

# Jeux Vidéo

Emma Genthon - Vincent Merrouche

Janvier 2023



## 1 Introduction

Unity est un moteur de jeu et une plateforme de développement de contenu en 3D qui permet de créer des jeux, des applications et des expériences interactives pour de nombreuses plates-formes, y compris les ordinateurs de bureau, les consoles de jeu, les mobiles et les réalisations virtuelles. Depuis sa création en 2005, Unity a été adopté par de nombreux développeurs et studios de jeux pour créer des titres de grande qualité, allant des jeux indépendants aux blockbusters triple A.

Dans ce TP, nous allons utiliser Unity pour découvrir les principaux concepts et outils de développement de jeux en 3D. Nous allons apprendre à créer une scène, à ajouter des objets et des personnages, à utiliser les fonctionnalités de script et de physique. Nous effectuerons cela dans le but de créer un jeu de flipper.

Le flipper est un jeu de table qui consiste à envoyer une boule de métal sur un plateau incliné en utilisant des flippers pour frapper la boule et l'envoyer vers les différents éléments du plateau. Nous allons découvrir les différentes étapes de la création de ce type de jeux, de la conception du plateau à la programmation du gameplay et des effets sonores.

## 2 Création du Menu

Dans cette partie nous allons nous attarder sur la création du menu, notre menu va être composé de différents canva. Chaque canva va représenter une des fenêtres de notre menu. Il nous faut pour cela commencer par appliquer un fond à notre canva qui va faire office de fond d'écran. Nous appliquons ensuite. Nous commençons par créer notre canva qui va faire office de fond d'écran :



FIGURE 1 – Canva de fond de notre menu

Ensuite nous plaçons sur ce canva différent UIIn, il s'agira ici de bouton qui prendront la forme de sprite, nous décomptons 3 boutons, un bouton start qui va nous servir à lancer le jeux, un bouton option qui va nous permettre de régler différent paramètre dans le jeux et un bouton quit pour quitter l'application.



FIGURE 2 – Bouton start



FIGURE 3 – Bouton Option



FIGURE 4 – Bouton Quit

Nous positionnons nos différents point et obtenons donc notre menu final :



FIGURE 5 – Menu final

A présent nous allons maintenant écrire notre script qui va permettre de réaliser les fonction de changement de scène et quitter. En effet de base, il n'y pas de fonction existante qui permette ce genre d'action nous allons donc devoir les coder.

### 3 Création de notre scène

Notre scène va être composé de plusieurs éléments, pour commencer nous allons avoir besoin d'une base, un socle. Nous choisissons pour cela un mesh de flipper 3D sur lequel nous rajouterons ensuite nos obstacles ainsi que tous les différents éléments nécessaire.

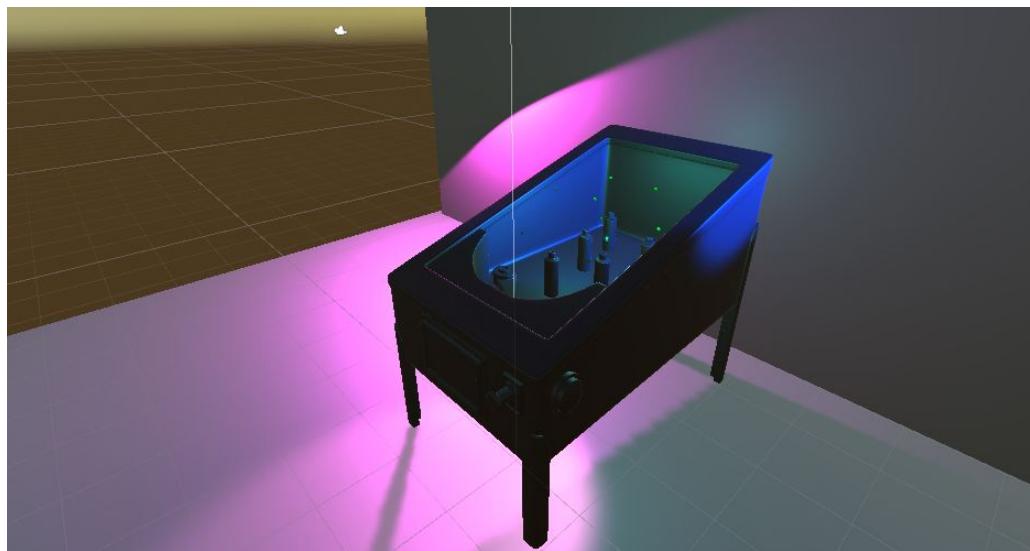


FIGURE 6 – Mesh flipper

On va ensuite ajouter un plan dans le flipper qui va faire office de sol pour notre balle, afin d'éviter

que notre balle sorte du flipper nous ajoutons également un plan transparent qui va faire office de plaque en verre empêchant la balle de sortir du flipper.

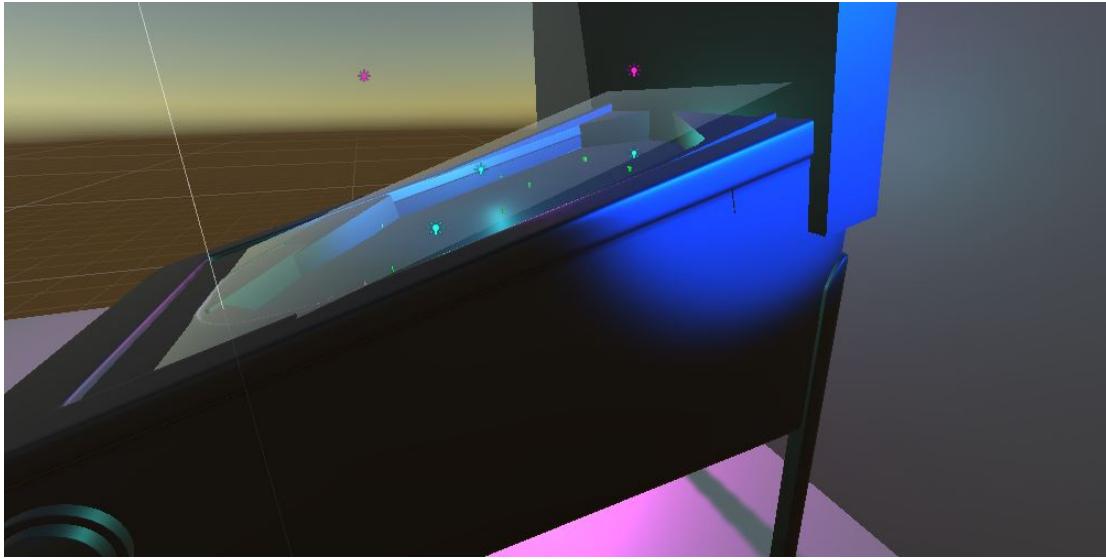


FIGURE 7 – Plan du flipper

On va alors rajouter sur notre flipper les obstacles, ils seront composé soit de sphère, de cylindre ou de mesh réalisé sur blender (particulièremment pour les coins du flipper). Nous attribuerons à chacun de notre obstacle une hitbox qui va permettre de gérer les collisions avec notre balle.

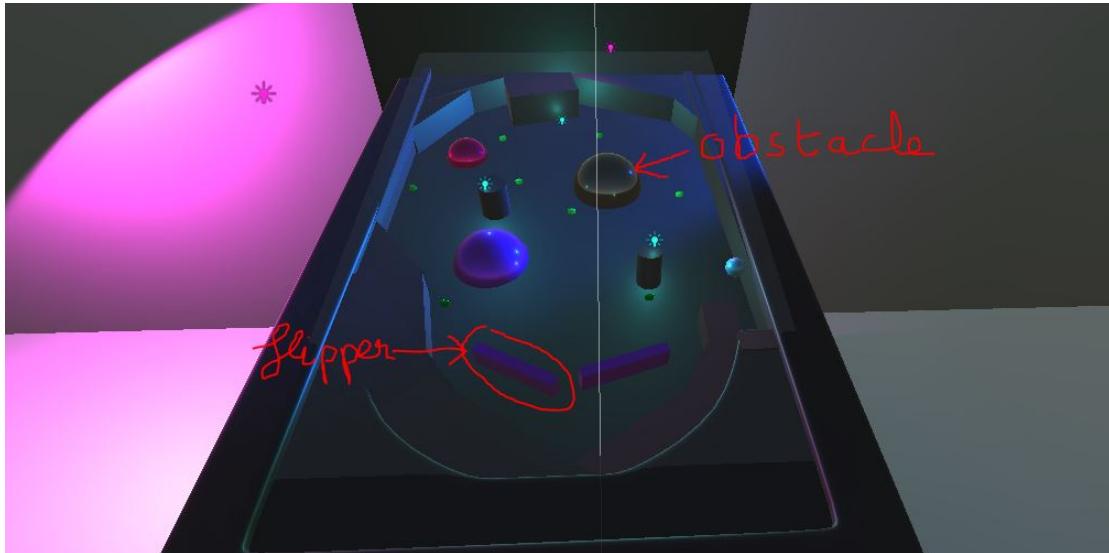


FIGURE 8 – Obstacle du flipper

Nous ajoutons ensuite les collectibles de notre flipper et nous obtenons enfin notre scène. Il ne nous reste plus qu'à fixer une caméra et gérer les éclairages. Nous avons juger qu'une caméra fixe en vu du dessus était la meilleure option pour notre jeux de flipper.

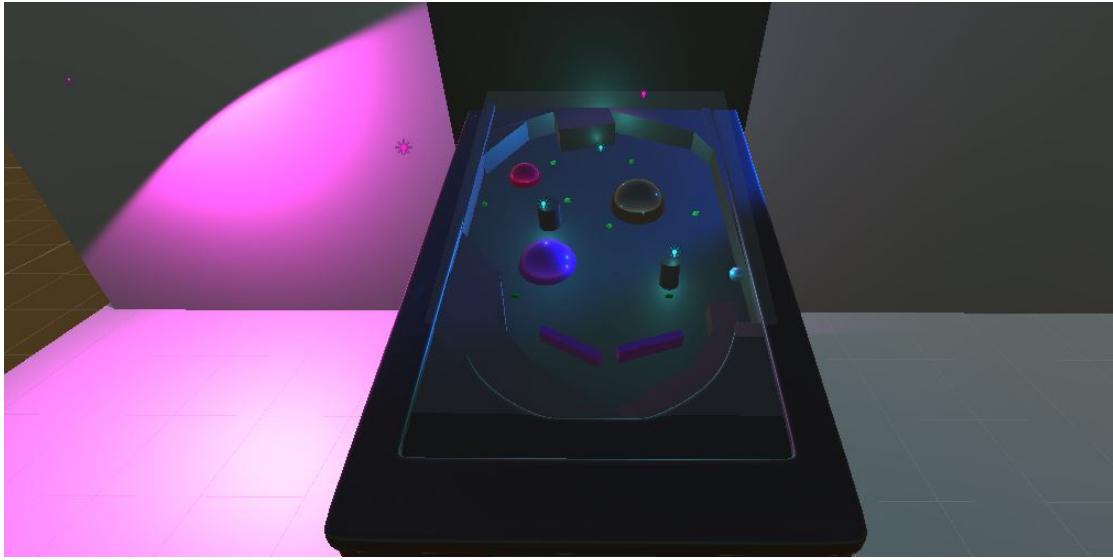


FIGURE 9 – Scène complète

## 4 Mise en place des fonctionnalité du flipper

Après avoir créer la scène, nous allons créer les éléments nécessaires à notre jeu du flipper. Nous allons avoir besoin des flippers (pour renvoyer la balle durant le jeu), le plunger (pour envoyer la balle au départ) et les obstacles de notre jeu.

### 4.1 Flippers (Hinge Joint)

Nous avons précédemment créer deux cubes et nous les avons insérer sur notre scène en bas de notre plan. Nous les redimensionnons pour qu'ils soient plus larges.

Nous ajoutons ensuite le composant 'Hinge Joint' à nos deux flippers. Les Hinge Joint vont permettre de faire pivoter nos cubes selon un axe qu'on définit nous même. On peut observer ci-dessous comment est composé notre hinge joint :

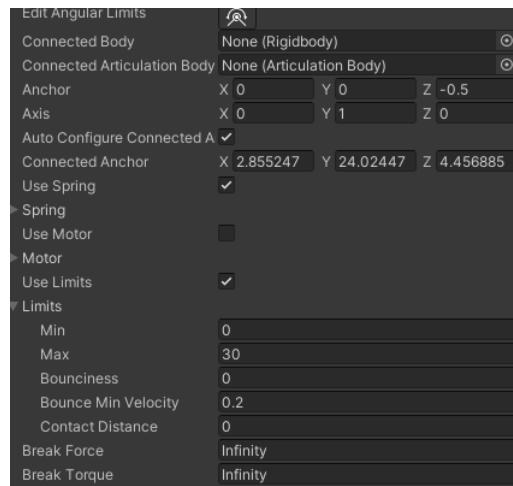


FIGURE 10 – Composant Hinge Joint du flipper de droite

On a fixé des limites d'angles pour chacun des flippers. On définit aussi l'axe de rotation : ici par exemple, on le fixe à un -0.5 selon la direction z du repère directement fixé sur l'objet. Les Hinge joint vont alors pouvoir pivoter avec des limites selon un axe.

On réalise par la suite l'activation des flippers selon les touches du clavier. On souhaite activer la

rotation du flipper de gauche avec la touche 'a' et celui de droite avec la touche 'e'. Dans le script, on fixe alors la vitesse de rotation, la position au repos et la position après pression de la touche ainsi que le nom de l'entrée du clavier (InputName). On définit un ressort de l'hinge joint (pour avoir un effet de ressort et que l'hinge joint reprenne sa position initiale après avoir atteint la position maximale) et on change la position de sa target lorsque l'on reçoit la valeur de touche correspondante du clavier. Afin de récupérer la valeur du clavier, on utilise l'Input Manager, intrinsèque à Unity, et on définit deux nouveaux axes : L\_Flipper et F\_Flipper. On appelle de la même manière, respectivement dans les paramètres du flipper de gauche et de droite, le nom de la variable globale InputName. Les deux flippers posséderont le même script mais auront un InputName différent, 'a' pour le flipper de gauche et 'e' pour le flipper de droite.

## 4.2 Plunger

On a aussi importé dans notre scène, deux cubes superposés : un qui est 'transparent' (sans mesh renderer) et défini comme un trigger, et un cube non trigger mais apparent (avec un mesh renderer). On souhaite envoyer la balle après pression de la touche Espace et augmenter, en même temps une barre de jauge sur la droite (dans le canvas). On incrémente donc la barre de jauge ainsi que la valeur de la puissance tant que on est appuyé, et on utilise la méthode AddForce sur la balle avec la puissance calculée lorsqu'on relâche la touche. On remet aussi la barre de jauge à zéro lorsqu'on relâche. On ajoute aussi le fait que la jauge et la possibilité d'augmenter la puissance de notre balle seulement lorsqu'on détecte une collision entre la balle et le plunger.

## 4.3 Obstacles

On souhaite que les obstacles 'rejetent' la boule lorsqu'il y a collision entre les deux. On effectue alors le même fonctionnement que pour le plunger : on définit un obstacle 'transparent' (sans mesh renderer) et défini comme un trigger, et un obstacle non trigger mais apparent (avec un mesh renderer). Des qu'il y a collision entre l'obstacle et la sphère, on ajoute une force à la sphère. Le jeu sera ainsi plus dynamique et plus pertinent.

# 5 Mise en place des UI

Les UI vont représenter les interfaces utilisateurs du jeu, c'est-à-dire dans notre cas le comptage des points, les différents textes indiquant le début du jeu, si on a perdu, si on a gagné. Pour réaliser cela nous allons utiliser un script qui va compter les collectibles et ainsi mettre à jour le texte du compteur. Une fois tous les collectibles ramassés nous affichons notre texte indiquant que nous avons gagné. Les textes sont affichés à l'aide d'un canvas.

# 6 Axe d'amélioration du jeu

Comme axe d'amélioration du jeu, nous aurions pu concevoir différents niveaux de difficulté en créant différentes scènes offrant une disposition d'obstacle et un autre de nombre de collectibles plus ou moins important, on peut ainsi imaginer une évolution dans le jeu allant de scène en scène de plus en plus compliqué. Nous aurions pu également, pour améliorer le côté esthétique du jeu, mettre en place de petites animations pour nos obstacles (comme par exemple des planètes qui tournent sur elles-mêmes).

## 6.1 Gagner/Perdre

Afin de gagner, il suffit de récolter tous les pick-up de la scène. Lorsque cela arrive, un texte de victoire s'affiche. Il est cependant possible de perdre aussi : on a ajouté un cube sans mesh renderer mais avec l'option trigger activé et qui, en cas de contact avec la sphère, affiche un texte de défaite.

# 7 Conclusion

En résumé, la création d'un jeu vidéo de flipper sur Unity a été une expérience enrichissante qui a mis en avant les nombreuses possibilités offertes par le moteur de jeu. Nous avons été en mesure de

profiter de la puissance et de la flexibilité de Unity pour développer un jeu de flipper qui est à la fois divertissant et réaliste.

Le processus de développement a impliqué la création de scripts pour gérer la physique du jeu, la conception d'un niveaux , ainsi que la création de graphismes et de musique.