Clement Lion – GamingCampus groupe Bogota – Octobre 2025

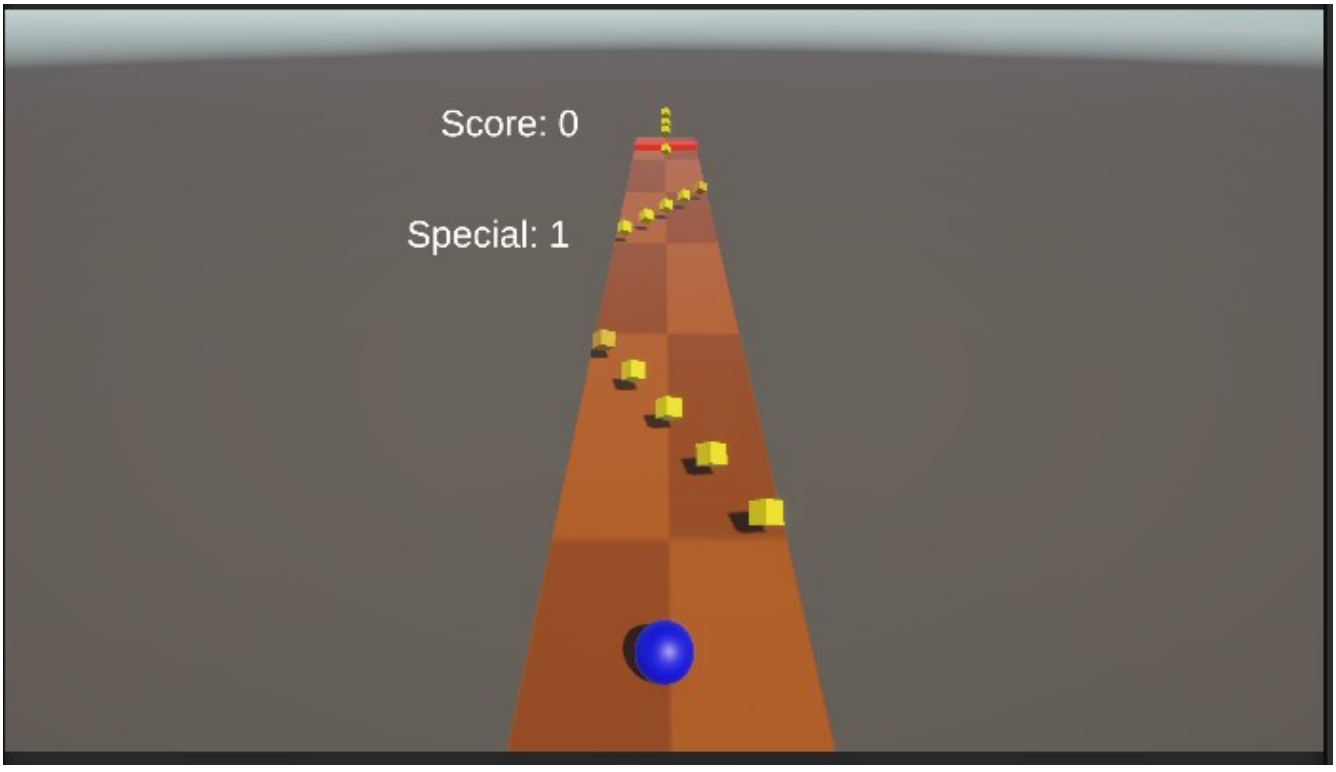
Présentation projet InfiniteRunner.

Ce projet a été codé pour Unity. Si vous souhaitez l'utiliser, créez un nouveau projet unity (appelons le InfiniteRunner) et mettez le code de ce repository dans le dossier InfiniteRunner.

Description

Presentation

Ce projet est un Infinite Runner le but du jeu est de récupérer le score le plus élevé possible sans mourir. Vous pouvez voir ci-dessous un écran de jeu :



Comment jouer ?

Le joueur peut controler la sphère bleue avec les flèches directionnelles (il peut se déplacer de droite à gauche, le niveau défile tout seul). Le joueur peut ****sauter en appuyant sur la barre d'espace**** et ****activer son pouvoir spécial en utilisant la touche entrée****. Le nombre de charge de pouvoi du joueur est indiqué à l'écran à gauche.
Le joueur peut collecter des grains (les cubes jaune) pour augmenter son score.

Condition de victoire : Pas de condition de victoire, le joueur doit juste cumuler le plus grand score possible.

Condition de défaite : Le joueur doit éviter les obstacles rouges. Tout contact avec un obstacle rouge fait perdre la partie. Si le joueur quitte

Progression

En fonction du nombre de grains que le joueur a collecté, il change d'états et débloque de nouveaux pouvoirs spéciaux et des aptitudes. Un pouvoir spécial remplace le précédent et nécessite une charge afin d'être activé, une aptitude est une capacité que le joueur peut activer à tout moment et qui vient enrichir son arsenal. Les pouvoirs spéciaux sont actifs pendant 10s, le joueur gagne une charge de pouvoir spécial à chaque fois qu'il change d'état.

****A chaque changement d'état, le niveau s'accélère.****.

Grains collectés	Pouvoir Spéciaux	Aptitudes
0	Bouclier (Le joueur peut survivre à un obstacle)	N/A
50	Ralentissement (La vitesse du niveau est divisée par 2)	N/A
100	Ralentissement (La vitesse du niveau est divisée par 2)	Planer (En maintenant la barre d'espace, le joueur peut planer dans les airs jusqu'à 3 secondes)
250	Ralentissement (La vitesse du niveau est divisée par 2)	Double saut
500	Ralentissement (La vitesse du niveau est divisée par 2)	Dash (appuyer sur la touche ‘e’ et une direction pour faire un dash dans cette direction)
1000	Ralentissement (La vitesse du niveau est divisée par 2)	N/A

Description du code

Vue d'oiseau

Le dossier principale de ce repository est le dossier Assets. Il contient les éléments suivants :

- Events : contient tous les scriptable objects.
- Materials : contient tous les matériaux.
- Resources : contient tous les prefabs.
- Scenes : contient toutes les scènes de jeu (mainmenu, game over et scène de niveau. Contient aussi un niveau de test afin de créer et tester des chunks).
- Scripts : contient tout le code du jeu.

Focus sur les scripts

Notre dossier Scripts contient les éléments suivants :

- Camera : Scripts pour gérer les mouvements de la caméra en fonction du joueur.
- Components : Script pour gérer les éléments constitutifs d'une scène (chunks, collectibles, obstacles). Gère le spawn et la destruction des Chunks. Les chunks sont choisis au hasard en piochant dans les prefabs à disposition.
- Menus : Scripts pour définir les menus.
- Player : Scripts pour définir les mouvements du joueur, la mise à jour des scores, la gestion de la santé, le changement d'"état du joueur et tous les éléments liés au joueur.
- ScriptableObjects : Scripts pour définir nos scriptableObjects.
- Services : Scripts gérant les services transverses (Gestion des GameOver et singletons).

Focus sur la création d’un niveau infini

Afin de créer un niveau infini, on a besoin de quelques éléments :

- Un dossier rempli de prefabs de types chunks. Ils sont les éléments à partir desquels on va construire notre niveau.
- Un Script Chunks qui gère l’activation et la desactivation des chunks. Chaque chunk possède un ScriptableObject qui controle la rareté du chunk (quelle est sa fréquence d’apparition dans le niveau).
- Un Monobehaviour Spawner. Le Spawner gère la création et l’update des Chunks. Au début du niveau, il crée une liste des chunks de niveau. Le nombre d’instance pour chacun des chunks de niveau dépend de la rareté du chunks. A chaque Update, le Spawner déplace les chunks. Si un

chunks est marqué comme “desactivé”, le Spawner le desactive. Il choisit alors au hasard un nouveau chunks à rajouter depuis la liste des chunks de niveau.
- Un Monobehaviour Destroyer marque comme “désactivé” tous les chunks qui arrivent jusqu’à lui.

Focus sur le Player

Le prefab player est composé de plusieurs éléments.
- Une sphère avec un rigidbody auquel on attache un script PlayerController. On peut ainsi controler la sphère avec les touches. L’objet PlayerController possède toutes les instructions relatives au mouvement (quels inputs utiliser et quelle action déclencher en pressant les touches).
- Un Monobehaviour CollisionHandler. Ce composant déclenche les Events de collisions en fonction de l’objet avec lequel le joueur rentre en collision.
- Un Singleton pour gérer la santé du joueur. Ce singleton reçoit les events de collision et change la santé du joueur en conséquence.
- Un Singleton pour gérer le score du joueur. Il reçoit les events de récupération des collectibles et change le score du joueur en conséquence. Quand le joueur récupère suffisamment de coillectibles, il envoit un Events de Changement d’état (les seuils sont précisés à la partie précédente).
- Un Monobehaviour PlayerPowerUp. Ce composant gère les aptitudes que le joueur a déjà débloqué. Il reçoit les events de changement d’état et débloque les aptitudes au fur et à mesure.
- Une machine à état pour gérer les transitions d’états du joueur et l’utilisation des capacités spéciales en fonction de l’état du joueur.

Mon expérience

Temps consacré au projet (approximation) : 10 jours complets.

Difficultés rencontrées :
- **Impression de perte de contrôle**. Par rapport au code “brut”, où on sait à tout instant quelles sont les entités créées et comment est-ce qu’elles interagissent, il m’est apparu plus compliqué de gérer les composants dans Unity.
- **Méthode d’apprentissage inadaptée à mon profil**. (Cette difficulté est exclusive à moi). Je suis quelqu’un qui apprend en lisant et en faisant de la théorie (j’ai beaucoup de mal à me concentrer sur des vidéos). Le fait qu’il faille faire souvent des manipulations sur l’interface Unity m’obligeait à regarder des vidéos, ce qui contrariait mon apprentissage.

Leçons apprises :
- **Essayer de faire une chose à la fois**. Dans le cadre du projet, on avait un cahier des charges complet avec beaucoup de choses à gérer et en même temps, Unity à apprendre. Démarrer a été laborieux.

- **Faire le tri**. Si je devais expliquer à quelqu'un comment se lancer sur Unity, je lui expliquerais qu'il n'y a pas tant que ça d'éléments importants : les prefabs, les ScriptableObjects, les CoRoutine et les MonoBehaviours. Le reste, c'est du bruit qui va t'empêcher de te focaliser sur l'essentiel.

Passez une excellente journée.