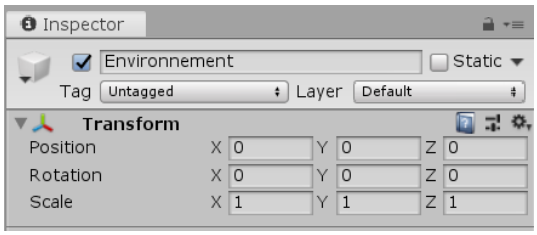
# TP1 Unity

Pour réaliser ce TP, vous pouvez vous aider de la documentation disponible sur le site de Unity :

- Manuel utilisateur : https://docs.unity3d.com/Manual
- Documentation technique de l'API : https://docs.unity3d.com/ScriptReference

## 1. Création du projet

- Lancez Unity et créez un nouveau projet. Choisissez le template « 3D », le nom et l'emplacement du projet.
- Ouvrez le package « Sci-Fi Styled Modular Pack » et importez-le dans le projet.
- Ouvrez la scène « outpost on desert ». Créez un objet vide à la position (0,0,0) nommé « Environment » et déplacez tous les objets de la scène (sauf « Directional Light ») sous celui-ci. Enregistrez la scène sous un nouveau nom.




*Conseil : organisez votre projet en utilisant des sous-dossiers pour chaque type d'objets (Scripts, Scenes, Materials, Textures, etc.).*

- Basculez sur l'onglet « Game », qui permet de voir l'aperçu du jeu. Que voyez-vous ?
- Ajoutez une caméra dans la scène.

## 2. Navigation à la souris et au clavier

Nous allons implémenter la navigation en vue subjective à la première personne, utilisant la souris et le clavier.

**Navigation « caméra »**

Créez un nouveau script appelé « Navigation »

- Ajoutez le composant « Navigation » à la caméra.
- Dans la fonction Update du script Navigation, implémentez la translation selon les 4 directions : avant, arrière, gauche, droite.

Vous aurez besoin des éléments suivants (cf. documentation technique Unity) :

| | |
|---|---|
| `Input.GetKey` | Permet de récupérer l'état d'une touche du clavier. |
| `transform` | Attribut d'un GameObject qui permet d'accéder à son composant « Transform ».<br>Vous pouvez utiliser la méthode Translate du transform. |
| `Time.deltaTime` | Permet de récupérer le temps écoulé depuis la dernière frame. Utiliser ce paramètre pour calculer la distance sur laquelle avancer. |

- Testez en cliquant sur le bouton play.

- Implémentez l'orientation de la caméra. Vous pouvez pour cela utiliser la méthode RotateAround du transform.

| | |
|---|---|
| `Input.GetMouseButton` | `Permet de récupérer l'état d'un bouton de la souris.` |
| `Input.GetAxis("Mouse X")`<br>`Input.GetAxis("Mouse Y")` | `Permet de récupérer le déplacement de la souris selon les deux axes.` |

- Ajoutez deux attributs publics au script « Navigation » : « walkingSpeed » et « rotationSpeed », de type float.
- Sélectionnez la caméra dans l'éditeur Unity : qu'observez-vous ?
- Utilisez ces deux attributs pour régler la vitesse de déplacement et de rotation.

**Navigation « capsule »**

- Ajoutez un objet de type « **Capsule** » à la scène. Cette capsule représentera notre personnage, vous pouvez renommer l'objet « Player ».
- Supprimez le script « Navigation » de la caméra et ajoutez-le à la capsule. Qu'observez-vous ?
- Ajoutez un composant « Rigidbody » à la capsule. Qu'observez-vous ?
- Ajoutez des contraintes pour éviter à la capsule de tomber à la collision avec des objets : utilisez la contrainte « Freeze rotation » sur les axes X et Z dans les options du Rigidbody. Qu'observez-vous ?
- Placez la caméra au niveau de la « tête » de la capsule, et rattachez-la comme enfant de la capsule. Désactivez le composant « MeshRenderer » de la capsule. Qu'observez-vous ?
- Dans le script « Navigation », ajoutez un attribut public de type « Camera ». Appliquez l'orientation haut/bas à cet objet plutôt qu'à la capsule. Faites le lien entre cet attribut et la caméra de la scène. Qu'observez-vous ?

## 3. Ouverture automatique de la porte

Vous allez créer un nouveau script permettant d'ouvrir automatiquement la porte (nommée « door_3 » dans la scène) lorsque le personnage s'en approche.

Ce script (à appliquer à l'objet « Player ») doit réaliser les opérations suivantes :

- Créer un nouveau GameObject enfant de l'objet portant le script.
- Ajouter un **Collider** (de forme sphérique, par exemple) à ce nouvel objet. Le collider doit être de type « trigger ».
- Appeler l'ouverture de la porte lorsqu'elle rentre dans le collider. Pour cela, le script doit récupérer le composant « Animator » de la porte et appeler sur ce composant la méthode suivante :

```
SetBool("character_nearby", true);
```

- De la même façon, lorsque la porte sort du collider, le script doit appeler la méthode suivante du composant Animator :

```
SetBool("character_nearby", false);
```

Méthodes à utiliser :

```
GameObject.AddComponent
GameObject.GetComponent
GameObject.OnTriggerEnter
GameObject.OnTriggerExit
```

# 4. Tir au pistolet laser

- Importez le modèle 3D « laser-gun » dans votre projet ainsi que ses textures.
- Créez un nouveau matériau.
- Appliquez les textures au matériau créé :
  - Albedo : laser-gun_c.tga
  - Normal Map : laser-gun_m.tga
  - Occlusion : laser-gun_ao.tga
  - Emission / Color : laser-gun_E.tga
- Importez le pistolet dans la scène et appliquez-lui le matériau.
- Placez le pistolet comme enfant de l'objet « Player » et à un emplacement approprié (vous pouvez utiliser l'onglet « Game » pour vous aider à le placer correctement).

Créez un modèle de balle pour le pistolet. Pour cela :

- Créez un cylindre appelé « Bullet » de dimension adaptée au pistolet.
- Créez un nouveau matériau (par exemple, rouge avec une couleur d'émission rouge) et appliquez-le au cylindre.
- Ajoutez un Rigidbody au cylindre.
- Faites glisser le cylindre de la scène vers un dossier du projet. L'objet passe alors en bleu : il s'agit d'un **prefab**. Vous pouvez le mettre dans un dossier nommé « Prefabs ».
- Supprimez le cylindre de la scène.
- Faites glisser votre prefab vers la scène : qu'observez-vous ?
- Supprimez le cylindre de la scène.

Créez un nouveau script, à appliquer au pistolet, permettant de tirer. Ce script doit réaliser les opérations suivantes :

- Au clic, instantier le prefab à la position du pistolet.
- Appliquer une force à l'objet instantié.

Méthodes à utiliser :

```
GameObject.Instantiate
Rigidbody.AddForce
```

**Disparition des balles**

Pour que les balles disparaissent au bout d'un certain temps, vous allez utiliser une **coroutine**.

Cf. manuel : https://docs.unity3d.com/Manual/Coroutines.html

- Déclarez une coroutine :
```
private IEnumerator LaunchBullet()
```
- Appelez la coroutine :
```
StartCoroutine (LaunchBullet());
```
- Ajoutez une temporisation, par exemple pour 3 secondes :
```
yield return new WaitForSeconds(3f);
```
- Détruiser la balle après la temporisation en utilisant la méthode `GameObject.Destroy`.

**Mode rafale**

Toujours en utilisant une coroutine, modifiez le script de tir pour tirer en mode rafale : le pistolet enverra des balles à intervalle régulier tant que le bouton de la souris reste enfoncé.

**Mode « rayon laser »**

Modifiez votre script de tir pour le pistolet envoie un rayon laser :

- A la place du prefab, créez un nouvel objet avec un composant de type **LineRenderer**.
- Appliquez le matériau « bullet » précédemment créé.
- Lancez un rayon avec la méthode `Physics.Raycast`.
- Si un objet est touché par le rayon, appliquez-lui une force. Testez sur différents objets du décor. Qu'observez-vous ?

# 5. Build

- Allez dans File -> Build settings.
- Si elle n'y est pas déjà, ajoutez votre scène au build.
- Sélectionnez la plateforme PC Windows.
- Build : sélectionnez un dossier de destination.