

Objectif :

- Prendre en main l'interface Unity et manipuler ses concepts notamment à travers l'écriture de scripts C#
- Utiliser la documentation officielle
- Réaliser une petite application interactive 3D

Pré-requis

- Bases de la programmation objet. Lecture du cours de présentation Unity.

0. Ressources

Logiciel : <http://unity3d.com/unity/>

Documentation et tutoriels : <http://unity3d.com/learn>

- Manual
- Scripting API
- Tutorials
- Resources

1. Construction du projet : interface & éléments de base

- Créer un nouveau projet (= dossier) (sans charger aucun package prédéfini)
- Dans l'onglet project, créer des dossiers Scripts, Materials, Prefabs et Scenes. Ils serviront à organiser vos assets.
- Enregistrer la scène courante (extension *.unity*) (NB : n'oubliez pas de sauvegarder régulièrement la scène !)
- Insérer un plan dans la scène
 - o GameObject -> 3D Object -> Plane
 - o Observer ses composants dans l'Inspector
 - o Modifier sa position pour qu'il soit visible par la Main Camera (par les outils du menu et par les champs du composant Transform)
- Insérer un éclairage de type quelconque
 - o GameObject -> Light -> xxx light
 - o Modifier sa position pour qu'il éclaire le plan
- Insérer et placer un cube
 - o GameObject -> 3D Object -> Cube
- Créer un nouveau Material *Red*
 - o Assets (ou clic droit dans le project) -> Create -> Material
 - o Choisir la couleur rouge
 - o Appliquer ce material au cube par glisser-déposer
- Utiliser une image pour texturer le plan
 - o Enregistrer une image quelconque sur le PC
 - o L'importer dans le projet : Assets -> Import new asset -> ... ou par glisser-déposer
 - o L'ajouter au plan (notez la création automatique d'un material correspondant)

- Ajouter de la physique à la scène
 - o Sélectionner le cube
 - o Ajouter le Component -> Physics -> Rigidbody
 - o Jouer la scène.
 - o Lire la documentation de Rigidbody et BoxCollider. Modifier différents paramètres et tester
- Glisser-déposer le cube depuis l'onglet Hierarchy vers l'onglet Project : cela va créer un « Prefab » à partir du cube (cf. icône bleue)
 - o le renommer en CubePrefab
 - o Glisser-déposer le prefab depuis l'onglet Project vers l'onglet Hierarchy ou la scène pour insérer 2 autres instances de cubes

2. Premier script pour comprendre la Game Loop

- Créer un nouveau script C# appelé Logs
 - o Assets -> Create -> C# Script -> entrer le nom souhaité
 - o L'éditer par double clic (par défaut avec Visual Studio)
 - o Dans la méthode Start(), faire afficher un message de votre choix dans la fenêtre de log, avec la commande [Debug.Log\(\)](#)
 - o Dans la classe Logs, créer un attribut int compteur initialisé à 0
 - o Dans la méthode Update(), faire afficher ce compteur dans la console puis l'incrémenter
 - o Sauvegarder
- De retour dans Unity (cela met un peu de temps), ajouter ce script à la MainCamera et tester
 - o notez l'apparition en tant que composant dans l'inspecteur ; on peut le désactiver ou le supprimer de l'objet (**NB : Attention à ne pas l'ajouter plusieurs fois !**)
 - o notez ce qui se passe dans la console, et les boutons à votre disposition

3. Programmation d'une première animation : rotation des cubes

- Créer un nouveau script Rotate
 - o Dans la méthode Update(), faire une rotation de 5° selon l'axe y (cf. cours et classe Transform) et sauvegarder
- Ajouter ce script au prefab CubePrefab par glisser-déposer dans l'onglet Project
 - o notez que toutes les instances de CubePrefab sont modifiées par cet ajout
- Tester
- Pour plus de généricité du script
 - o Ajouter un attribut avant le Start() : public float speed = 5.0f;
 - o Modifier l'instruction de rotation pour prendre en compte speed
 - o Vous remarquerez l'apparition de cet attribut dans l'inspecteur pour pouvoir modifier sa valeur. Tester avec différentes valeurs.

4. Programmation d'interactions clavier simples

On souhaite contrôler l'activation de la rotation précédente :

- Rotation tant que la touche R est appuyée
 - o créer un nouveau script RotateDuringKeyPressed
 - o reprendre le contenu du script Rotate en ajoutant un test sur la touche R (cf. documentation Input)
 - o remplacer le script dans CubePrefab et tester (rq : désactiver la gravité)
- Généralisation

- Modifier le script précédent pour pouvoir choisir la touche de rotation dans l'inspector (type string ou KeyCode)
- Paramétrer 3 touches différentes pour les 3 instances de cube et tester
- Rotation démarrée ou arrêtée par 1 seul appui sur 1 touche
 - créer un script RotateOnOffKey
 - quelle méthode de Input et avec quoi l'utiliser pour arrêter/reprendre la rotation ?
 - ajouter le script dans CubePrefab, désactiver le script précédent, paramétrer les touches dans les instances et tester

5. Programmation d'interactions souris simples

- On souhaite changer la couleur du 1^{er} cube lorsque la souris passe dessus :
 - Créer un script ChangeColorOnMouse
 - Pour la détection de l'interaction souris, utiliser des événements spéciaux des scripts (cf. MonoBehaviour)
 - Pour la modification de la couleur, 2 possibilités (cf. Renderer et Material) :
 - Modifier la couleur du matériau pour l'objet en cours
 - Ou changer directement le matériau utilisé
 - NB : La nouvelle couleur peut être créée/choisie simplement dans le script, mais pas le nouveau matériau qui doit être en attribut public et donné dans l'inspector
 - Pour retrouver la couleur ou le matériau original :
 - Où et comment les sauvegarder ?
 - Quel événement de script utiliser ?
 - Tester

6. Un peu de transformations spatiales

- On souhaite naviguer en faisant orbiter la caméra autour et vers le centre de la scène avec les touches du clavier
 - Créer un script OrbitalCamera
 - Récupérer les valeurs des axes correspondants aux flèches verticales et horizontales
 - Appliquer ces valeurs dans les 2 rotations voulues (attention aux axes et repères de rotation)
 - Effectuer un changement de zoom avec + et -
- On souhaite déplacer les cubes sur le plan horizontal selon les mouvements de la souris¹
 - Créer un script ManipulationMouse
 - Dans Update
 - Tester l'appui sur le bouton gauche avec une fonction de Input². Confirmer le test par un message dans la fenêtre de log
 - Stocker le déplacement souris entre chaque frame, toujours en utilisant une fonction de la classe Input
 - Effectuer la translation correspondante à ce déplacement souris
 - Ajouter au CubePrefab et tester
 - Proposez une solution pour que les cubes se déplacent uniquement si on a cliqué sur eux

¹ Pour simplifier on considèrera que OrbitalCamera n'est pas actif lors de la manipulation

² Bouton gauche = 0 ou "Fire1" selon la fonction utilisée ; les détails des Inputs sont définis dans le projet : Edit -> Project settings -> Input Manager

7. Création dynamique d'objets

- Créer un nouveau Script CreateOnClick
- Ajouter ce script à la caméra
- Coder le script :
 - o Créer 2 attributs pour référencer un modèle d'objet à copier et la copie
 - o Tester si le bouton droit de la souris ("Fire2") a été cliqué
 - o Instancier la copie à partir du modèle (avec Instantiate()) à la position/rotation courante (ie. celle donnée par le transform du propriétaire du script)
 - o Changer la couleur de la copie en bleu
 - o La positionner à quelques unités devant la position courante de la caméra
- Créer un nouveau prefab d'objet (ex. sphère) et tester

8. Interface Utilisateur (UI)

- On veut ajouter un texte explicatif sur l'utilisation de l'application
 - o Insérer un GameObject -> UI -> Text
 - o Noter la création d'un Canvas contenant cet objet texte et d'un EventSystem
 - o Remplissez le contenu du texte pour donner les touches à utiliser dans votre application
 - o Paramétrer le composant Rect Transform pour que cette UI soit toujours en bas à gauche de l'écran
- Pour quitter l'application
 - o Créer un script Quit dans la caméra
 - o Appeler Application.Quit() lors de l'appui sur la touche échap
 - o *Rq* : ne fonctionne pas dans l'éditeur, seulement dans le build

9. Build

Générer votre application pour Windows dans un nouveau dossier

- Contenu de la scène :
 - o 3 cubes avec ChangeColorOnMouse et ManipulationMouse
 - o 1 tourne avec Rotate, 1 avec RotateDuringKeyPressed (touche R) et 1 avec RotateOnOffKey (touche T)
 - o Caméra avec OrbitalCamera, CreateOnClick et Quit
 - o GUI explicatif
 - o Sans physique
- Paramétrer les BuildSettings -> PlayerSettings pour une exécution en mode fenêtré (résolution 1680x1050 par défaut)
- Observez les fichiers/dossiers générés (variables selon la version de Unity)
- **Tester** l'exécutable
- **Zipper l'ensemble du dossier généré**

10. Enrichissement de la scène

Explorer les Assets, les GameObjects, les Components et la documentation pour ajouter des propriétés ou des animations à votre scène (eg. particules, sons...).

Annexes : quelques conseils d'utilisation et raccourcis

Sauvegarder (control + s) très régulièrement le projet et la scène
Les changements de la vue dans la scène vous permettent de mieux visualiser les 3 dimensions et les positions relatives de vos objets et caméras : <ul style="list-style-type: none">- translations = clic milieu- rotations = clic droit ou alt + clic gauche- zoom = molette- zoom auto sur objet sélectionné dans la hiérarchie = Shift + F
Lancer et <u>arrêter</u> le jeu avec le bouton Play Eviter la mise en pause : toutes les modifications que vous faites dans l'inspecteur pendant la pause seront perdues lors de l'arrêt (qui revient aux conditions initiales)
Si vos objets ne sont plus visibles : revenir à des positions/orientations canoniques via l'inspecteur (par ex. (0,0,0)) pour ces objets ET pour la caméra
Privilégiez une disposition des vues qui ressemble à celle du cours : onglets scène et game visibles en même temps, console visible... Si vous perdez/fermez des onglets ou que la disposition ne vous convient pas : Menu Window ou Window -> Layout Dans le pire des cas, sauvegardez et fermez l'onglet concerné par le problème, voire fermez le projet. Au démarrage d'un projet aucune scène n'est chargée (donc hiérarchie et scène vide), il faut la lancer manuellement
Nommez de manière explicite vos objets, prefab, scripts, matériaux et rangez les dans des dossiers également bien nommés
Supprimer un composant entraîne la perte des paramètres si vous les avez modifiés. Préférer une désactivation (case à cocher) pour les tests si c'est possible