Gestions des scènes

Dans Unity, il est possible et **souhaitable** de travailler en utilisant plusieurs scènes. Un exemple simple pourrait être l'utilisation d'une scène d'introduction, d'une scène menu, d'une scène jeu et finalement d'une scène de conclusion.

L'utilisation des scènes permet une scénarisation plus efficace et moins complexe en plus d'assurer une bonne gestion de la mémoire de l'ordinateur. En effet, lorsqu'on quitte une scène pour se diriger dans une nouvelle, la première scène est purgée de la mémoire. Notez cependant que les variables statiques sont conservées en mémoire lors des changements de scène. Par ailleurs, il est possible de conserver un *GameObject* lors d'un changement de scène en utilisant la commande ***DontDestroyOnLoad()***. (voir plus bas)

Chaque scène possède sa propre structure et ses propres *GameObjects*.

Pour créer une nouvelle scène : menu ***File →New Scene (ou File →Save Scene as)***

Avant de pouvoir faire des changements de scènes, il faut s'assurer que toutes les scènes nécessaires ont été ajoutées dans le projet : ***menu File à Build Settings…***

Changement de scène

SceneManager.LoadScene("nom de la scène") ou SceneManager.LoadScene(index de la scène)

Pour utiliser la classe *SceneManager*, il est obligatoire de l'importer dans le haut du script (juste après le
using UnityEngine.SceneManagement;

Pour que les scènes puissent être chargées, elles doivent avoir été ajoutées dans la fenêtre "*scenes in build*" accessible dans le menu : ***File-->Build Settings***

L'index d'une scène (son numéro) correspond à l'ordre dans lequel la scène a été placée dans la fenêtre "*build settings".*

Pour obtenir le nom de la scène active, on peut utiliser :
SceneManager.GetActiveScene().name

Pour voir les autres propriétés et méthode voir:
https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html

Ceci est un Exemple

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; // Nécessaire pour gestion de scènes

public class ChangementScene : MonoBehaviour {
      void  Start ()
      {
          if(  SceneManager.GetActiveScene().name != "SceneJeu")
          {
                SceneManager.LoadScene("SceneJeu");  // Chargement de la scène "jeu"
          }
          else
          {
                SceneManager.LoadScene(0);        // Chargement de la scène 0
          }
   }
}
```

Préserver un objet d'une scène à une autre

DontDestroyOnLoad(Object);

Commande indiquant à Unity de ne pas supprimer l'objet lors d'un changement de scène. L'objet peut être un objet vide ou un GameObject. Notez que toute la hiérarchie de l'objet sera également préservée, incluant ses composants (Collider, AudioSource etc.)

```
// Lors du lancement, on indique à Unity que le GameObject sur lequel se trouve ce script doit
être conservé lors des changements de scène.

void Start () {
   DontDestroyOnLoad(gameObject);    //ne pas détruire l'objet du script actuel
}

ou
//Pour ne pas détruire un objet spécifique
public GameObject  objetANePasDetruir;
void Start () {
   DontDestroyOnLoad(objetANePasDetruir);     //ne pas détruire l'objet indiqué
```

```
}
ou
Pour  empêcher de refaire le DontDestroy si  on revient dans la scène de départ, il faut utiliser une variable static pour mémoriser l'état, sinon il y aura deux fois le même objet dans la scène initiale.

public static bool dontDestroyDejaFait = false;
void Start () {
  if( dontDestroyDejaFait == false)
  {
      DontDestroyOnLoad(objetANePasDetruir);
       dontDestroyDejaFait = true;
  }
  else
  {
     Destroy(objetANePasDetruir);
  }

}
```

Consignes pour l'exercice (suite du projet "*SurvieShooter*")

N'oubliez pas d'ajouter les scènes dans le ***File/BuildSettings*** pour pouvoir changer de scène.

Scène d'introduction

Créez une scène d'introduction semblable à celle du démo. L'image de fond et la police de caractères utilisées pour le texte sont disponibles sur le site du cours. L'image de fond (UI de type image) doit recouvrir l'ensemble de la scène.

Créez un objet vide (exemple: "GestionScene") et lui associer un script pour les changement de scènes.
Lorsqu'on appuie sur la barre d'espacement, la scène principale doit être chargée. Si on est dans la scène principale alors il ne faut pas permettre de recharger la scène. Vérifiez le nom de la scène sur laquelle on se retrouve en utilisant :

SceneManager.GetActiveScene().name

Musique

Vous pouvez associer la musique à l' objet vide ("GestionScene") ou créer un objet musique qui est l'enfant de cette objet.

La musique du jeu doit commencer dès la scène d'introduction et continuer dans la scène principale et dans la scène finale. La lecture de la musique ne doit pas être interrompue lors du changement de scène. Pour y arriver, vous aurez

besoin de la commande ***DontDestroyOnLoad*** pour que le *gameObject* qui joue la musique soit préservé d'une scène à l'autre.

Dans le Start() du script de l'objet GestionScene, appliquer le ***DontDestroyOnLoad*** (gameObject);, ainsi la musique et le script de changement de scène seront présents dans chaque scène.

Pointage

Dans la scène principale (le jeu!), il faut maintenant compter et afficher les points. Chaque monstre tué doit faire augmenter le pointage (Hellephant = 20 points, ZomBunny = 10 points et ZomBear = 5 points).

Le pointage doit être conservé dans **une variable statique** puisque vous en aurez besoin dans la scène finale. Vous pouvez déclarer la variable dans le script du joueur. Il faut la déclarer une seule fois.

L'augmentation du pointage peut être fait dans la fonction Touche() des ennemis.

Astuce: Si vous déclarez une variable dans le script Ennemi qui mémorise le nombre de points accordés par l'ennemi et lui donner une valeur dans l'inspecteur alors vous pouvez y accéder lorsqu'un ennemi est touché, pour savoir combien de points doivent être ajoutés au pointage.

Affichez le pointage dans une zone de texte (UI texte), de façon semblable au démo. Il faut créer le UI texte.

Mort du personnage et scène finale

Créez d'abord une scène finale semblable à celle du démo. N'oubliez pas d'ajouter cette scène dans le *buildSettings*.

Lorsque le joueur est touché par un ennemi:

- son animation de mort joue,
- Les ennemis arrêtent de se déplacer et jouent leur animation Idle. Utilisez une variable statique qui détermine la fin de la partie dans le script du joueur et tous les ennemis vérifient sa valeur constamment.
- la scène finale doit être chargée.

La scène finale :

- doit présenter au joueur le pointage qu'il a réalisé.
- Lorsqu'on appuie sur la barre d'espacement, la scène du jeu doit être chargée et le pointage doit être réinitialisé à 0.

Attention: Comme mentionné dans la vidéo du cours sur les NavMesh, je m'attends que les AIs puissent monter sur la voiture et sur un cube que vous avez ajouté dans la scène.

REMISE:
Remettre la version exécutable et les scripts dans le lecteur Remise Exr5 (ou me le montrer).
Pour vendredi 1 mai, la fin de la journée
S'il n'y a pas d'exécutable alors (-1 point). Vous devez être capable de publier votre jeu.