

Groupe CERCO


Piscine AR & E-Commerce Odoo 12 (PAREO12)

Travaux Pratiques : Jour 4

Règles spécifiques à la journée

- L'utilisation de l'API Physique d'Unity est strictement interdite.

Exercice 00 : Gonflage de Ballon Simulator

	Exercice : 00
Exercice 00 : Gonflage de Ballon Simulator	
Dossier de rendu : ex00/	
Fichiers à rendre : Un fichier scene ex00, Balloon.cs, les assets liés à l'exercice	
Fonctions Autorisées : Debug.Log, Mathf.RoundToInt, Input.GetKeyDown, GameObject.Destroy	


Bien que le titre soit relativement explicite nous allons quand même préciser. Vous devez mettre en place une scène comportant un ballon. On doit pouvoir gonfler ce ballon à l'aide de la barre d'espace. Si le ballon est trop gonflé il explose ou si on ne le gonfle pas assez au bout d'un certain temps le jeu est perdu. Le ballon doit visuellement grossir en fonction de son niveau d'air. Lorsque l'on appuie plus sur espace, alors le ballon se dégonfle.

Vous devez également prendre en compte le souffle. Plus on gonfle vite plus on s'essouffle. A partir d'un certain niveau d'essoufflement on ne peut plus gonfler le ballon il faudra donc attendre que notre souffle se régénère.

Quand le jeu se termine vous devez afficher le temps arrondi en entier dans la console de la manière suivante :

```
Balloon life time : 120s
```

Exercice 01 : Quick Time Event

	Exercice : 01
Exercice 01 : Quick Time Event	
Dossier de rendu : ex01/	
Fichiers à rendre : Un fichier scene ex01, CubeSpawner.cs, Cube.cs, les assets liés à l'exercice	
Fonctions Autorisées : Debug.Log, Random.Range, GameObject.Instantiate, GameObject.Destroy, Input.GetKeyDown, Transform.Translate	

Vous avez sûrement déjà joué à Guitar Hero, ou même à des jeux d'actions vous demandant à un moment où à un autre d'appuyer sur certaines touches au bon moment. C'est ce qu'on appelle pour les plus néophytes d'entre vous des QTE ou Quick Time Event.

Lancez la démo pour vous faire une idée. Vous devez utiliser les A, Set D pour jouer à ce jeu.

Le but de cet exercice vous l'aurez compris c'est donc de réaliser un QTE. Mettez en place une ligne vers le bas de l'écran et une autre vers le haut ainsi que trois lignes verticales rejoignant ces deux. Faites-en sorte que des carrés soient générés de manière aléatoire sur une des trois lignes verticales tous les X secondes. Ces carrés doivent chacun correspondre à une touche (des touches d'exemples vous sont fournis dans le zip de la journée). Chaque carré doit descendre avec une vitesse aléatoire en direction de la ligne horizontale du bas. Lorsqu'il atteint cette ligne le joueur doit appuyer sur la touche correspondante au bon moment. Chaque carré doit avoir sa propre ligne.


Vous devrez afficher la précision du joueur à chaque pression. La précision correspond à la distance entre le carré et la ligne au moment où le joueur appuie sur la touche. Elle doit être affichée de cette manière :

Precision : 23.454



Les cubes doivent avoir une durée de vie et ne pas rester instanciés dans la scène éternellement une fois sortis de la zone de jeu

Exercice 02 : Mini Golf

	Exercice : 02
Exercice 02 : Mini Golf	
Dossier de rendu : ex02/	
Fichiers à rendre : Un fichier sceneex02, Ball.cs, Club.cs, les assets liés à l'exercice	
Fonctions Autorisées : Debug.Log, Mathf.Clamp, Transform.Translate, Input.GetKey	


Rien de tel qu'un peu de sport. Vous allez donc créer un Mini Golf. Vous devrez donc faire en sorte qu'on puisse tirer dans une balle avec une force définie par la barre d'espace. Plus la pression est longue, plus la balle ira loin. La balle doit avoir une inertie et ralentir avec le temps. Le but étant donc de mettre la balle dans un trou. Si la balle touche un mur elle rebondit dans la direction opposée. Si la balle passe trop vite au-dessus du trou, elle n'y tombe pas.

Pour ce qui est du score nous utiliserons une version simplifiée des règles du mini golf. Le joueur part avec -15 points, à chaque coup manqué il en gagne 5. Si son score est supérieur à 0 il perd, mais la partie ne s'arrête pas pour autant.

A chaque coup le score devra être affiché dans la console de la manière suivante :

```
Score : -5
```

Exercice 03 : Flappy Bird

	Exercice : 03
Exercice 03 : Flappy Bird	
Dossier de rendu : ex03/	
Fichiers à rendre : Un fichier scene ex03, Bird.cs, Pipe.cs, les assets liés à l'exercice	
Fonctions Autorisées : Debug.Log, Input.GetKeyDown, Transform.Rotate, Transform.Translate, Mathf.RoundToInt	


Pour cet exercice c'est une version simplifiée de Flappy Bird que vous allez réaliser. Mettez donc en place l'oiseau au centre de la scène. Il doit tomber jusqu'à ce que le joueur appuie sur espace pour le faire remonter. Si l'oiseau touche le sol ou un des tuyaux la partie est perdue.

Quelques considérations techniques maintenant. L'oiseau ne doit jamais bouger sur l'axe x. Les tuyaux seront toujours à la même hauteur parce qu'on n'a pas encore vu comment faire ça proprement. Il ne peut y avoir plus de deux tuyaux instanciés, vous devez donc les réutiliser. La vitesse doit augmenter progressivement au fur et à mesure que le joueur passe des tuyaux.

Vous devrez également mettre en place un système de score et de temps. Chaque tuyau passé rapporte 5 points. À chaque fin de partie vous devez donc l'afficher de la manière suivante dans la console :

```
Score : 15  
Time : 25s
```

Exercice 04 : Pong !

	Exercice : 04
Exercice 04 : Pong !	
Dossier de rendu : ex04/	
Fichiers à rendre : Un fichier scene ex04, PongBall.cs, Player.cs, les assets liés à l'exercice	
Fonctions Autorisées : Debug.Log, Random.Range, Input.GetKey, Transform.Translate	

Pour terminer un concept qui a fait ses preuves et qui est relativement universel. Vous devrez réaliser un Pong. Si vous ne savez pas ce que c'est, lancez la démo. Le joueur de gauche déplace sa raquette avec les touches Wet S, et le joueur de droite déplace sa raquette avec les touches UpArrowet DownArrow.

Faites-en sorte qu'il y ait donc deux rectangles disposés de part et d'autre d'un terrain. Un carré doit rebondir entre eux deux, sa direction est inversée à chaque rebond, y compris verticalement.

Les deux rectangles doivent pouvoir être contrôlé indépendamment l'un de l'autre de manière verticale à l'aide des touches du clavier, avec pour limites, les murs du haut et du bas. Si le carré atteint un des bouts du terrain il est remis au centre et le joueur opposé gagne un point.

Le score doit être affiché à chaque point dans la console de Unity de la manière suivante :

```
Player 1: 0 | Player 2: 0
```