
TP – SPACE INVADERS

--

SIMON CHAUVIN

SIMON.CHAUVIN@UNIV-PARIS13.FR

-

1 – RÉCUPÉRATION DU PROJET

2 – AJOUT DE LA FONCTION DE TIR

- Utilisez la fonction `Input.getKeyDown(keycode)` pour savoir si la touche espace à été enfoncée. Cette fonction prend en entrée un `keycode` que l'on peut trouver dans la classe `Keycode` (celle-ci possède toutes les touches du clavier). Appelez alors la fonction shoot qui existe déjà (mais qui est vide) dans la classe `Ship`.
- Dans la fonction shoot vous allez maintenant devoir instantier le prefab Bullet et positionner la bullet au niveau du vaisseau. Pour cela vous utiliserez la fonction `Instantiate` et le composant `Transform`.
- Vous pouvez maintenant modifier l'attribut `velocity` du `rigidbody2D` de l'objet `Bullet` nouvellement instantié afin de lui donner une vitesse verticale positive puisque l'on veut que la bullet aille vers le haut.

3 – CRÉATION DES INVADERS

- Pour créer les envahisseurs vous allez devoir utiliser deux boucles imbriquées l'une dans l'autre afin de créer les lignes et colonnes de la formation alien. Cela se passera dans la fonction `Start()` de la classe `InvadersManager`.
- Il existe trois prefabs d'envahisseurs différents déjà disponible en attribut de la classe `InvadersManager`. Assurez vous de respecter cette ordre :

  11111111111

  11111111111

  22222222222

  22222222222

  33333333333

- Il y a donc 5 lignes et 11 colonnes. Vous devrez également mettre chaque envahisseur dans un tableau de `Transform` d'une taille de 55 éléments.

4 – Destruction des envahisseurs

- Dans la fonction `OnCollisionEnter2D (Collision2D collision)` de la classe `Invader` vous devrez détruire l'objet `Bullet` mais aussi l'envahisseur concerné.
- Pour l'objet `Bullet` vous pouvez vous contenter d'un simple appel à la fonction `Destroy` mais pour l'objet `Invader` nous allons devoir faire appel à la fonction `destroyInvader(Transform invader)` présente dans la classe `InvadersManager` et qui attend d'être complétée.
- A l'intérieur de cette fonction nous allons donc parcourir l'ensemble des envahisseurs et vérifier si le `Transform` passé en paramètre est égal à un des éléments du tableau, qui se trouve être un tableau de `Transform`.
- Lorsque le bon envahisseur aura été trouvé nous pourrons détruire l'objet `Invader` en question mais également mettre l'élément du tableau correspondant à `null`. Par ce biais nous signifions simplement que l'envahisseur référencé par le tableau n'existe plus et il n'est donc plus référencé par le tableau.

5 – Déplacement des envahisseurs

- Afin de déplacer les envahisseurs nous devons changer leur position à chaque frame. Placez vous dans la méthode `Update()` de la classe `InvadersManager`.
- Vous pouvez maintenant parcourir le tableau des envahisseurs afin de déplacer chacun d'entre eux. Dans la mesure où il est possible que le tableau contienne des éléments `null` vous devrez vérifier que ce n'est pas le cas avant de manipuler l'envahisseur en question.
- Afin de déplacer un envahisseur vous pouvez utiliser la méthode `Transform.Translate (float x, float y, float z)`. Dans un premier temps nous déplacerons les envahisseurs uniquement vers la droite.

6 – Inversion du déplacement des envahisseurs

- Nous allons maintenant faire en sorte que les envahisseurs se déplacent dans l'autre sens lorsque l'un d'entre eux rencontre le bord de l'écran.
- Pour arriver à un tel comportement vous allez devoir créer deux triggers à mettre sur les côtés gauche et droite de l'écran. Pour créer un trigger vous devez créer un `GameObject` vide (GameObject → Create Empty) et lui affecter un composant `BoxCollider2D` (Physics2D → Box Collider 2D) dont vous devez cocher la case `Is Trigger`. C'est ce qui permet de transformer un collider en simple trigger, un élément non solide sur lequel il n'est pas possible de rentrer en collision. Par contre ce trigger appellera la méthode `OnTriggerEnter2D` lorsqu'un objet entrera dans la zone définie par son collider. Pour redimensionner le trigger à votre convenance vous pouvez vous servir de

l'attribut Size visible dans l'`Inspector`.

- Maintenant vous pouvez attacher le script `InversionTrigger` (Add Component → Scripts → InversionTrigger) et compléter la méthode `OnTriggerEnter2D (Collider2D collider)` afin d'appeler la méthode `moveInvadersCloser()`, méthode qui se trouve dans la classe `InvadersManager`.
- Placez vous dans cette méthode `moveInvadersCloser()` et passez le booléen `goingRight` à son inverse grâce au symbole `!`. Cela permettra d'inverser le mouvement des envahisseurs sans se préoccuper de savoir si c'est le trigger de gauche ou de droite qui a été déclenché.
- Vous pouvez maintenant tester ce booléen dans l'`Update` de la classe `InvadersManager` pour savoir si vous devez déplacer les envahisseurs vers la droite ou la gauche (translation sur x positive ou négative).

7 – DÉPLACEMENT DES ENVAHISSEURS VERS LE BAS

- Nous voulons maintenant qu'à chaque fois qu'un envahisseur rencontre un des triggers d'inversion que l'ensemble des envahisseurs inverse alors leur mouvement.
- Placez vous dans la méthode `moveInvadersCloser()` de la classe `InvadersManager` et parcourez l'ensemble des envahisseurs pour tous leur appliquer la même instruction de translation vers le bas.
- Enfin, vous devez augmenter la vitesse des envahisseurs à chaque fois que la méthode `moveInvadersCloser()` est appelée. Vous allez devoir sauvegarder la vitesse des envahisseurs dans un attribut de type `float` afin que celle-ci soit accessible dans l'`Update` et être modifiée dans `moveInvadersCloser()`.

8– PRISE EN COMPTE DU FRAME RATE VARIABLE

- Afin que les mouvements des objets du jeu soient les mêmes pour une machine tournant à 30 FPS que pour une machine à 60 FPS il faut que nous passions nos vitesses en unités par seconde.
- Partout où nous avons utilisé la fonction Translate nous devons modifier la valeur fournie en la multipliant par `Time.deltaTime` qui correspond au temps écoulé depuis la dernière frame. Si par exemple vous appliquiez une translation de -1 pour déplacer les envahisseurs vers le bas il faudra maintenant utiliser une valeur plus haute si vous voulez une vitesse similaire car c'est maintenant des unités par seconde que nous utilisons et non plus des unités par frame. Même sur une machine très lente une seconde correspond à plusieurs frame.

9 – AUGMENTATION DU SCORE

- Lorsqu'un envahisseur est détruit par le joueur il faut augmenter le score.
- Pour ce faire vous devrez utiliser une variable entière pour sauvegarder et augmenter le score ainsi que l'objet `scoreLabel` présent dans la classe

`Invader` pour afficher le score en temps réel.

10 – CONDITION DE DÉFAITE

- Lorsque le joueur est touché par un envahisseur alors il perd une vie, jusqu'à ce qu'il n'ait plus de vie disponible.
- Pour réaliser cela vous devrez utiliser une variable de entière pour sauvegarder et diminuer la vie ainsi que l'objet `Lives` présent dans la classe `Ship` pour afficher les vies en temps réel.
- Enfin, vous devrez créer un nouvel objet `GUIText` à partir de l'éditeur (GameObject → Create Other → GUIText), le placer manuellement dans l'éditeur et l'utiliser dans la classe `Ship` en créant un nouvel attribut public et en lui affectant directement ce nouvel objet dans l'éditeur. Ensuite vous pourrez accéder à l'attribut `text` qui permet de modifier le contenu du texte à afficher.

11 – CONDITION DE VICTOIRE

- Le joueur à gagné lorsqu'il n'y a plus d'envahisseurs à l'écran.
- Nous allons devoir parcourir le tableau d'envahisseurs et afficher un message dans le cas où tous les éléments du tableau sont à `null`.
- Pour afficher le message vous pouvez utiliser la même méthode que dans la question précédente.

12 – AJOUTER LA SOUCOUPE VOLANTE

- Vous allez maintenant faire apparaître une soucoupe volante en haut de l'écran de façon aléatoire. Pour cela nous allons utiliser la classe `Random` qui fournie un attribut décimal appelé `value` dont la valeur aléatoire s'étend de 0 à 1.
- Grâce à cette valeur vous allez pouvoir déterminé la probabilité d'apparition de la soucoupe volante. Par exemple, vous pourriez la faire apparaître toutes les fois où `Random.value` renvoie une valeur supérieure à 0.9.
- Le comportement de la soucoupe est simple. Elle doit traverser l'écran et tirer des objets instanciés à partir du prefab `SaucerBullet`. Ces objets doivent pouvoir détruire le joueur et lui faire perdre une vie.