Tutoriel 3: les bases d'Unity 3D

1. Animer un objet de la scène

Des scripts en C# ou Javascript permettent de donner un comportement aux objects. Nous préférons les scripts en C# car ce langage est plus proche du Java que vous connaissez probablement mieux (typage, etc.). Nous allons donc créer un nouveau script C# permettant de faire tourner les cubes que nous avons crées :

- Créer un nouveau script (`Assets > Create > C# Script`) que vous nommerez `RotateCube`.
- L'éditer en double cliquant dessus (par défaut, il sera ouvert avec `Mono`).
- La fonction `Update()`, qui est héritée de la classe `MonoBehaviour`, est appelée à chaque pas de temps lors de l'exécution de la scène. Dans cette fonction `Update()`, effectuer une rotation de 3° selon l'axe vertical (Y). Pour cela, vous pourrez accéder à la position de l'objet auquel le script sera associé au travers de l'attribut `transform` (déjà défini dans la classe mère `MonoBehaviour`). Vous pourrez utiliser l'auto-complétion de `Mono` pour voir les fonctions que l'on peut appliquer à cet attribut `transform` et trouver la fonction qui permette d'effectuer la bonne rotation.
- Ajouter le script au `CubePrefab` par glisser-déposer sur l'icône du prefab dans l'explorateur de projet (4). Noter l'apparition du script en tant que composant dans l'inspecteur (3) : on peut le désactiver ou le supprimer. Noter également que toutes les instances de CubePrefab ont été modifiées par cet ajout.
- Exécuter la scène.
- Pour une plus généricité du script, ajouter un attribut public dans la classe `RotateCube` avant la fonction `Start()` : `public float speed = 3.0f;`
- Modifier le script en conséquence.
- Vous remarquerez l'apparition de cet attribut dans l'inspecteur (3), ce qui permet de modifier la valeur de la vitesse de rotation sans éditer le script à chaque fois. Noter que la valeur `3.0f` est la valeur par défaut lorsque l'on ajoute le script à un nouvel objet.
- Tester avec différentes valeurs.

2. Gestion des interactions

Unity 3D permet de capter les entrées de base comme les appuies sur les touches du clavier ou les actions de la souris. Cependant, afin de permettre une plus grand généricité et de ne pas coder en dur toutes les commandes dans les scripts, Unity 3D propose un système de gestionnaire d'`Input`. Pour plus d'information, voir la documentation accessible à l'adresse suivante :
https://docs.unity3d.com/Manual/ConventionalGameInput.html

Nous voulons donc maintenant modifier le script qui fait tourner les cubes pour que l'on puisse activer et désactiver la rotation. Par défaut, la rotation sera désactivée. Lorsque l'utilisateur appuiera sur la touche « r » (et la relâchera ensuite), les cubes se mettront à tourner. Puis lorsqu'il appuiera à nouveau sur la touche « r » (et la relâchera ensuite), les cubes s'arrêteront de tourner, et ainsi de suite : dès que la touche « r » est appuyée, les cubes changeront leur comportement. Pour cela :

- Aller dans le menu (`Edit > Project Setting > Input`), ajouter un nouvel axe en modifiant la taille `Size`. Changer son nom en « `ActivateRotation` » par exemple et définir le `Positive Button` comme la touche « `r` ».
- Pour récupérer l'événement « `ActivateRotation` » dans votre script, vous pourrez utiliser les fonctions suivantes qui retournent des booléens (cf. documentation avec l'auto-complétion de `Mono`) :
  - `Input.GetButton("ActivateRotation")`
  - `Input.GetButtonDown("ActivateRotation")`
  - `Input.GetButtonUp("ActivateRotation")`
- Pour vérifier le bon fonctionnement de votre script, vous pouvez afficher dans la console de l'éditeur (7) un message chaque fois que la touche « r » est appuyée. Pour cela, vous pourrez utiliser la syntaxe suivante : `Debug.Log ("votre message");`