Filière: CII-3- GLSI

Réalité virtuelle et animation 3D

Chapitre 4: Intéractions

Enseignant: M. Majdi JRIBI

Introduction

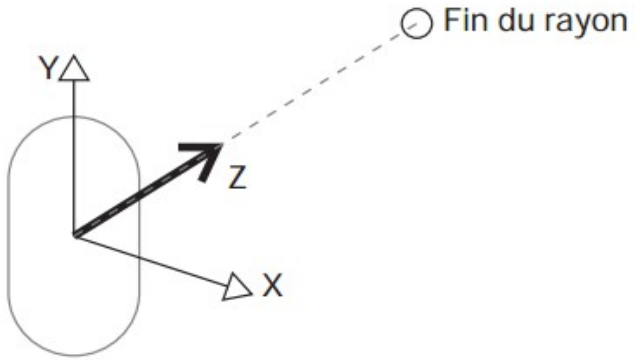
- Pour détecter les interactions physiques entre les objets de jeu, la méthode la plus courante consiste à utiliser un ***collider*** – un filet invisible tout autour d'un objet chargé de détecter les collisions avec d'autres objets.

- La détection de collision regroupe à la fois la détection et la récupération des informations provenant de ces collisions.

Introduction(Suite)

- Il est non seulement possible de détecter que deux colliders (composants de collision) interagissent, mais également d'anticiper une collision et d'effectuer de nombreuses autres tâches à l'aide d'une technique appelée raycasting.

- Nous allons voir en détail, deux éléments essentiels au développement d'un jeu : **la détection de collision** et **le raycasting**(tracé de rayon).



I. Les collisions

- Lorsque des objets entrent en collision dans n'importe quel moteur de jeu, des informations sur cet événement deviennent alors disponibles. En enregistrant différentes informations au moment de l'impact, le moteur de jeu peut ensuite réagir de manière réaliste.

- Dans un jeu tenant compte des lois physiques par exemple, si un objet tombe au sol d'une certaine hauteur, le moteur a besoin de savoir quelle partie de cet objet touche le sol en premier pour contrôler de façon correcte et réaliste la réaction de l'objet à cet impact.

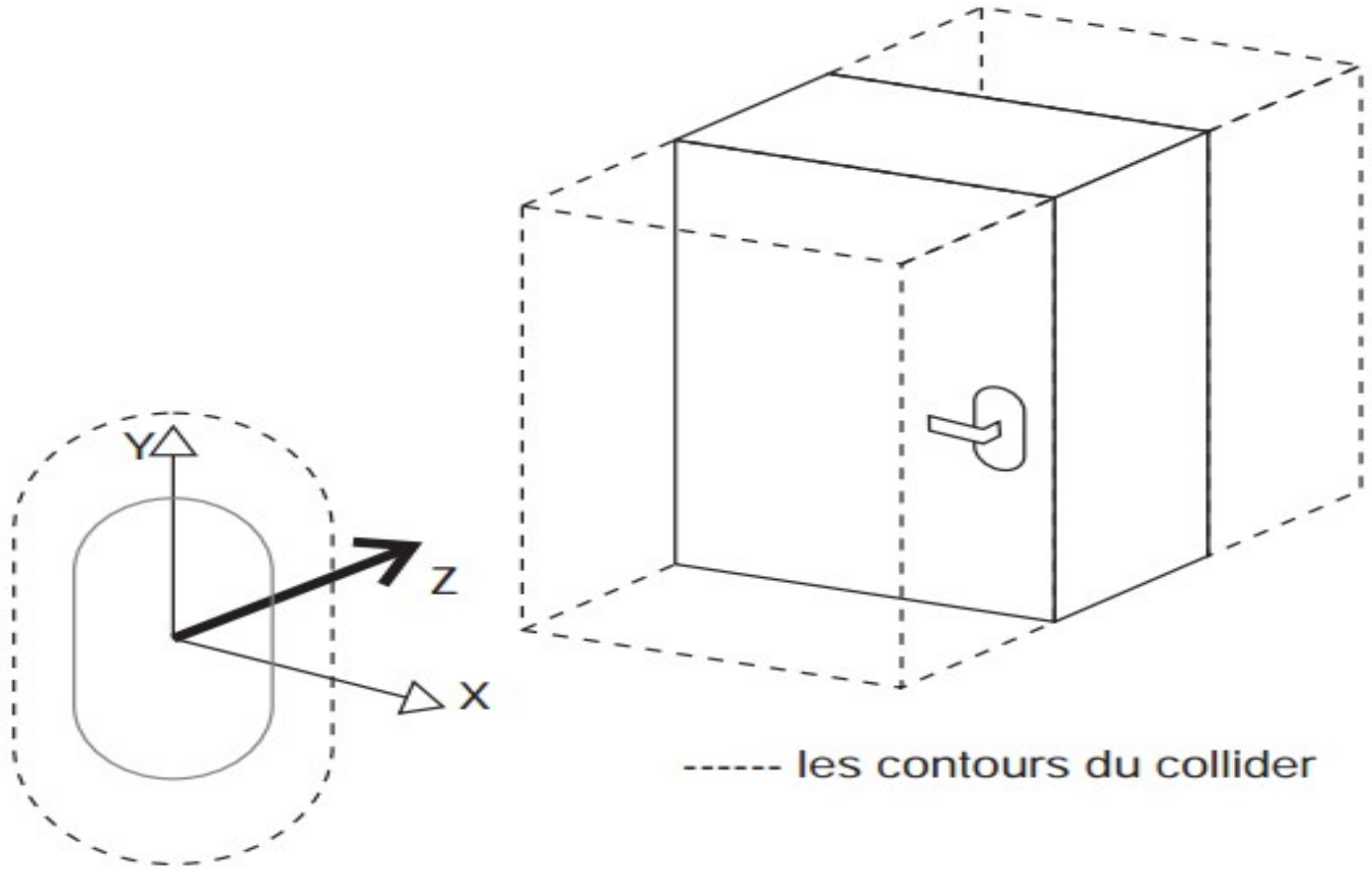
I. Les collisions

- Dans le cas de l'ouverture d'une porte, vous avez besoin de détecter les collisions entre le collider du personnage du joueur et celui situé sur – ou près de – la porte.

- Cela n'aurait guère de sens de détecter les collisions ailleurs, puisque l'animation de la porte doit se déclencher lorsque le joueur est assez proche pour espérer qu'elle s'ouvre ou pour franchir le seuil.

- Par conséquent, vous allez vérifier les collisions entre le collider du personnage et celui de la porte. Vous devez toutefois étendre la profondeur du collider de la porte

I. Les collisions



I. Les collisions

- Si le collider de la porte dépassait la surface visuelle de la porte, le personnage entrerait en collision avec une surface invisible qui l'arrêterait net. En outre, bien que vous puissiez utiliser cette collision pour déclencher l'animation d'ouverture de la porte, l'impact initial dans le collider semblerait peu naturel pour le joueur, ce qui nuirait à l'immersion dans le jeu.

- Si la détection de la collision fonctionne très bien entre le collider du personnage et celui de la porte d'un point de vue technique, cette approche présente donc trop d'inconvénients pour qu'un développeur de jeux créatif ne recherche pas une approche plus intuitive. C'est là qu'intervient le **raycasting.**

II. Le raycasting

- Bien qu'il soit possible de détecter les collisions entre le collider du personnage et un collider qui s'adapte à l'objet porte, il semble plus judicieux que la porte s'ouvre lorsque le joueur lui fait face et à une certaine distance, ce qui peut être réalisé en projetant un rayon d'une longueur limitée depuis l'avant du personnage.

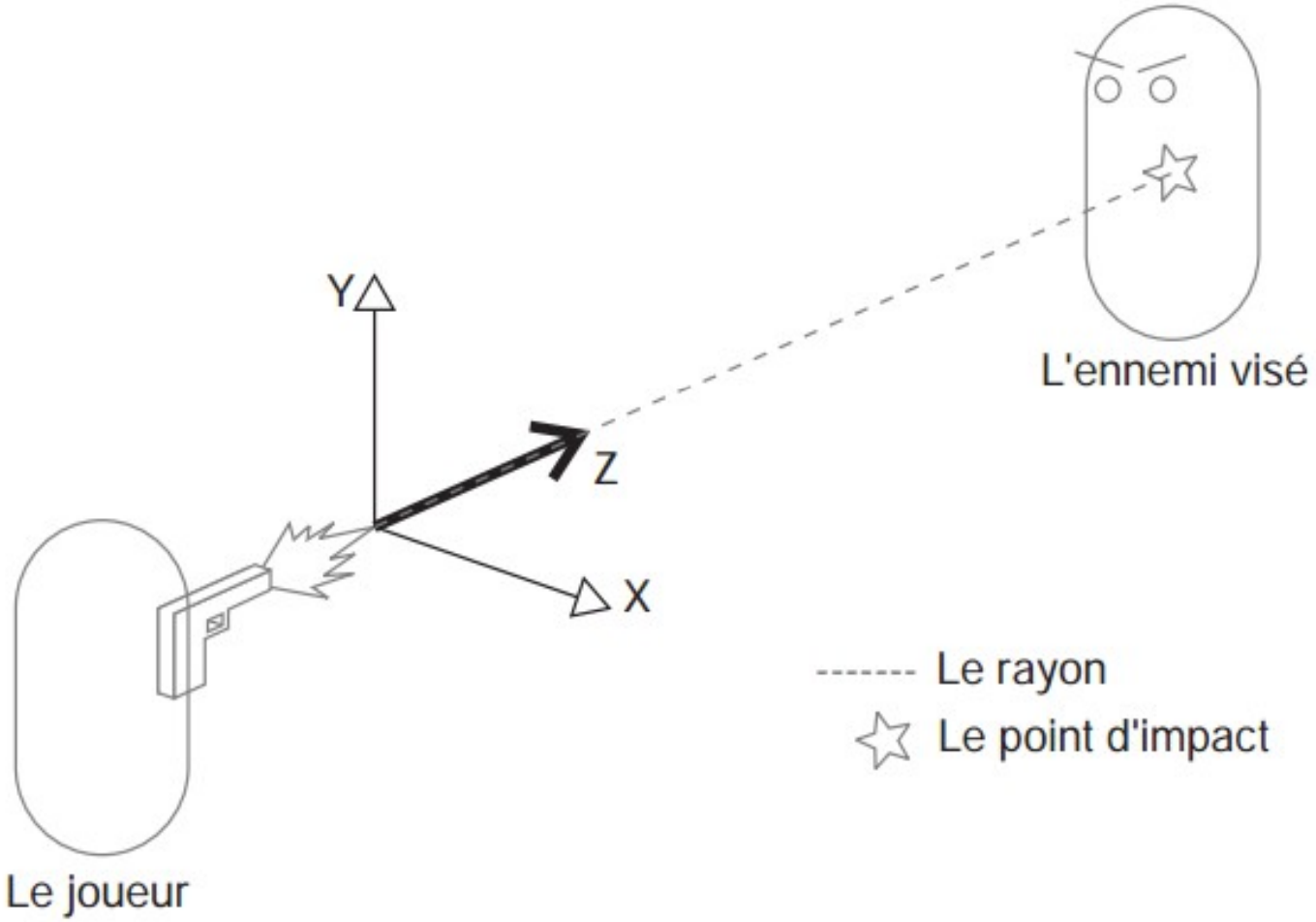
- Cette méthode garantit également que le joueur ne peut pas ouvrir la porte s'il lui tourne le dos ; avec le raycasting, il doit faire face à l'objet pour l'utiliser, ce qui est plus logique.

II. Le raycasting

- Dans le raycasting, un rayon est tracé dans la direction visée. Il est alors possible de récupérer de informations sur le collider que ce rayon touche. Une fois le collider identifié, un script peut alors désigner l'objet de jeu lui-même et scénariser son comportement en conséquence.

- Vous pouvez obtenir des informations très détaillées, comme le point d'impact, et les utiliser pour définir la réaction de la cible (faire reculer l'ennemi dans une direction particulière par exemple).

II. Le raycasting



III. Exemple

- Dans ce qui reste, nous allons illustrer les deux mécanismes de détection de collisions sur un exemple.
- Il s'agit de déclencher l'ouverture et la fermeture de la porte d'une cabane(outpost), créée avec le logiciel Maya.
- Trois animations(Idle, ouverture, fermeture) sont attachées à la porte.
- Les mécanismes de détection de collisions vont nous permettre de déclencher ces animations.

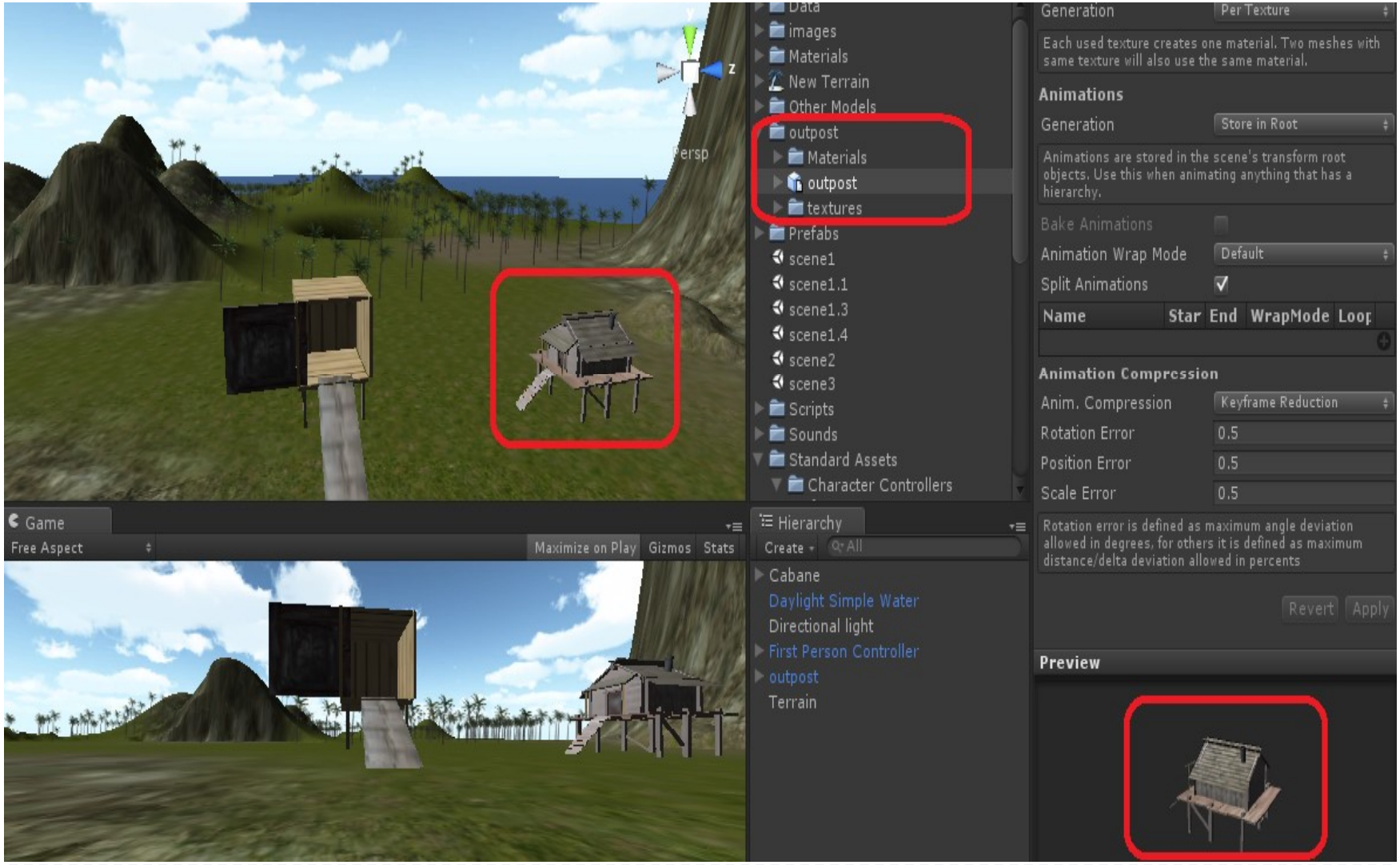
III. Exemple: L'ajout de l'avant-poste

- Avant de commencer à utiliser à la fois la détection de collision et le raycasting pour ouvrir la porte, vous devez intégrer à la scène l'avant-poste(Cabane).

- Plus loin, vous écrirez le script qui contrôle les états d'animation de ce modèle.

- Une fois l'avant-poste dans le panneau Scene, son nom apparaît dans le panneau Hierarchy et il est automatiquement sélectionné.

III. Exemple: Positionner le modèle



III. Exemple: Redimensionner le modèle

- Modifiez la valeur **Scale Factor** de 1 à 2, puis cliquez sur le bouton **Apply**. Ainsi, la taille de l'avant-poste est suffisante pour que l'objet Character Controller passe par la porte et la taille de la pièce sera réaliste lorsque le personnage se trouvera à l'intérieur.

III. Exemple: Colliders et tag de la porte

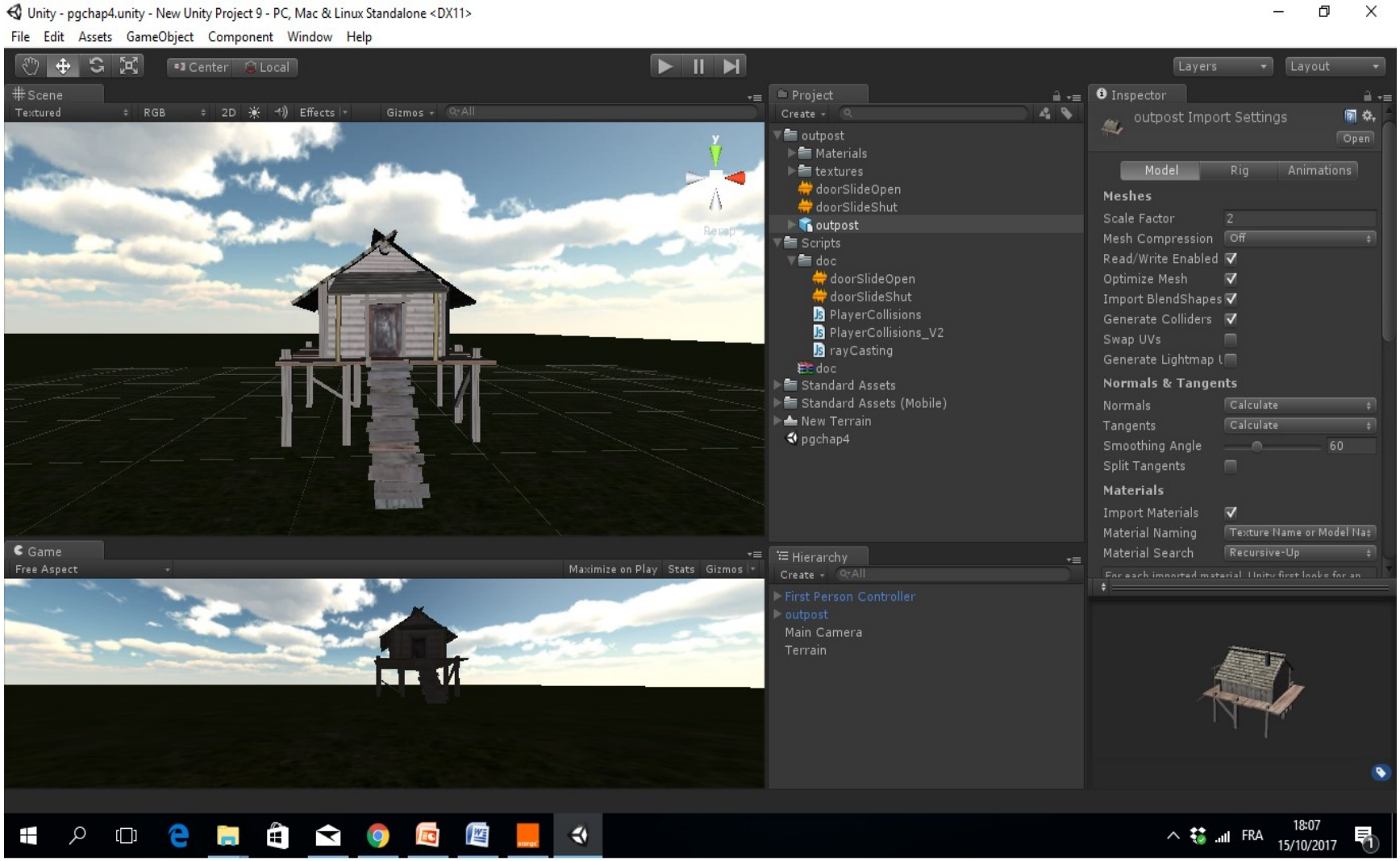
- La porte doit être identifiée comme un objet particulier pour pouvoir s'ouvrir lorsqu'elle entre en collision avec le joueur.

- Pour cela, l'objet doit posséder un composant Collider

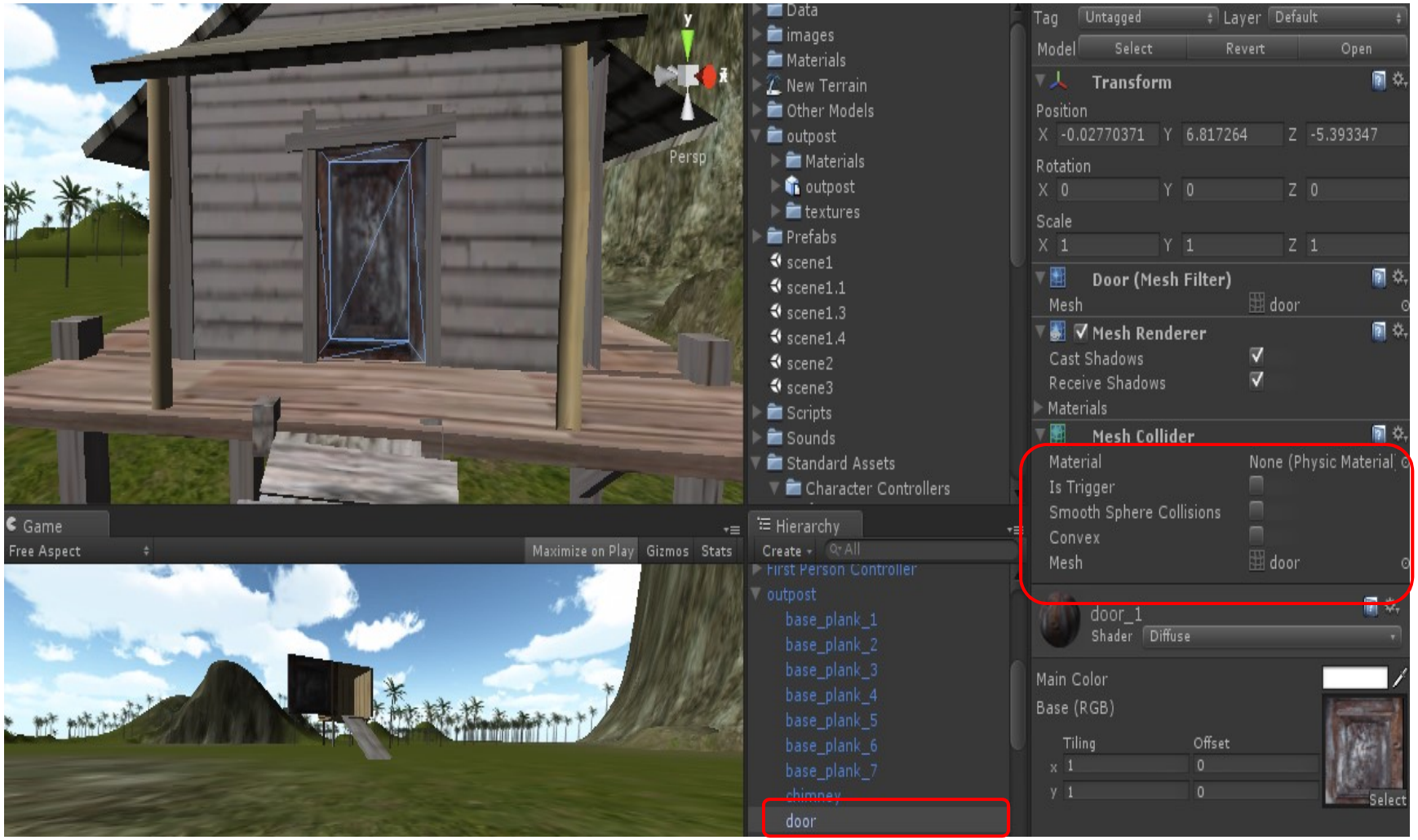
et un tag spécifique qui désignera l'objet dans le script.

- Le Collider **Mesh Collider** est assigné à tous les maillages des objets enfants d'un modèle lorsque l'option **Generate Colliders** est activée.

III. Exemple: Colliders et tag de la porte



III. Exemple: Colliders et tag de la porte



III. Exemple: Colliders et tag de la porte

- Vous devez le remplacer par un collider plus simple et plus efficace en forme de boîte.

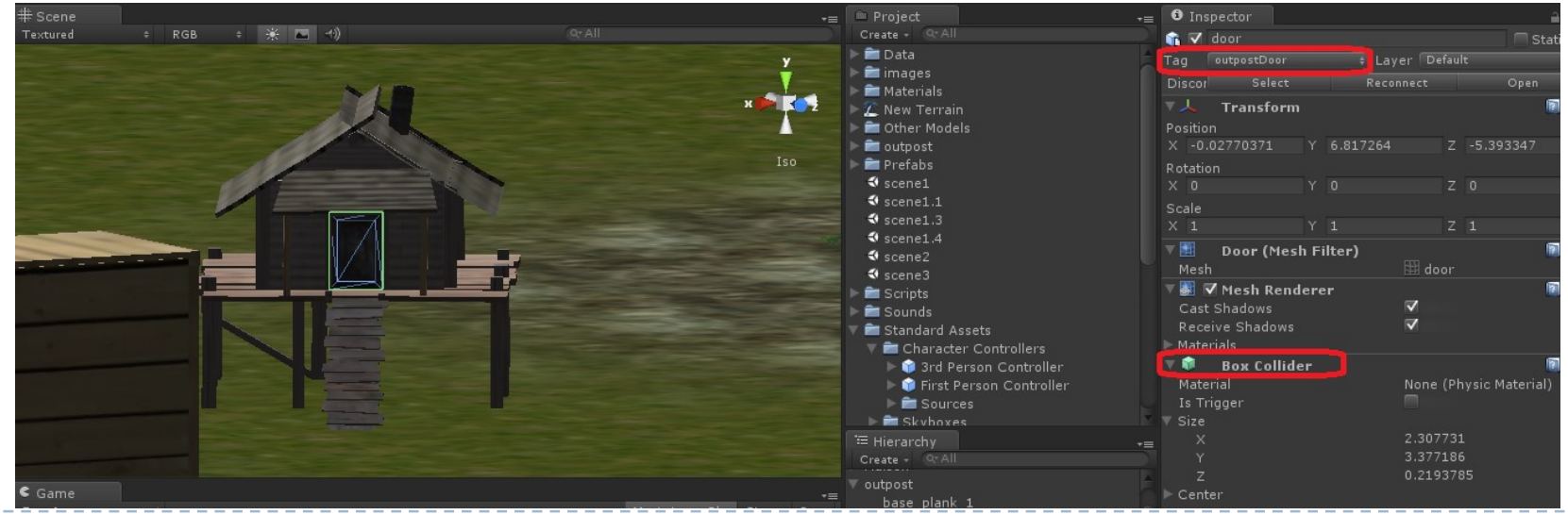
- Cliquez sur **Component > Physics > Box Collider** dans le menu principal.

III. Exemple: Colliders et tag de la porte

- Le composant Box Collider que vous venez d'ajouter apparaît alors autour de la porte, figuré par un contour vert.

III. Exemple: Colliders et tag de la porte

- L'objet enfant door étant toujours sélectionné, cliquez sur le menu déroulant **Tag** situé en haut du panneau Inspector, puis choisissez Add Tag. Ajoutez le tag **outpostDoor** dans le Tag Manager qui s'affiche dans le panneau Inspector.

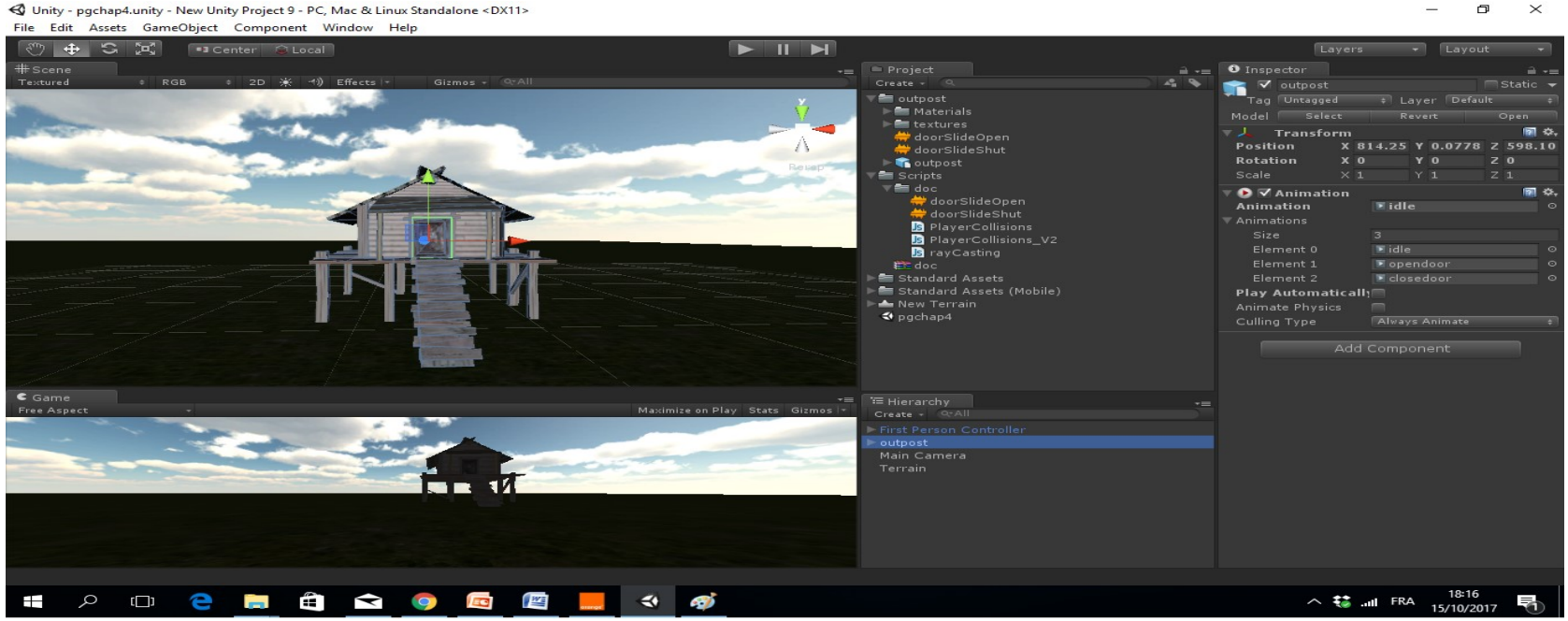


III. Exemple: Désactiver l'animation automatique

- Par défaut, Unity suppose que toutes les animations des objets placés sur la scène doivent être lues automatiquement.
- Si vous laissez Unity lancer automatiquement les animations des objets, ceux-ci risquent de s'afficher dans un des états d'animation plutôt qu'avec leur apparence par défaut.
- Pour corriger ce problème, il suffit de décocher la case **Play automatically** dans le panneau Inspector.

III. Exemple

- L'objet ***outpost*** est maintenant prêt à interagir avec le personnage du joueur. Vous allez donc commencer à écrire les scripts pour gérer les collisions avec la porte, en utilisant soit **la détection de collision** soit le **raycasting**.



III. Exemple

- Nous allons étudier les deux méthodes permettant de déclencher l'animation afin que vous puissiez ensuite utiliser l'une ou l'autre lors du développement de vos jeux en fonction de la situation.

- Dans un premier temps, vous utiliserez la détection de collision – une notion essentielle à connaître lorsque vous commencez à travailler sur des jeux dans Unity.

- Puis dans un second temps, vous implémenterez simplement une projection de rayon depuis le personnage du joueur.

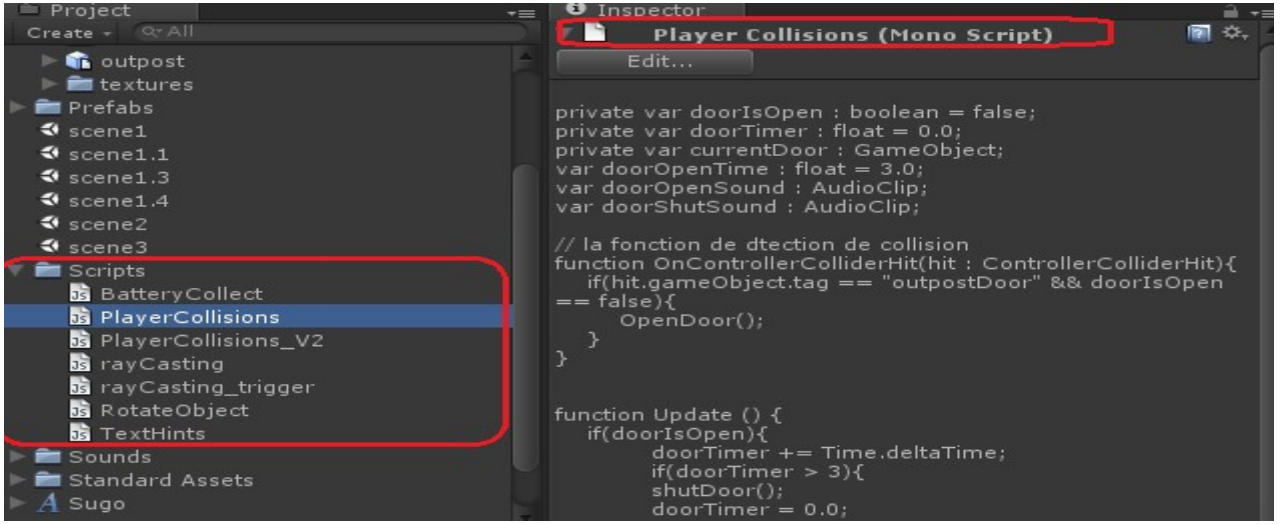
Méthode 1. La détection de collision

- Lors du développement de jeux, il est souvent plus efficace d'écrire un seul script pour un objet qui interagira avec plusieurs autres objets, plutôt que d'écrire de nombreux scripts pour chacun des objets afin de vérifier que ceux-ci entrent en collision avec un objet en particulier.

- Pour un jeu comme celui-ci, vous allez donc écrire un script à appliquer au personnage du joueur et non un script pour chaque objet avec lequel le joueur peut interagir.

Méthode 1. La détection de collision

- Créer un nouveau répertoire dans le projet, nommer ce répertoire (Folder)Scripts
- Sélectionner ce répertoire et créer un nouveau fichier javascript
- Nommer ce fichier en **PlayerCollisions**



Méthode 1. La détection de collision

- ***Utiliser OnControllerColliderHit***

Par défaut, tous les nouveaux scripts JavaScript contiennent la fonction ***Update(),*** qui s'execute lorsque vous les ouvrez pour la première fois. Pour commencer, vous allez déclarer les variables que vous utiliserez.

Pour définir ces variables, ajoutez les lignes suivantes au tout début du script PlayerCollisions que vous modifiez

Méthode 1. La détection de collision

- ***Utiliser OnControllerColliderHit***

```
private var doorIsOpen : boolean = false;
private var doorTimer : float = 0.0;
private var currentDoor : GameObject;

var doorOpenTime : float = 3.0;

var doorOpenSound : AudioClip;
var doorShutSound : AudioClip;
```

Méthode 1. La détection de collision

- ***Utiliser OnControllerColliderHit***

Et ajoutez la fonction suivante:

function ***OnControllerColliderHit***(hit : ControllerColliderHit){
}

- Cette fonction de détection de collision, appelée *OnControllerColliderHit*, est spécialement destinée aux personnages jouables qui utilisent le composant CharacterController, comme c'est le cas ici. Son seul paramètre, ***hit***, est une variable qui stocke des informations lorsqu'une collision se produit, notamment l'objet de jeu avec lequel le joueur est entré en collision.

Méthode 1. La détection de collision

La fonction de collision complète doit maintenant être semblable à celle-ci :

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

La fonction personnalisée OpenDoor() complète doit être analogue à ceci:

```
function OpenDoor(){
            audio.PlayOneShot(doorOpenSound);
            doorIsOpen = true;
            var  myOutpost : GameObject =     GameObject.Find("outpost");
            myOutpost.animation.Play("opendoor");
            }
```

Méthode 1. La détection de collision

```
function Update(){
        if(doorIsOpen){
                doorTimer += Time.deltaTime;
                if(doorTimer > 3){
                shutDoor();
                doorTimer = 0.0;
                }
        }
}
```

Méthode 1. La détection de collision

- function **shutDoor()**{

```
audio.PlayOneShot(doorShutSound);
doorIsOpen = false;
var myOutpost : GameObject =
GameObject.Find("outpost");
myOutpost.animation.Play("closedoor");
```

}

Méthode 1. La détection de collision

- Pour le moment, vous disposez de deux fonctions personnalisées, OpenDoor() et shutDoor(), qui effectuent les trois mêmes tâches : elles lancent la lecture d'un son, définissent une variable booléenne et déclenchent une animation.

- Pourquoi ne pas alors créer une seule fonction qui soit capable de jouer différents sons, de définir si la variable booléenne est true ou false et de lire les différentes animations ?

Méthode 1. La détection de collision

- function **Door**(aClip : AudioClip, openCheck : boolean, animName : String,thisDoor : GameObject){

```
        audio.PlayOneShot(aClip);
        doorIsOpen = openCheck;
    thisDoor.transform.parent.animation.Play(animName);
}
```

- Pour ouvrir la porte, par exemple, vous pourriez appeler cette fonction et définir chaque paramètre de la façon suivante :

```
Door(doorOpenSound, true, "opendoor", currentDoor);
```

Méthode 1. La détection de collision

- **Finaliser le script**

Pour compléter le script, vous allez vous assurer que l'objet qu'il ajoute possède un composant Audio Source, ce qui est nécessaire pour lire les extraits sonores puisque votre fonction Door déclenche du son.

Ajoutez la ligne suivante tout en bas du script, en vous assurant que la ligne NE se termine PAS par un point-virgule.

@script RequireComponent(AudioSource)

Méthode 1. La détection de collision

- **Attacher le script à l'objet:**

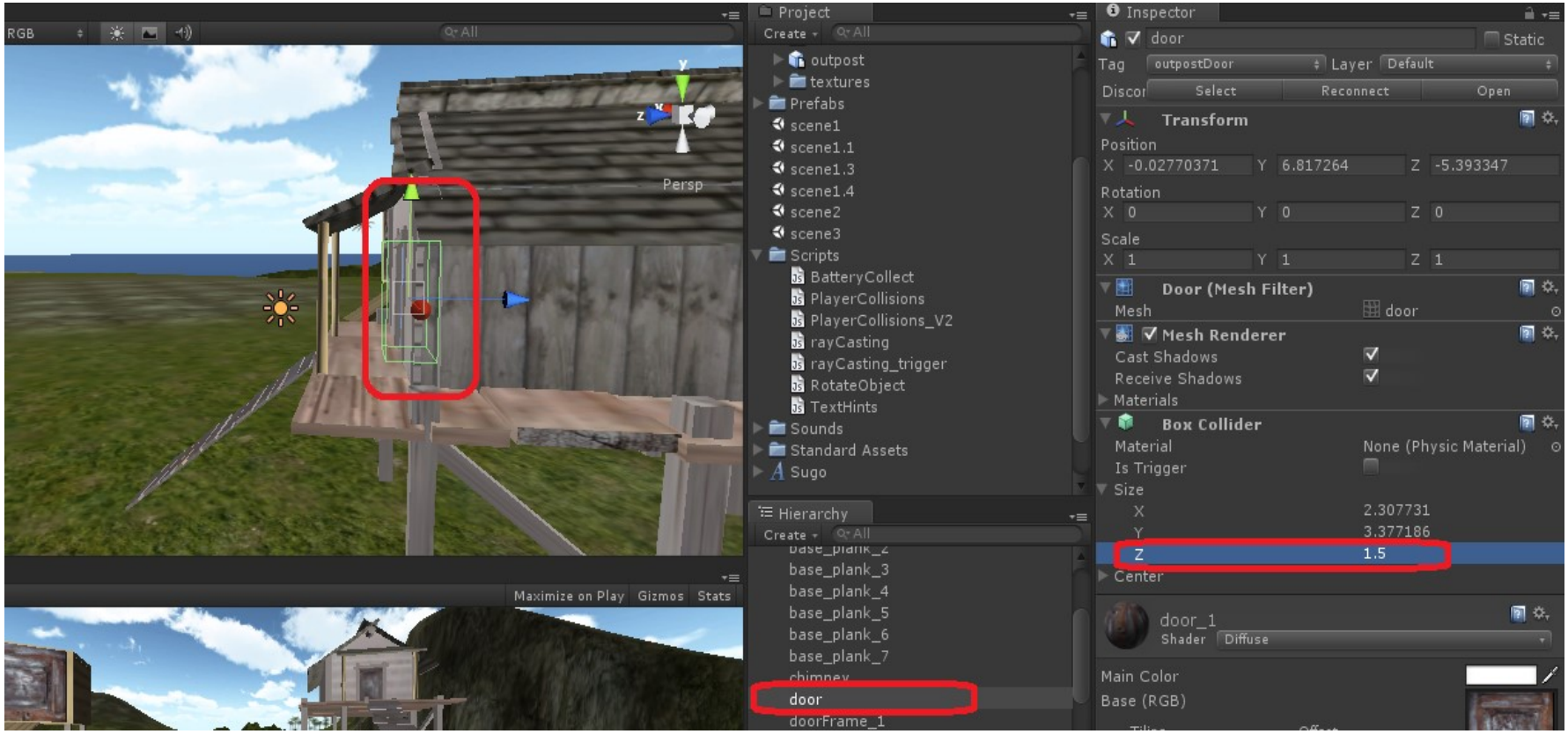
Il existe deux méthodes pour attacher un script à un objet dans Unity :

1-faire glisser le script depuis le panneau Project sur l'objet dans les panneaux Hierarchy ou Scene.

2-sélectionner l'objet puis cliquer sur Component > Scripts > PlayerCollisions dans le menu principal de Unity.

Méthode 1. La détection de collision

Pour entrer en collision avec l'objet Door, nous devons augmenter la valeur Z du Box Collider.



Méthode 2. Le raycasting

- Bien que l'utilisation de la détection de collision ou de déclencheur soit une méthode tout à fait valable, le raycasting vous permet de vous assurer que le joueur ouvre seulement la porte de l'avant-poste s'il lui fait face.

- En effet, le rayon part toujours dans la direction vers laquelle l'objet First Person Controller est tourné, si bien qu'il ne coupera pas la porte si le joueur lui tourne le dos, par exemple.

Méthode 2. Le raycasting

- **Réinitialiser le collider de la porte**

Comme vous voulez éviter l'effet de repoussement qui survient lors du contact avec le collider invisible de la porte, sélectionnez l'objet enfant door du composant Box Collider dans le panneau Inspector, puis donnez au paramètre Size une valeur de 0,1 sur l'axe Z.

- Cela correspond à la profondeur visuelle de la porte.

Méthode 2. Le raycasting

- **Ajouter le rayon**

Le raycasting est placé dans la fonction Update() pour la simple raison que le rayon doit être projeté vers l'avant à chaque image.

Méthode 2. Le raycasting

```
var hit : RaycastHit;
if(Physics.Raycast (transform.position, transform.forward, hit, 5)) {
   if(hit.collider.gameObject.tag== "outpostDoor" && doorIsOpen ==
   false){
   currentDoor = hit.collider.gameObject;
         Door(doorOpenSound, true, "opendoor", currentDoor);}}
         if(doorIsOpen){
         doorTimer += Time.deltaTime;
if(doorTimer > 3){
Door(doorShutSound, false, "closedoor", currentDoor);
         doorTimer = 0.0;
   }     }
```

Méthode 2. Le raycasting

- **Ajouter le rayon**

  var hit : RaycastHit;

  if(Physics.Raycast (transform.position, transform.forward, hit, 5))

  transform.position: Le point d'émission du rayon

  transform.forward: La direction du rayon

  hit: La structure des données

  5: La longueur du rayon