UNITY - TP2

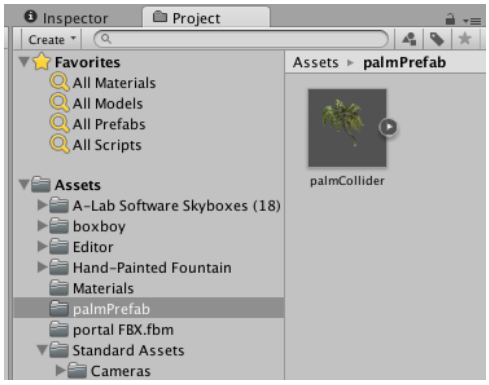
Dans ce TP vous allez gérer les collisions entre les objets, utiliser le graphe de scène, créer et modifier des prefabs[1], créer une nouvelle scène avec une modélisation simple, utiliser les matériaux et puis réaliser l'interface graphique. Vous verrez ensuite comment lier vos deux scènes. **N'oubliez pas de faire de captures d'écran de votre avancement pour votre compte rendu.**

LES COLLIDERS

La détection de collisions est un élément essentiel d'un environnement virtuel réaliste. Pour éviter à notre avatar de traverser les arbres, nous allons utiliser les *colliders*. Un **Collider** contient la géométrie utilisée par la simulation physique d'Unity pour calculer les collisions entre les objets.

Pour les arbres, vous allez créer un nouvel objet à partir de l'arbre de base et lui rajouter un Collider qui sera placé autour du tronc uniquement. Au travail !
Modifier votre prefab FPSController par le prefab RigidBodyFPSController.
Dans la fenêtre Project, sélectionnez l'arbre (par exemple PalmDesktop) dans ***Assets > Standard Assets > Environment > SpeedTree > Palm*** et glissez-le vers la scène (près de votre contrôleur pour vous faciliter la tâche). Ajoutez à l'arbre sélectionné un collider de type capsule : ***Add Component > Physics > Capsule Collider***. Testez le jeu et essayez de traverser l'arbre, ce n'est plus possible ! Vous pouvez modifier le collider de manière à entourer uniquement le tronc de l'arbre en modifiant le rayon de la capsule.

Maintenant, comment ajouter des colliders à tous les arbres de la scène ? Dans la fenêtre Assets créez un nouveau dossier « Prefabs » . Créez dans ce dossier un nouveau Prefab, en faisant clic droit et puis ***Create > Prefab***. Glissez ensuite l'arbre de la scène dans le Prefab et effacez-le de l'hiérarchie. Glissez-le à nouveau dans la scène mais cette fois à partir du prefab.



Pour modifier les autres arbres, sélectionnez le terrain et cliquez ensuite sur l'outil « Place Trees » dans l'Inspector. Cliquez sur l'arbre « Palm » et puis sur « **Edit Trees > Edit Tree** ». Dans la fenêtre, glissez votre prefab ou cliquez sur le cercle à coté de l'arbre et sélectionnez ensuite l'arbre prefab que vous avez créé. Validez, cliquez sur le bouton Refresh et vérifiez dans l'Inspector que la case « Enable Tree Collider » est cochée. Si vous avez d'autres types d'arbre, il faudra refaire la procédure pour chaque type.

Testez votre jeu !

Copiez dans votre fenêtre Projet les dossiers textures et sounds à partir des ressources du TP. Glissez le dossier BoxBoy dans votre dossier Prefabs. Pour compléter les ressources, téléchargez quelques matériaux gratuits à partir de l'asset store, par exemple

[1] Un prefab est un objet prédéfini, réutilisable

https://assetstore.unity.com/packages/2d/textures-materials/abstract/early-prototyping-material-kit-51761 ou https://assetstore.unity.com/publishers/4986

LES PIECES D'OR

L'objectif de cette partie est de placer des objets à collecter pour le joueur, d'utiliser des sons et de réaliser l'interface utilisateur. Les objets à collecter seront des pièces d'or, mais vous pouvez utiliser un autre objet de votre choix.

Pour la géométrie des pièces, créez un cylindre et modifiez-le pour qu'il ait la forme d'une pièce. Puis, collez la texture « or » de votre préférence sur le cylindre.bCréez un Prefab « Piece » dans votre dossier Prefabs et glissez le cylindre vers lui. Ajoutez à votre objet un composant ***Component > Audio > Audio Source***. Glissez le fichier coinSound.wav vers le champ AudioClip. Enlevez l'option Play on Awake pour éviter que le son s'active lorsque l'objet est créé.

Cliquez sur la case isTrigger du collider de l'objet Piece pour l'activer et générer les événements de collision. Nous allons créer un *script* pour les pièces. Il nous permettra de « désactiver » la pièce une fois qu'elle aura été trouvée. Pour les scripts on utilisera el langage C#. Pour configurer l'éditeur allez dans le menu ***Edit > Preferences > External Tools***. Dans la liste déroulante External Script Editor choisissez Visual Studio Community.

Dans le volet Projet, créez un nouveau dossier Scripts. Faites clic droit sur le dossier et sélectionnez Create > C# Script. Un nouveau fichier apparaît, modifiez le nom à CoinsBehaviour. Faites clic droit sur le script pour ouvrir l'éditeur. Le script hérite de la classe MonoBehaviour, qui contient des méthodes comme como Start o Update. (Au cas où, les deux premières lignes font référence aux libraires contenant des méthodes nécessaires au développement).

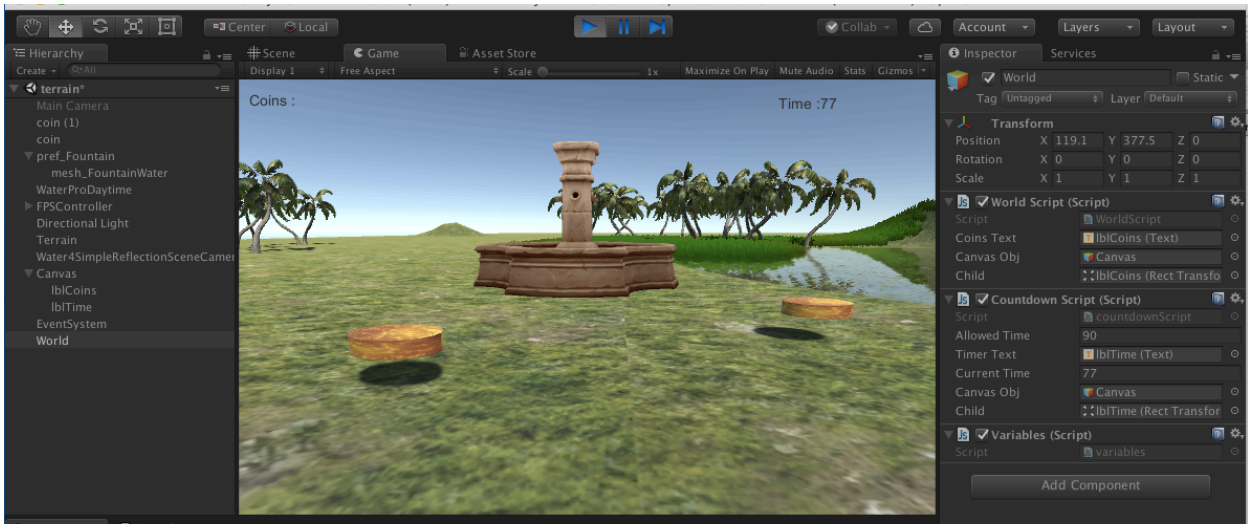
```
using UnityEngine;
using System.Collections;
```

Ensuite vous trouverez la déclaration de la classe coinsBehaviour. Pour pouvez avoir des variables et constantes privées (private const) et des variables publiques (exposées) qui seront visibles par les autres classes et, très important, dans l'Inspector. Cela va nous permettre de configurer le script comme n'importe quel autre composant d'Unity.

```
public class CoinsBehaviour  : MonoBehaviour {

AudioSource aud;

void Start() {
        aud = gameObject.GetComponent.<AudioSource>();
}

void OnTriggerEnter( Collider other ) {
        Renderer objRenderer ;
        Collider objCollider ;
        objRenderer = GetComponent<Renderer>() ;
        objCollider = GetComponent<Collider>() ;
        objRenderer.enable = false;
        objCollider.enable = false;

        if ( aud ) {
                aud.Play();
        }
}
```

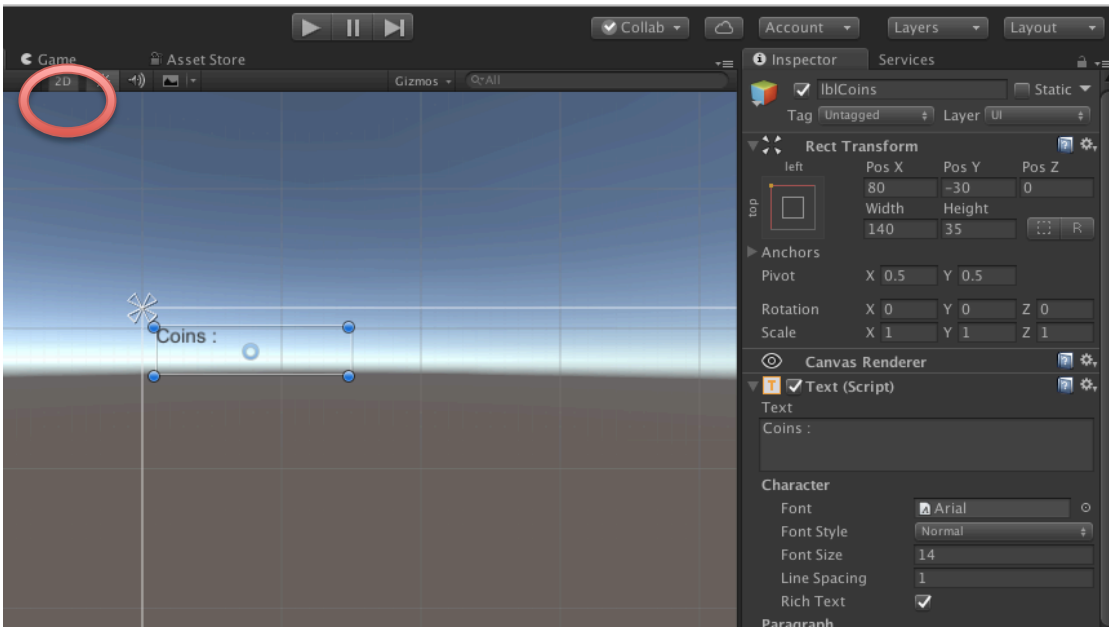
La fonction Start() est exécutée de manière automatique lorsque la simulation commence. La fonction Update(), quant à elle, est exécutée une fois par frame (à chaque fois donc qu'une nouvelle image est générée). La fonction OnTriggerEnter() est appelé lorsque le Collider other entre dans le trigger. Grâce à ce script, la pièce va détecter la proximité du joueur et puis va lancer l'audio clip et disparaitre.

Sauvegardez et ajoutez votre script à l'objet Piece : dans le volet Inspector, faites clic sur Add Component. Puis sélectionnez votre script (vous pouvez vous aider du champ recherche). Ajoutez maintenant plusieurs pièces à votre scène.



L'INTERFACE

Maintenant nous allons compter les pièces collectées par le joueur et afficher cette information sur l'écran.
Pour créer l'interface, il est nécessaire d'ajouter un canvas à la scène : menu ***GameObject > UI***. Créez le canvas, il apparaitra comme un long rectangle à coté de votre scène. Créez un objet Texte ***GameObject > UI > Text*** et placez-le en haut à gauche de votre canvas. Nommez-le lblCoins.



Créez ensuite deux éléments UI Text. Ils se placent sur le canvas. Vous pouvez les déplacer (en cliquant-glissant sur leur milieu) ou changer leurs dimensions grâce aux poignées. Pour travailler plus facilement avec ces éléments, vous pouvez cliquer sur le bouton **2D**.

Nous allons créer maintenant l'objet « World ». Créez un nouveau GameObject vide et appelez-le World. Puis, créez un script World avec le code suivant :

```
(…)
using UnityEngine.UI ;

public class WorldBehaviour  : MonoBehaviour {
        Text coinsText ;
        int coins = 0;
        GameObject canvasObj;
        Transform child ;

        void Start() {
           canvasObj = GameObject.Find("Canvas");
           child =  canvasObj.transform.Find("lblCoins");  //le nom de votre objet UI Text
           coinsText = child.GetComponent<Text>();
        }

        public void AddCoin()
         coins++;
          coinsText.text = "Coins: " + coins;
        }
```

Liez le script à l'objet World. Ajoutez à votre script CoinBehaviour les lignes suivantes :

```
public GameObject worldObject;
Au début de la fonction Start :
        worldObject = GameObject.Find("World");
Au début de la fonction OnTriggerEnter :
        worldObject.SendMessage("AddCoin");
```

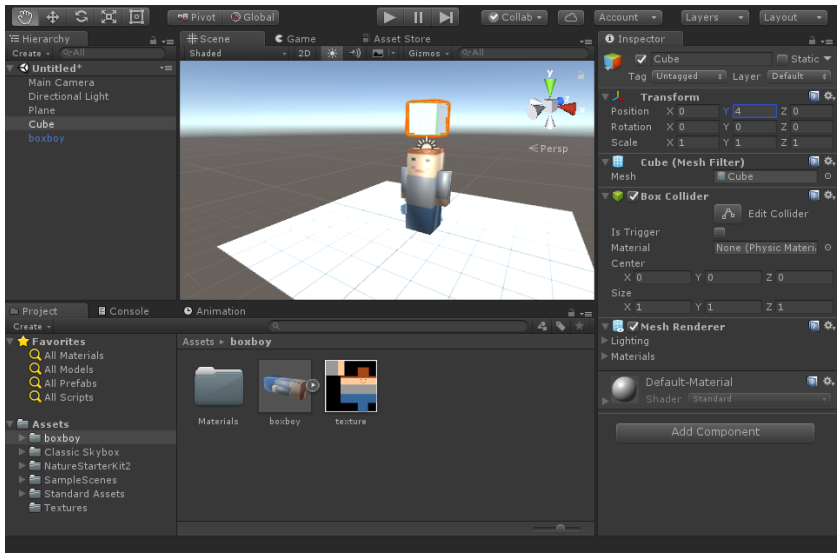
Testez votre jeu !

Utilisez le menu ***File > Build Settings*** pour ajouter la scène au jeu (Add Open Scenes).

MEET BOXBOY

Créez une nouvelle scène dans votre projet et nommez-la appelée BoxBoy. Ajoutez maintenant un plan 2D pour créer un sol ***Game Object > 3D Object > Plane***. Cela vous permettra de créer un plan 2D. Modifier l'échelle par un facteur 3. Placez le plan à l'origine (0,0,0). Créez un second plan et appliquez une transformation en X de -90° pour le positionner à la verticale.

Ajoutez le prefab BoxBoy à votre scène. Faites deux clics pour le rendre visible. Ajustez dans l'Inspector la taille du boybox (j'ai appliqué un facteur de 50 dans chaque axe). Il est toujours possible de réinitialiser les transformations d'un objet en cliquant sur la roue crantée à droite du Transform dans l'Inspector.
Créez un cube ***Game Object > 3D Object > Cube*** et modifiez sa position pour le placer sur BoxBoy N'hésitez pas à comparer les transformations et à changer de point de vue pour mieux localiser les objets les uns par rapport aux autres.



LES MATERIAUX

Il est possible de donner un aspect particulier aux objets grâce aux matériaux et aux textures. Vous pouvez choisir un matériau parmi ceux que vous avez téléchargé ou définir un nouveau. Les Materials utilisent des *shaders* pour définir les propriétés du rendu de l'objet. Dans le Material vous pouvez ajuster ces propriétés et assigner les assets (textures, couleurs, cubemap,…).
Pour créer un nouveau matériau, faites clic droit sur l'onglet Assets, puis choisissez ***Create > Material***. Utilisez le shader Standard et choisissez la couleur de l'objet dans l'attribut **Albedo**. Modifiez la couleur du cube en glissant le matériau sur l'objet. N'hésitez pas à tester les différents shaders et à jouer avec les couleurs et les différents paramètres.

Appliquer également un matériau sur le plan du sol. Pour le plan « mur » perpendiculaire au sol, glisser la texture pixelart du dossier textures directement sur l'objet. Un nouveau matériau sera créé à partir de la texture. Vous pouvez bien sur contrôler le processus en créant un nouveau matériau, en glissant la texture vers celui-ci (ou dans le cadre à gauche de l'albedo) et puis en glissant le matériau vers l'objet.

Créez un nouveau script MouseBehaviour et modifiez sa fonction update :

```
public void Update()  {
        transform.Translate(Input.GetAxis("Horizontal"),Input.GetAxis("Vertical"),0) ;
}
```

Associez le à BoxBoy. Testez la scène et déplacer l'objet avec les flèches du clavier. Pour modifier la position d'un objet, il est nécessaire de changer les valeurs de sa « transform ». La transform est l'entité (matrice de transformation) qui stocke la position, rotation et échelle d'un objet. La fonction Translate de l'entité transform permet de modifier la position et reçoit 3 paramètres, correspondant au déplacement dans chaque axe.

Input est une classe et la fonction GetAxis retourne une valeur entre -1 et 1. Dans le cas de l'axe horizontal, la touche flèche gauche est indiquée par -1 et la droite par 1. Les axes sont prédéfinis dans la configuration d'entrée. Si vous souhaitez modifier les noms et les raccourcis clavier, allez dans ***Edit > Project Settings > Input***.

Il est possible de déplacer les objets à une vitesse plus intuitive en l'exprimant en mètres par seconde. Pour ce faire, il faut multiplier la valeur retournée par Input.GetAxis() par Time.DeltaTime et par la vitesse de mouvement que nous souhaitons :

```
public void Update () {
        float speed = 5.0F;   // déplacer l'objet 5 m par seconde
        float x = Input.GetAxis("Horizontal") * Time.deltaTime * speed;
        float y = Input.GetAxis("Vertical") * Time.deltaTime * speed;
        transform.Translate(x, y, 0);
}
```

Maintenant que le script est prêt, faites un glisser-déposer du script dans le BoxBoy de votre scène. Exécutez votre jeu en cliquant sur le bouton Play, vous devez pouvoir contrôler le personnage grâce aux flèches.

Important : n'oubliez pas que les changements que vous faites pendant que le jeu tourne seront perdus lorsque le jeu s'arrêtera. Vérifiez donc toujours que vous n'êtes pas en mode aperçu du jeu lorsque vous réalisez des modifications. Si vous activez l'option « Maximize on play » de l'onglet Game, cela vous arrivera peu ☺

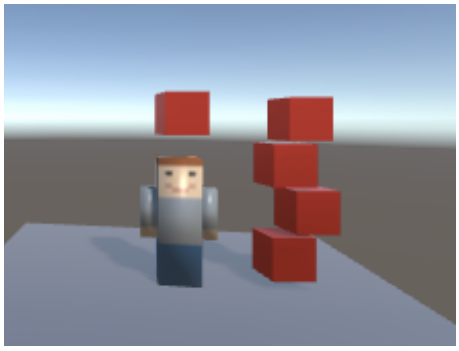
LA PHYSIQUE

Vous avez surement noté que le cube ne se comporte pas en suivant les lois de la physique. Pour qu'il le fasse, il est nécessaire de lui ajouter un composant RigidBody.

Sélectionnez le cube et puis, dans l'Inspector, cliquez sur addComponent. Sélectionnez ensuite ***Physics > Rigidbody***. Le composant doit être visible dans la vue Inspector. Déplacez le cube pour qu'il se trouve sur la tête du bonhomme. Exécutez le jeu et observez le comportement du cube.

DUPLICATION

Lorsque l'on duplique un objet, toutes ses caractéristiques sont aussi dupliquées. Sélectionnez le cube et appuyez sur Ctrl+D (disponible aussi dans le menu Edition ou dans le menu contextuel de l'objet dans la vue Hiérarchie). Créez plusieurs cubes et déplacez-les en les empilant les uns sur les autres. Exécutez le jeu, vous verrez les cubes interagir en suivant les lois de la physique.



LE GRAPHE DE SCENE

La vue Hiérarchie permet de définir les relations parents-enfants entre les objets de la scène, c'est-à-dire le graphe de scène. Prenez un cube et placez-le dans la main de BoxBoy. Testez le jeu, le cube ne suit pas BoxBoy, il reste à sa place. Maintenant, dans la vue Hiérarchie, glissez le cube vers BoxBoy. Il est maintenant son « enfant » et va donc subir toutes les transformations du parent. Testez votre jeu, le cube reste dans la main !!

STARGATE

Vous allez maintenant faire un lien entre les deux scènes du projet. Ouvrez votre scène terrain et créez pour votre objet Fontaine un script qui charge la scène BoxBoy lorsque le controller s'approche suffisamment de la fontaine. Activez pour la fontaine l'option IsTrigger.

```
(…)
using UnityEngine.SceneManagement ;

string levelToLoad= "BoxBoy" ;
void OnTriggerEnter(Collider other) {
        SceneManager.LoadScene(levelToLoad) ;
}
```

Utilisez le menu ***File > Build Settings*** pour ajouter la scène au jeu (Add Open Scenes). **Testez votre jeu !**
Maintenant lorsque vous vous approchez de la fontaine vous êtes télé-transporté auprès de BoxBoy !

Voilà ! ☺ Déposez votre compte rendu de TP (captures d'écran, vos impressions/astuces/difficultés, lien vers vos assets) sur Moodle.

À TRÈS BIENTÔT!