TP - Découverte du logiciel Unity

Table des matières

1 Introduction

1.1 Découverte d'un moteur de jeu

Dans ce paragraphe, nous allons apprendre ce qu'est un moteur de jeu et ce qu'il va nous permettre de faire. Ce type de logiciel réalise des calculs de géométrie et de physique primordiaux au fonctionnement d'un jeu vidéo. L'objectif est de simplifier la création d'un jeu avec un simulateur en temps réel de ce que l'on est en train de créer. Ils sont très nombreux, on peut citer Unreal Engine, Blender Game Engine, Unity, ...

1.2 Unity

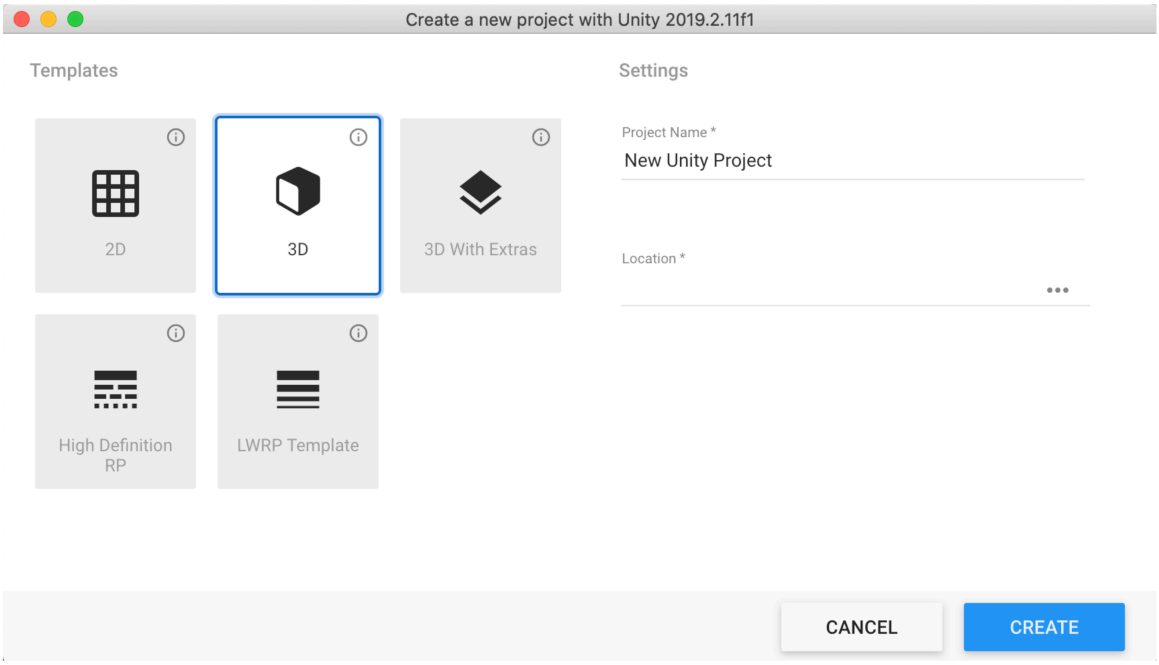
Aujourd'hui, nous allons apprendre à utiliser Unity. C'est sûrement ce que vous utiliserez pour réaliser votre projet du deuxième semestre à Epita. Notre objectif est de vous apprendre à maîtriser l'interface d'Unity et de pouvoir, à l'issue de cette journée, créer un jeu rapide qui utilisera différentes fonctionnalités du logiciel.

Pour se faire, vous allez être guidé par ce TP, du même style que ceux qui rythment notre quotidien à l'école. L'objectif sera aussi de réussir à travailler en groupe pour arriver au bout de ce projet. Mais avant tout cela, il faut voir la théorie.

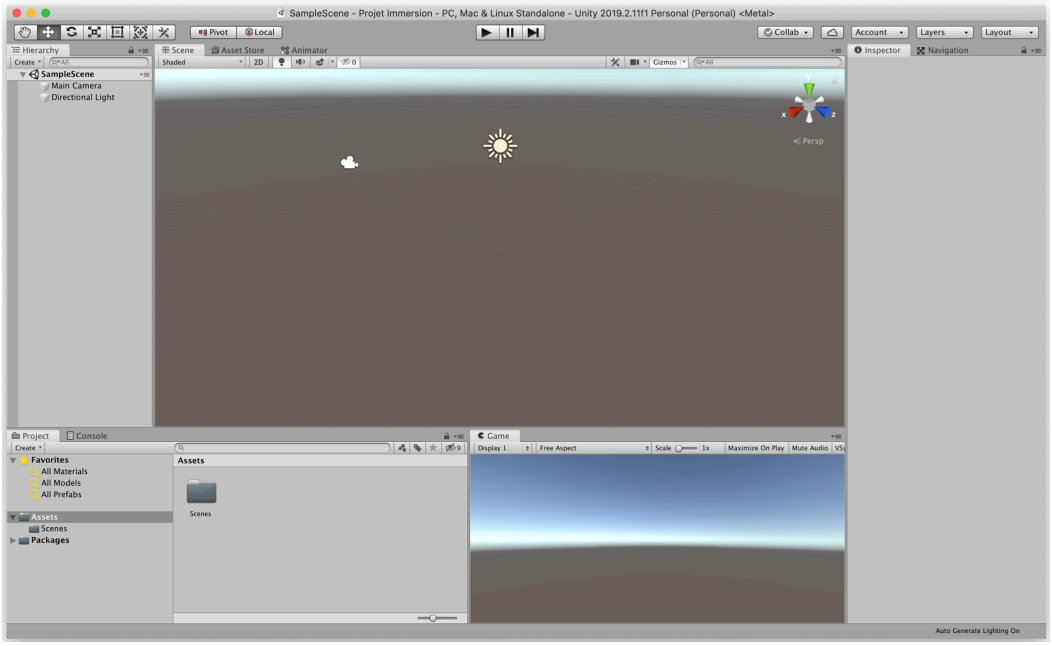
2 Un peu de cours

2.1 L'interface graphique

Avant tout, il faut commencer par ouvrir Unity Hub, c'est ici que sont résumés tous vos projets commencés, sur votre ordinateur ou en collaboration avec d'autres personnes. À partir de cette fenêtre, nous allons pouvoir créer un nouveau projet Unity, utilisant la 3D.



Très bien, nous voici maintenant sur le logiciel, mais on se retrouve vite perdu. Pas de panique, tout d'abord nous allons découvrir les différentes fenêtres qui se sont ouvertes. Nous nous retrouvons alors devant cette fenêtre. (Vidéo pour le lancement disponible sur le site epita-immersion.wistia.com/projects/jcnbqxh3tl)



Les différents onglets

Hierarchy : ici se trouve la scène sur laquelle vous travaillez ainsi que tous les objets qui y sont associés.

Game : il vous permet de visualiser en temps réel ce que l'utilisateur verra lorsqu'il lancera votre jeu.

Inspecteur : (inspector) qui permet de visualiser toutes les propriétés d'un objet.

Project : c'est ici que vous aurez tous les fichiers associés à votre projet, un dossier Asset est créé automatiquement avec un dossier Scenes contenant notre scène, il se remplira au fur et à mesure de notre avancement. Le second dossier, Packages, ne nous sera pas utile aujourd'hui.

Scene : c'est dans cet onglet que le plus gros de notre travail va se passer, on y déplacera, ajustera tous nos objets qui composeront notre jeu.

Vous pourrez découvrir les autres onglets plus tard, ils sont très utiles lorsque vous complexifiez vos projets, aujourd'hui nous voyons les bases. A noter également que tous les onglets sont modulables pour qu'ils puissent convenir à votre utilisation.

Petit + Nous aurons la possibilité de visualiser notre scène comme si nous étions un joueur en utilisant le gros bouton play au-dessus de l'onglet scène.

2.2 Navigation dans l'onglet Scene

Nous avons vu à quoi servait l'onglet le plus grand à l'ouverture d'Unity mais naviguer à l'intérieur de ce dernier peut s'avérer complexe.

Un simple clic permet de sélectionner un ou plusieurs objets déjà insérés dans la scène.

Un clic avec ALT enfoncé permet de modifier la rotation de la caméra sur les 3 axes (3 dimensions).

Un clic avec ALT et CTRL permet de zoomer ou dézoomer dans la scène.

Ce sont ici des raccourcis indispensables à votre efficacité dans l'usage de Unity, il en existe d'autres mais qui demandent une maîtrise plus poussée du logiciel.

2.3 Les objets

Maintenant que vous connaissez les différents onglets, nous allons pouvoir développer un peu plus concernant leur utilisation. Vous avez dû remarquer que des objets étaient déjà créés (une caméra et une lumière). Ce sont les objets qui permettent de bien démarrer la création de notre jeu. La lumière éclaire toute la scène directement, et la direction de son rayon permet à Unity de faire la gestion des ombres.

Passons directement au concret. Commencez par effectuer un clic droit dans l'onglet Hierarchy et sélectionnez 3D object -> Cube. Nous voici avec un cube au milieu de la scène, un bon début. Il est apparu dans l'onglet Hierarchy, sur notre scène et même dans l'onglet Game. Pour finir, dans l'onglet Inspector, il y a toutes les propriétés du cube, que nous découvrirons rapidement. Essayez de découvrir comment faire bouger votre cube avec les flèches présentent autours de lui quand vous cliquez dessus.

Voyons maintenant les types d'objets

3D Object : ce sont des objets qui comme leurs noms l'indiquent sont en 3 dimensions, on retrouve des formes classiques telles que le cube, le cylindre, la sphère, ... (à vous de les essayer).

GameObject : c'est un objet vide qui permet de regrouper plusieurs objets entre eux (nous nous en servirons sûrement plus tard).

UI : ce sont tous les objets d'interaction avec l'utilisateurs, on utilise un canvas qui va nous permettre de gérer le menu, l'affichage d'un score, de points de vie, ...

2D Object : à l'identique avec les objets 3D, ici ce sont des objets à 2 dimensions que vous pouvez découvrir.

Nous allons modifier notre cube à présent. Dans l'onglet de l'inspector, vous pouvez voir des paramètres de taille (scale) et de rotation. Le but est de comprendre comment sont orientés les axes pour modifier les bonnes valeurs et ainsi transformer son cube en un rectangle, un pilier, un terrain, ... tout est possible il suffit d'essayer.

2.4 L'inspector

Maintenant que vous savez ce que sont les objets, nous allons étudier leurs propriétés. L'inspector est en fait l'inventaire de tous les components associés à un objet. Il en existe beaucoup et de tous types, de la physique à l'ambiance sonore et l'animation, les components sont primordiaux. On les ajoute en cliquant sur "Add component" en bas de l'onglet puis en les recherchant dans la barre de recherche présente.

2.4.1 Rigidbody

Le rigidbody est un component qui permet d'associer à un objet la gravité. Par exemple il sera attiré vers le sol, il glissera sur une surface. Cela simplifie l'application de la physique sur notre objet.

2.4.2 Collider

Le collider est obligatoire pour gérer les collisions entre objets, appliquer un rigidbody n'est pas suffisant sans collider l'objet passera à travers les autres, ce qui n'est que rarement le comportement voulu. Il existe plusieurs types de collider, qui s'adaptent ou non à notre forme de l'objet. Il existe des box collider, sphère collider, ... Vous pouvez l'expérimenter en recherchant collider dans la barre de recherche de component.

Vous avez des options pour chaque component, dans notre cas nous pouvons décider de la taille du collider mais aussi une option très intéressante appelée IsTrigger, qui permet de désactiver la collision sans pour autant désactiver la zone de collision. Cela permet, avec du code, d'effectuer une action quand un objet entre dans la zone de collision d'un autre.

Voici un aperçu rapide des components, vous en découvrirez de plus en plus au fur et à mesure de l'utilisation de Unity, pour l'instant cela suffit pour débuter.

3 Un peu de pratique !

Commencez par fermer le projet de jeu que vous avez créé jusqu'à présent. Dans Unity Hub, ouvrez le projet appelé Immersion EPITA. C'est dans ce dernier que vous ferez les exercices qui vont sont proposés.

Maintenant que vous commencez à comprendre comment fonctionne Unity il est temps de le mettre en pratique, rien de mieux pour apprendre. Je vous propose un exercice simple pour commencer, essayons de créer une scène avec un terrain de taille 64x64 et de hauteur 0.5. Y ajouter un tremplin d'un angle de 30°, une sphère puis un carré qui subissent la gravité. Au lancement de la scène le carré doit tomber sur le terrain et y rester, la sphère doit tomber sur le tremplin et donc roulera automatiquement grâce à Unity. **Une fois le projet Immersion EPITA ouvert, rendez vous dans "Scenes/3 - Pratique" pour faire cet exercice**.

Si vous ne comprenez pas ce que vous devez faire ici ou que vous avez envie de savoir quoi faire, rendez vous dans la scene de demo qui se trouve dans **"Scenes/3 - Demo"**

4 Animation

Le domaine de l'animation vous semble peut-être encore loin de votre maîtrise de Unity mais en fait c'est un concept simple à prendre en main et qui permet de faire rapidement des choses plus réalistes. Pour commencer à travailler, **ouvrez la scene qui se trouve dans "Scenes/4 - Animation"** (elle comporte les éléments que vous allez devoir animer). Tout d'abord, il faut ouvrir l'onglet d'animation, il n'est pas chargé au lancement d'Unity mais pas de panique ! Il faut simplement aller dans la barre d'outils en haut, cliquer sur windows puis animation (et non animator !).

Parfait, maintenant que votre fenêtre est ouverte, nous allons essayer d'animer le cube rouge pour qu'il fasse des allers retours entre les deux murs verts. Rien de plus simple, sélectionnez le cube rouge puis allez dans l'onglet animation. L'objectif va être d'enregistrer le mouvement souhaité pour le cube. Nous voulons qu'il passe d'un mur à l'autre il va donc falloir le faire partir de l'un et au cours de l'animation l'emmener vers l'autre. Il faut savoir que nous ferons ensuite jouer l'animation en boucle, pas de problème pour les allers et les retours donc. Cliquez sur Create puis nommez l'animation comme bon vous semble.

À présent, il va falloir ajouter le type de transformation que va suivre notre cube pendant l'animation. Pour cela, nous utiliserons des properties (propriétés). Cliquez sur add property puis sur transform et position et enfin appuyez sur le petit + pour valider. Nous voici avec la propriété position de notre cube. Il va maintenant falloir la modifier pour créer l'animation. Tout d'abord, vous voyez que l'animation dure 1 seconde, nous allons alors faire un clic droit à la moitié et faire add key. Avec votre souris, mettez le curseur de l'animation sur la partie bleue juste au-dessus de la nouvelle clé puis lancez l'enregistrement avec le bouton rouge. Déplacez le cube rouge dans l'onglet Scène jusqu'à l'endroit voulu, soit contre le mur d'en face puis retirez l'enregistrement. Vous pouvez lancer la scène en cliquant sur play, le cube est maintenant animé.

Le cube est animé mais il va peut-être un peu trop vite à votre gout ? Pour modifier cela, rien de plus simple : déplacez la dernière clé à 2 secondes et la clé que vous venez de créer à 1 seconde. L'animation est 2x plus lente à présent. Vous pouvez donc ajuster à votre guise vos animations pour qu'elles correspondent aux attentes.

5 Et le code ?

Il est vrai que jusqu'ici nous n'avons pas parlé du code du tout. Sur Unity, on utilise du C#, pas de panique il est plutôt simple à prendre en main pour commencer sur Unity. C'est un langage de Microsoft, nous allons donc utiliser Visual Studio pour éditer notre code aujourd'hui (à l'EPITA, une license Rider est offerte, c'est un éditeur qui est proche de Visual Studio). Pour commencer, dans Unity, faites un clic droit dans l'onglet Project où se trouvent nos fichiers puis Create -> C# Script.

5.1 La syntaxe

Voilà votre script, cliquez dessus pour l'éditer. Vous pouvez voir le code par défaut, généré automatiquement.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class NewBehaviourScript : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start()
9      {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
```

5.1.1 Using

Les 3 premières lignes correspondent aux dépendances du Script, ici Unity les génèrent automatiquement aussi pour qu'il fonctionne avec l'utilisation que vous allez en faire.

5.1.2 Méthode Start

Les méthodes sont les fonctions, vous pouvez en créer avec le nom que vous désirez mais certaines sont prédéfinies pour Unity, ici Start est lancée dès le début de votre jeu/scène et permet par exemple de définir des variables, appliquer des modifications au lancement de la scène, ... les possibilités sont infinies.

5.1.3 Méthode Update

Cette méthode est appelée à chaque image et effectue le code qui est écrit entièrement. Il faut faire attention car elle peut vite devenir très couteuse en calculs, mais indispensable pour les déplacements, les animations du personnage, ...

5.1.4 Vos méthodes

Pour créer vos méthodes, il vous suffit d'imiter la syntaxe des deux méthodes générées automatiquement. Par convention, les noms des méthodes commencent par une majuscule. Essayez d'être toujours le plus explicite possible dans vos noms, histoire de ne pas vous perdre et de rendre votre code plus simple à comprendre. Pensez à ajouter des commentaires avec la syntaxe //.

5.2 Variables

Pour déclarer des variables en C#, vous devez toujours connaître le type de ce que vous déclarez : si vous voulez un nombre entier c'est int, float pour un nombre à virgule, string pour une chaine de caractère, ...

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

public class NewBehaviourScript : MonoBehaviour
{
    public int jesuisunentier = 0;
    private string jesuisunechainedecaratere = "Bonjour";
    float virgule = 3.5f;
}
```

Ce qui suit le type est le nom de votre variable, ici virgule aura la valeur 3.5f soit 3,5. À noter que vous devez toujours finir votre déclaration par un point virgule ; qui fait comprendre à C# que vous avez fini. Vous avez aussi remarqué que l'on peut ajouter public ou private devant notre déclaration, rien de bien compliqué : ce qui est défini en public est accessible depuis les autres scripts, le private ne l'est pas (aussi facilement).

Ce qui est encore mieux, c'est que vous pouvez aussi déclarer des variables qui pointent vers un objet de votre scène. Par exemple vous pouvez ajouter un cube avec un rigidbody à un Script et via une ligne de code, activer ou non ce rigidbody qui permet par exemple de déclencher sa chute quand on entre dans une zone. Pour cela, on utilise le type GameObject et nous ne sommes pas obligé de lui donner une valeur, simplement un nom.

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

public class NewBehaviourScript : MonoBehaviour
{
    GameObject moncube;
}
```

C# reconnait alors moncube comme un game object et vous pourrez le définir dans Unity par un simple glisser-déposer de l'objet désiré dans la zone qui apparaitra dans l'inspector au niveau du Script.

5.3 Attributs

Chaque type possède des attributs auxquels on peut accéder facilement après l'avoir défini dans une variable. On utilise pour cela le GetComponent<>, en appliquant derrière des paramètres que l'on souhaite modifier. Pour exemple, reprenons le cube d'au dessus. L'objectif est de désactiver son BoxCollider. On effectue cette action simplement, en utilisant :

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

public class NewBehaviourScript : MonoBehaviour
{
    GameObject moncube;
    void Start()
    {
        cube.GetComponent<BoxCollider>().enabled = false;
    }
}
```

Ici, au lancement de notre scène, le cube va perdre son BoxCollider, on pourra donc le transpercer.

Vous voici au bout des exercices de découverte de Unity, bravo ! Certaines notions peuvent encore paraître abstraites mais ne vous inquiétez pas, toutes les questions seront les bienvenues si vous avez besoin d'explications supplémentaires tout au long de la journée.

Maintenant, vous connaissez les bases du développement de jeu avec Unity et l'objectif va être de créer le vôtre ! La prochaine section y sera dédiée et vous allez mettre en pratique votre apprentissage. De la création de l'environnement 3D à la mise en place de mécanismes par du code, vous allez tout réaliser ! Bonne chance.

6 À vous de jouer !

6.1 Préparation du jeu

Avant de se lancer dans la création d'un jeu, il faut y réfléchir et penser à ce que nous voulons mettre en place. Ici, on ne va pas se compliquer trop la tâche : un jeu en solo avec des textures basiques à la première personne en utilisant les "Standard assets". On ne vous a pas présenté les assets dans ce tp ni ce qu'est l'asset store.

Pour faire simple, l'asset store est un onglet dans Unity qui permet d'accéder à des textures, personnages, animations et tout ce que l'on peut imaginer implémenter dans notre jeu. Ce sont des assets soit gratuits soit payants, créés par des utilisateurs ou bien par Unity lui même pour aider les développeurs. Aujourd'hui nous allons simplement voir les Standard assets qui appartiennent à Unity et contiennent des objets préfabriqué, le code d'un personnage FPS contrôlable grâce à la souris, et bien d'autres choses utiles.

Maintenant, vous pouvez ouvrir la scène appelée "6 - A vous de jouer" dans Unity, l'asset est déjà chargée dans l'environnement du jeu, nous allons commencer à l'utiliser.

6.2 L'environnement 3D

Vous pouvez commencer par ajouter un terrain en allant chercher un préfabriqué du standard asset. Je vous laisse le chercher, il s'appelle "Floor prototype 64x64x1". À partir de la vous pouvez commencer à ajouter des objets pour créer tout un environnement de jeu dans lequel vous allez faire évoluer un personnage. Pour l'instant, tout sera blanc. Vous êtes totalement libre du choix des objets, de leur taille, forme, ... Je vous laisse simplement ici un exemple de ce que vous pouvez réaliser, laissez exprimer votre créativité : tout est possible.

6.3 Un personnage

Ici on ne va pas prendre du temps pour créer votre propre personnage en gérant ses déplacements même si cela est tout à fait possible et abordable avec quelques lignes de code. Je vous propose d'utiliser l'asset avec le personnage FPS. Pour cela, rendez vous dans

Standard Asset/Characters/FirstPersonCharacter/Prefab/RigidBodyFPSController

Glissez le sur votre scène, maintenant vous avez un joueur avec des collisions, une caméra contrôlable à partir de la souris. Vous pouvez lancer la scène pour vous déplacer dans votre environnement, tout devrait fonctionner.

6.4 Les textures

Votre jeu est encore un peu trop blanc, nous allons voir comment ajouter les textures à vos objets sur la scène. Cela est très simple, il faut tout d'abord créer des "Materials", en leur appliquant une couleur par exemple. Pour ce faire, placez vous dans l'onglet project et faites un clic droit pour créer tout d'abord un dossier qui contiendra les matériaux que vous pouvez appeler "Matériaux". Ensuite entrez dans ce dossier et créez cette fois-ci un material. Vous pouvez l'appeler comme bon vous semble, ici nous n'allons voir que les couleurs, alors choisissez une couleur que vous voulez ajouter au jeu. Ensuite dans l'inspector du matériel, vous avez des options, ici nous allons utiliser seulement le rectangle qui est d'origine blanc pour y modifier la couleur. Pour appliquer la couleur aux objets de notre scène, il suffit maintenant de glisser le matériel sur l'objet directement dans l'onglet scène. Vous être maintenant capable d'ajouter autant de couleurs que vous voulez.

6.5 Animations

Comme vous l'avez appris plus tôt, vous pouvez ajouter des animations pour votre jeu sur n'importe quel objet, encore une fois tout est possible : amusez vous.

6.6 Implémenter du code

Pour créer votre premier code, nous allons voir comment implémenter un menu pause facilement. Je vous ai facilité la tâche en créant dans un premier temps ce que l'on appelle un canvas, qui est un objet qui vient se superposer à la vision de la caméra. J'y ai crée le menu de pause, il s'appelle MenuPause et il est désactivé pour ne pas venir gâcher la vue du joueur. Commencez par créer un script pour le menu de pause.
Je vais vous expliquer comment faire pour accéder à des objets de la scène depuis le script, ici nous aurons besoin du jeu et du menu de pause. Pour cela, il faut utiliser un champ serialisé. Il va permettre de donner un nom à un objet entré directement dans unity en paramètre du script. Pour déclarer cela, on utilise en C#

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

public class Pause : MonoBehaviour
{
    [SerializeField] GameObject nom;
}
```

Il nous faut ainsi 2 champs serialisés, un pour le joueur et l'autre pour le menu. Il nous fait également un booléen pour savoir si le jeu est en pause ou nous, déclaré avec la valeur False au départ. Ensuite, en utilisant la méthode Update(), il va falloir vérifier quand le joueur appuie sur la touche echap. Pour cela, on va utiliser la fonction Input.GetKeyDown(KeyCode.Escape) qui revoie un booléen pour savoir si la touche echap est pressée. Si cette condition est vraie, il faut vérifier si le jeu est en pause ou non et en fonction activer ou désactiver le personnage et le menu de pause. Pour activer ou désactiver un objet, il faut utiliser nom.SetActive(true) ou (false) en fonction de l'état souhaité. Il faut aussi modifier le Time.timeScale à 0 si le jeu est en pause et 1 sinon. Cela évite à vos animations, fonctionnalités de fonctionner pendant que le jeu est en pause. Il faut aussi penser à modifier la valeur du booléen pour qu'il soit à jour avec le statut de notre jeu.

Je vous laisse essayer par vous même arriver à un code qui fonctionne, la solution est disponible à la page suivante. Pour implémenter votre code, clissez le sur la lumière directionnelle. Cliquez ensuite sur cette dernière, dans l'inspector en naviguant vous devriez voir votre script, avec, si tout se passe bien, 2 objets à compléter.Glissez y le FPS controller à l'emplacement du joueur et MenuPause à l'emplacement du menu. Vous devriez pouvoir à présent lancer le jeu et voir qu'à l'appui de la touche echap le menu de pause apparait puis disparaît : c'est génial !

Après plusieurs tests vous vous rendez sûrement compte que cela n'est pas parfait et c'est tout à fait normal. D'abord les boutons ne sont pas fonctionnels et la souris disparaît peut être quand vous essayez d'y cliquer.

Nous allons améliorer tout cela. Créons notre propre système pour cacher la souris mais aussi rendre les boutons fonctionnels. Nous allons débuter par le plus simple : il faut 2 fonctions pour cacher la souris qui sont

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

public class Pause : MonoBehaviour
{
    Cursor.lockState = CursorLockMode.Locked; //lock le cursor
    Cursor.lockState = CursorLockMode.None; //unlock le cursor
    Cursor.visible = true; //cache ou non le curseur
}
```

Vous allez essayer de faire ce code par vous même, il faut penser à ajouter la méthode start qui doit cacher le curseur et le lock, mais aussi retirer le Cursor lock de votre FPS controller

Une fois cela fait, testez à nouveau le code cela devrait un peu mieux fonctionner mais nous allons rendre les boutons fonctionnels. Pour cela, dans le code il va falloir créer notre propre méthode qui permet de reprendre le jeu et une méthode qui quitte le jeu. La première est une simple reprise de ce que vous avez fait dans la méthode update, la seconde plus complexe et que peu utile à notre niveau, elle se trouve en dessous, vous pouvez la copier.

```
public void Quit()
{
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit();
#endif
}
```

À présent, il ne reste plus qu'à lier nos méthodes aux boutons. Pour cela ouvrez l'arborescence du menu de pause, et sélectionnez le bouton reprendre. Dans l'inspector vous voyez un champ OnClick, c'est ici que l'on va ajouter la méthode. Appuyez sur +, et glissez la lumière dans le rectangle. Au dessus, se trouve un menu déroulant actuellement sur "No fonction", ouvrez le et sélectionner le script de menu puis cherchez votre méthode pour reprendre le jeu. Pour le bouton qui sert à quitter, c'est exactement la même manipulation. Et voilà, plus qu'à tester le code et vous avez fini votre menu de pause !

Et voilà, vous avez réussi à suivre ce TP de découverte sur Unity. C'est un logiciel extrêmement vaste qui permet une grande liberté lorsqu'on le maîtrise. Il est le centre du projet du second semestre à EPITA et vous serez très sûrement amené à devoir l'utiliser. Pas de panique si aujourd'hui cela a été compliqué ! Le TP a pour but de vous faire découvrir une large partie des possibilités, à l'image de ceux qui vont seront proposé toutes les semaines, eux plus centrés sur de la programmation.