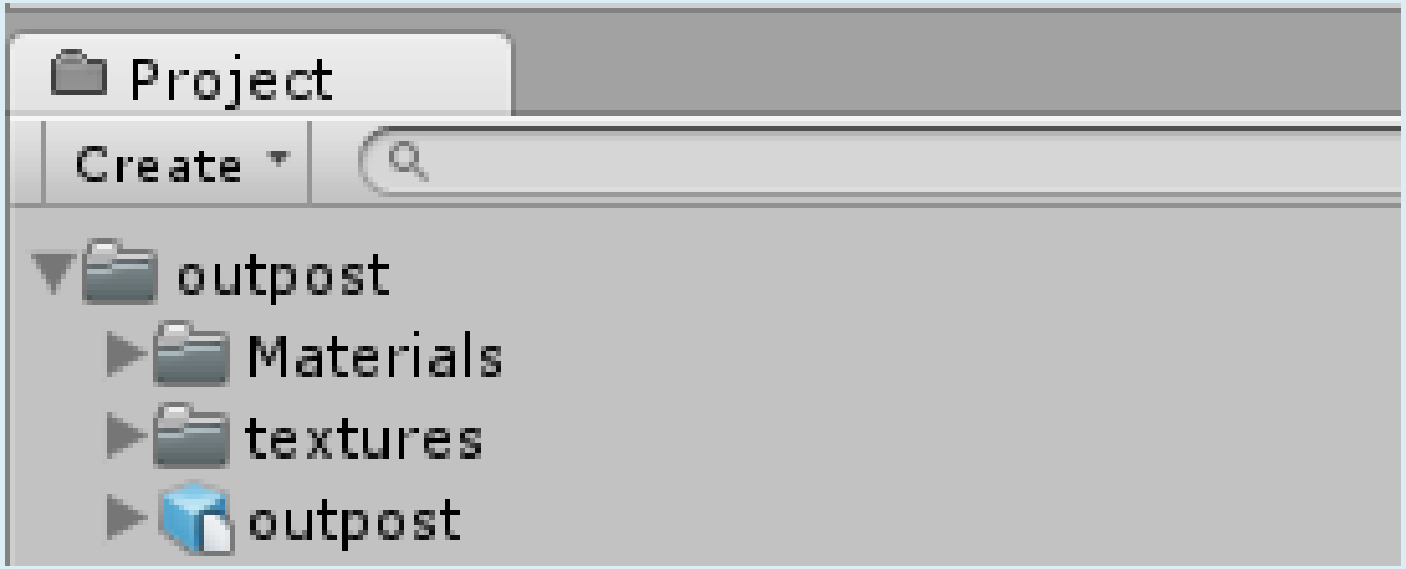


Ví dụ về 2 phương pháp

- Trong ví dụ này, ta sẽ dùng 2 phương pháp là Collision và Raycasting để mở cánh cửa khi nhân vật tiến lại gần.

Adding the outpost

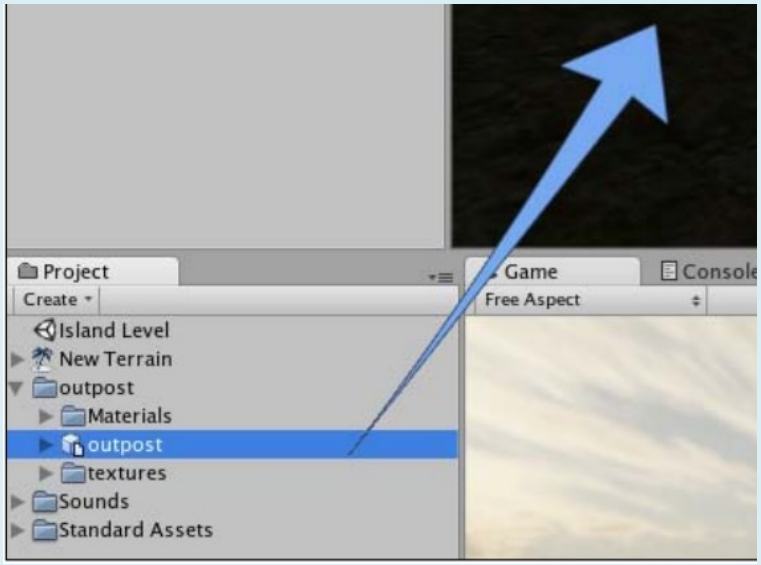
- Tải gói *Outpost.unitypackage*[1] về và Import vào dự án.

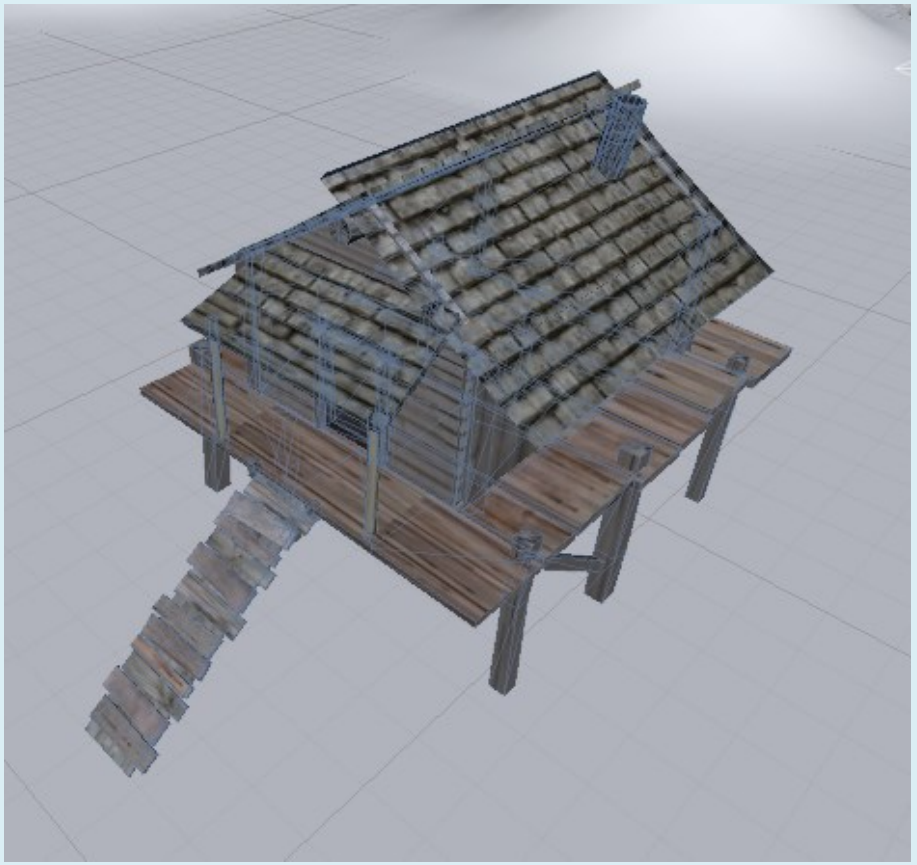


[1]. Sinh viên có thể liên hệ giáo viên để lấy gói này

Adding the outpost

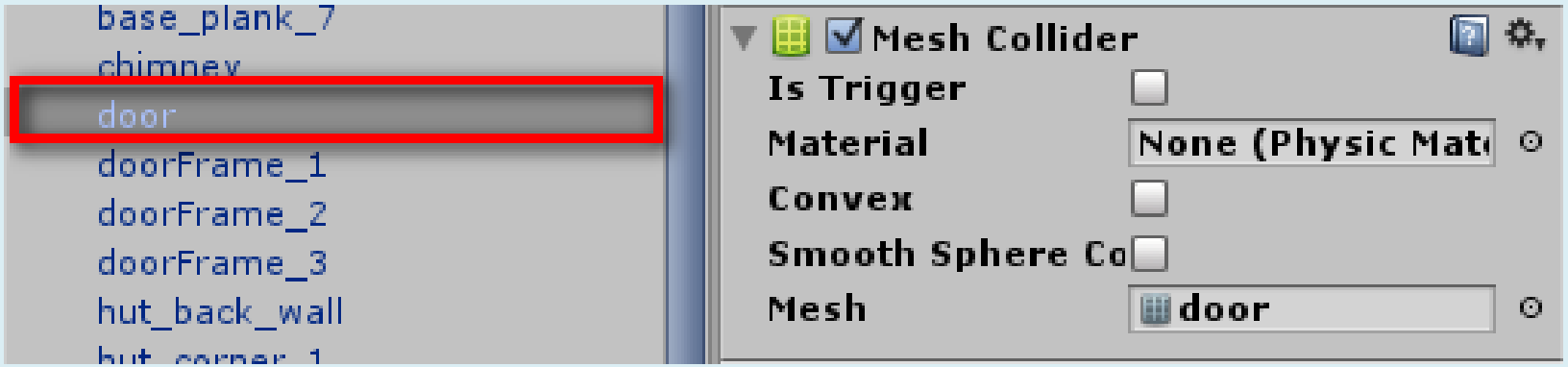
- Đưa đối tượng outpost vào Scense và đặt ở vị trí (500, 30.8, 505)
  - Thiết lập Scale Factor trong bản gốc là 1.5





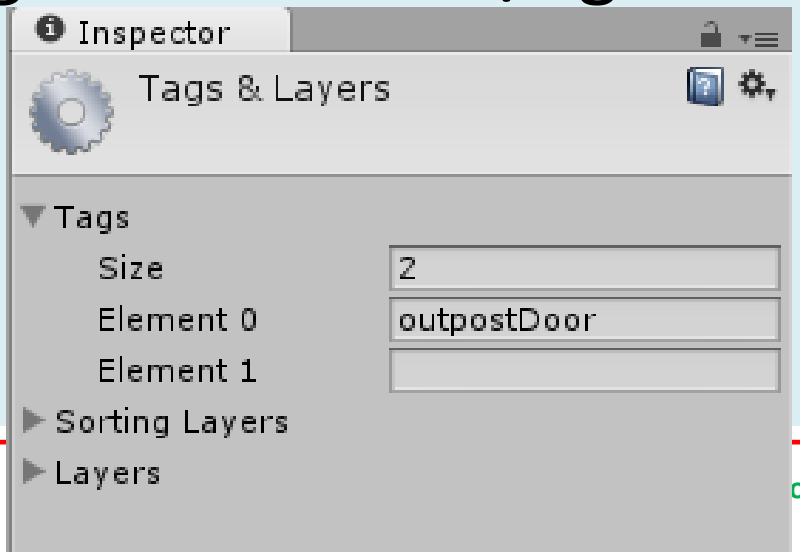
Tag Colliders to the door

- Tìm đến door trong outpost trên Scene và thêm vào một Mesh Collider.



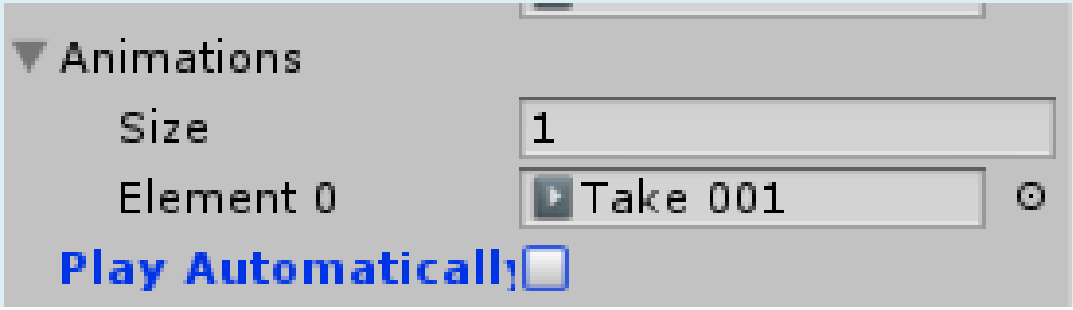
Address Tag to the door

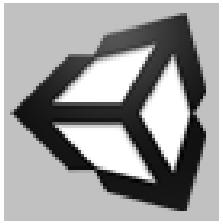
- Cần phải đánh dấu bằng thẻ tag với đối tượng door để sử dụng cho việc viết script sau này.
- Chọn đối tượng door, trên Inspector, chọn Tag và thêm thẻ mới (tên outpostDoor).
- Sau đó gán cho đối tượng với thẻ này.



Disabling automatic animation

- Outpost có một số Animation được gán sẵn, để không làm ảnh hưởng đến quá trình dò tìm đối tượng ta tắt chế độ tự động của Animation.
- Vào Inspector, chọn Animations và bỏ chọn Play Automatically.



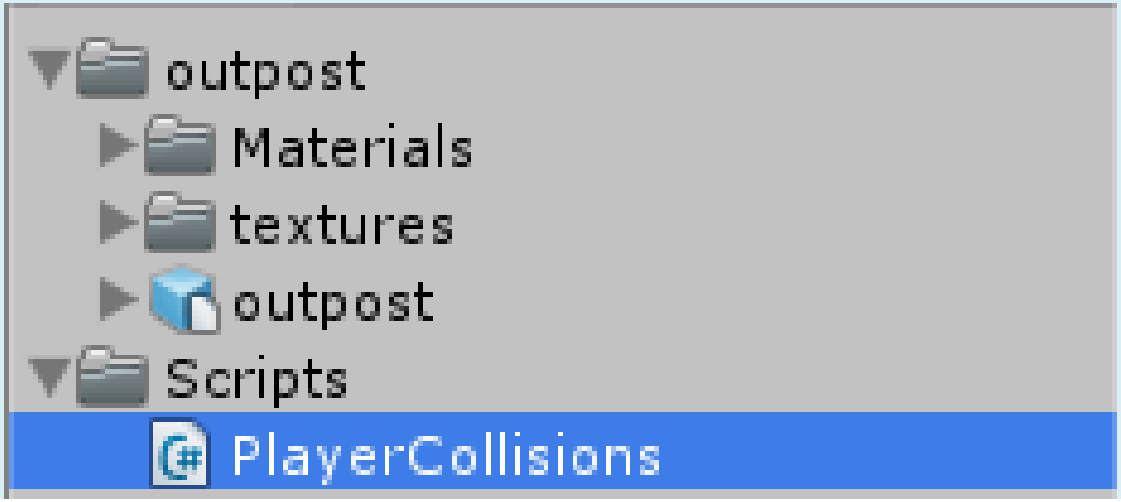


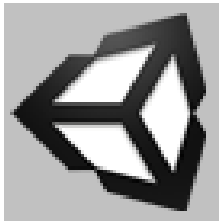
Opening the outpost

- Trong phần này, sẽ tìm hiểu & thực hiện hai phương pháp:
  - Mở cửa Outpost bằng cách dò tìm va chạm.
  - Mở cửa Outpost bằng phương pháp Ray Casting.

Approach 1:Collision detection

- **Creating new assets**: Tạo một Asset mới, đặt tên là Script, tạo một tập tin Script trong đó, đặt tên là *PlayerCollisions.cs.*
- Nhấp đôi vào file này, để mở trên Editor.





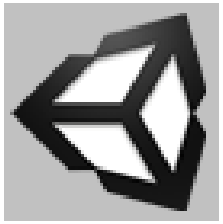
Approach 1:Collision detection

Khai báo các biến như sau:

```
private bool doorIsOpen = false; // kiểm tra nếu cửa đang mở
private float doorTimer = 0.0f; // biến thời gian cho từng lần mở cửa
private GameObject currentDoor; // Đối tượng cửa hiện hành
float doorOpenTime = 3.0f; // Thời gian để cửa mở
AudioClip doorOpenSound; // Âm thanh mở cửa
AudioClip doorShutSound; // Âm thanh đóng cửa
```

Sau đó viết phương thức

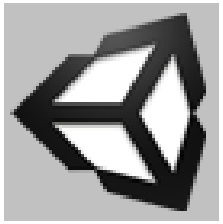
```
void OnControllerColliderHit(ControllerColliderHit hit)
{

}
```



Approach 1:Collision detection

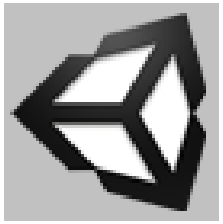
Thêm nội dung cho phương thức như sau:

```
void OnControllerColliderHit(ControllerColliderHit hit)
{
    // Nếu đối tượng va chạm có thẻ là outpostDoor và cửa đang đóng
    if (hit.gameObject.tag == "outpostDoor" && doorIsOpen == false)
    {
        OpenDoor(); // Thì mở cửa
    }
}
void OpenDoor()
{
    // do something
}
```



Giải thích

- Phương thức OnControllerColliderHit với biến hit sẽ lưu trữ toàn bộ các thông tin của một đối tượng bị va chạm.
- Nếu đối tượng va chạm có thẻ được gắn là “outpost” và trạng thái cửa đang đóng thì gọi phương thức Mở cửa



Approach 1:Collision detection

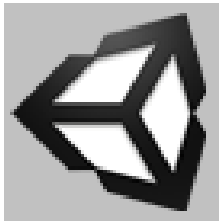
Nội dung của phương thức OpenDoor được viết như sau:

```
void OpenDoor()
{
    //chạy tập tin âm thanh doorOpenSound
    audio.PlayOneShot(doorOpenSound);
    doorIsOpen = true; // Chuyển trạng thái cửa về dạng đóng
    // Tìm đối tượng có tên là outpost
    GameObject myOutpost = GameObject.Find("outpost");
    // Chạy phần Animation có tên "Take 001"
    myOutpost.animation.Play("Take 001");
}
```

Approach 1:Collision detection

Viết phương thức đóng cửa và gọi nó trong phương thức Update như sau:

```
// Update is called once per frame
void Update()
{
    // Nếu cửa đang ở trạng thái mở
    if (doorIsOpen)
    {
        // Tăng biến doorTimer lên từng giây
        doorTimer += Time.deltaTime;

        if (doorTimer > 3) // Nếu quá 3s
        {
            shutDoor(); // Đóng cửa
            doorTimer = 0.0f;
        }
    }
}
```



Approach 1: Collision detection

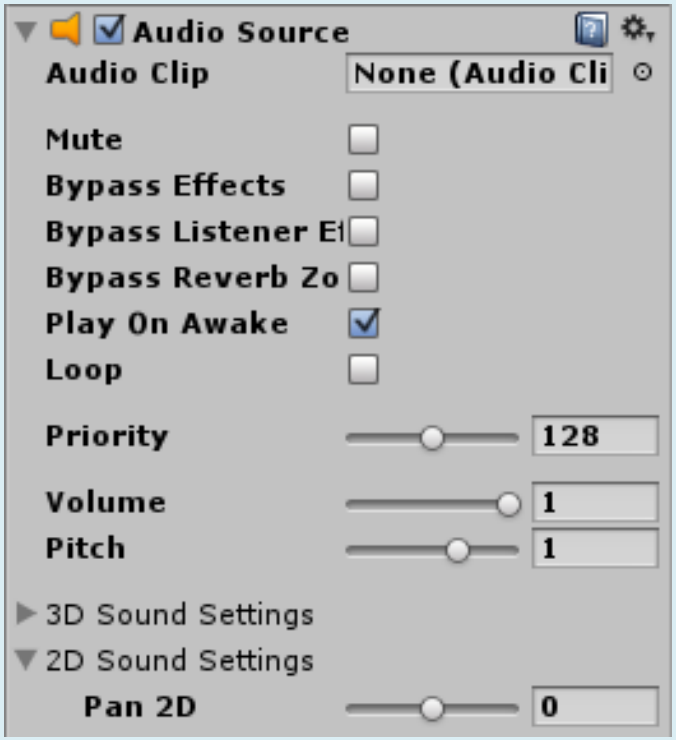
Tương tự như phương thức openDoor, phương thức shutDoor được viết như sau:

```
void shutDoor()
{
    // Chạy tập tin đóng cửa
    audio.PlayOneShot(doorShutSound);
    // Cập nhật trạng thái đóng cửa
    doorIsOpen = false;
    // Tìm kiếm đối tượng có tên là outpost
    GameObject myOutpost = GameObject.Find("outpost");
    // Thực thi animation có tên là Take 001
    myOutpost.animation.Play("Take 001");
}
```

Approach 1: Collision detection

❖Đính kèm script:

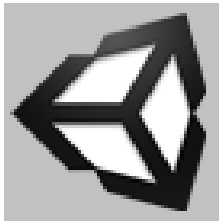
- ➢Thêm **Audio Source** cho Player



Approach 1: Collision detection

❖Đính kèm script:

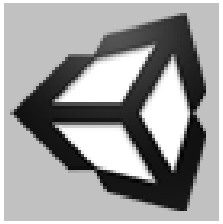
- ➢Đính kèm tập tin Script cho Player và đưa các tập tin âm thanh cho Player



Approach 2—Ray casting

- Che phương thức OnControllerColliderHit

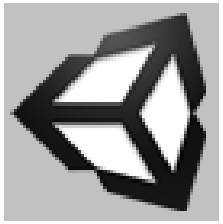
```
/*void OnControllerColliderHit(ControllerColliderHit hit)
{
    if (hit.gameObject.tag == "outpostDoor" && doorIsOpen == false)
    {
        OpenDoor();
    }
}*/
```



Approach 2—Ray casting

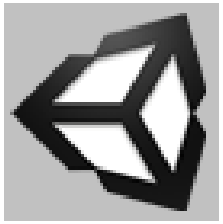
- Sử dụng Ray Casting bằng mã lệnh như sau trong phương thức Update:

```
// Update is called once per frame
void Update()
{
    RaycastHit hit;
    if (Physics.Raycast(transform.position, transform.forward, out hit, 5.0f))
    {
        if (hit.collider.gameObject.tag == "outpostDoor"
                    && doorIsOpen == false)
        {
            currentDoor = hit.collider.gameObject;
            Door(doorOpenSound, true, "dooropen", currentDoor);
        }
    }
}
```



Approach 2—Ray casting

- Giải thích: Phương thức Raycast có 4 đối số:
  - Ví trí ban đầu của tia: transform: đối tượng được gắn script.
  - Hướng của tia: *transform.forward:* hướng về phía trước của đối tượng.
  - RaycastHit: cấu trúc sẽ lưu trữ va chạm.
  - Độ dài tia: tính bằng mét.



Demo