Game Engine UX 2

Natalia GERARD et Paul RANC William LINA

Jeu vidéo Unity – Document explicatif

**EsCave**

Sommaire :

Présentation du projet

Structure de notre code

Difficultés rencontrées

Pour aller plus loin

1. Présentation du projet

Notre projet EsCave est un jeu vidéo de plateforme en 2D. Il comporte 3 niveaux de difficultés croissantes : facile, intermédiaire et difficile. L’univers du jeu est celui d’une grotte rocheuse style carrière d’extraction de mine.

Le but étant, pour chaque niveau, de le parcourir en évitant les obstacles et en récupérant la clé afin d’atteindre la porte ouverte. Lorsque la porte est franchie, un message ‘you win !’ apparait et le niveau est remporté.

Pour déplacer le personnage, les commandes sont les suivantes :

Space bar pour sauter

Q ou flèche gauche pour aller à gauche

D ou flèche droite pour aller à droite

* Se référer à la vidéo de présentation du projet pour une vision complète du jeu

1. Structure de notre code

Voici l’ensemble des noms de scripts que nous avons utilisés pour notre jeu :

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Dans le cadre de notre projet, nous avons adopté une organisation structurée et modulaire de notre code. Cette approche nous a permis de séparer les différentes fonctionnalités du jeu en scripts distincts, facilitant ainsi la maintenance, l'évolution et la collaboration au sein de l'équipe. Voici une présentation détaillée de la structure de nos scripts et de leurs interactions.

**1. Contrôle et Gestion du Joueur**

**PlayerMovement** et **PlayerMovementIntermediaire** sont les scripts centraux responsables de la gestion des mouvements et des états du personnage joueur dans les différents niveaux (facile, intermédiaire et difficile).

**-PlayerMovement.cs** : Ce script gère les mouvements de base du joueur, incluant le déplacement horizontal, le saut, et les interactions avec les objets clés et les portes. Il intègre également la logique de mort du joueur, en déclenchant l’animation de mort et en désactivant les contrôles lorsque le joueur meurt.

**-PlayerMovementIntermediaire.cs** : Adapté pour les niveaux intermédiaire et difficile, ce script étend les fonctionnalités de **PlayerMovement.cs** en intégrant des éléments spécifiques tels que l'affichage du panneau de mort (**DeathPanel**) lorsqu'une collision avec des obstacles mortels (pics, monstres) est détectée. Il maintient également la gestion des clés et des interactions avec les portes, en s'assurant que les contrôles sont désactivés en cas de mort.

**2. Gestion des Interactions avec les Objets**

**KeyPickUp** et **KeyPickUp1** sont des scripts dédiés à la gestion de la collecte des clés par le joueur.

**-KeyPickUp.cs** : Attaché aux objets clé dans les niveaux faciles, ce script détecte la collision avec le joueur, met à jour l’état du joueur en indiquant qu'il possède la clé, et détruit l'objet clé pour le faire disparaître visuellement.

**-KeyPickUp1.cs** : Similaire à **KeyPickUp.cs**, ce script est utilisé dans les niveaux intermédiaire et difficile et interagit avec le script **PlayerMovementIntermediaire.cs** pour mettre à jour l’état du joueur lorsqu'une clé est récupérée.

**DoorTrigger** et **DoorTrigger1** sont responsables de la gestion des interactions avec les portes de sortie des niveaux.

**-DoorTrigger.cs** : Utilisé dans les niveaux faciles, ce script vérifie si le joueur possède la clé lorsqu'il entre en collision avec une porte. Si la condition est remplie, il affiche le panneau de victoire (**WinPanel**).

**-DoorTrigger1.cs** : Adapté pour les niveaux intermédiaire et difficile, ce script fonctionne de manière similaire à **DoorTrigger.cs** mais interagit avec le script **PlayerMovementIntermediaire.cs** pour vérifier l’état de la clé avant d'afficher le panneau de victoire.

**3. Contrôle de l'Interface Utilisateur (UI)**

**DeathPanelController**, **VictoryRetour**, et **MenuController** sont des scripts dédiés à la gestion des éléments de l'interface utilisateur.

**-DeathPanelController.cs** : Ce script gère le bouton "Recommencer" dans le panneau de mort (**DeathPanel**). Lorsqu'il est cliqué, il recharge la scène actuelle, permettant au joueur de rejouer le niveau.

**-VictoryRetour.cs** : Attaché au bouton "Retour Menu" dans le panneau de victoire (**WinPanel**), ce script permet au joueur de retourner au menu principal en chargeant la scène correspondante.

**-MenuController.cs** : Ce script centralise la navigation entre les différents niveaux du jeu en chargeant les scènes correspondantes (Facile, Intermédiaire, Difficile) lorsqu’un bouton du menu est cliqué.

**4. Gestion des Effets Visuels et Animations**

Pour enrichir l'expérience visuelle du jeu, plusieurs scripts sont dédiés aux animations et aux effets graphiques.

**-FloatKey.cs** : Ce script applique un mouvement oscillatoire aux objets clés, les faisant flotter de manière fluide pour attirer l'attention du joueur.

**-FlipOnDirectionChange.cs** : Utilisé pour les ennemis, ce script permet à un objet de se déplacer entre deux points (haut et bas) tout en inversant son orientation visuelle (flip) pour simuler une marche ou un déplacement réaliste.

**-FadeOnKey.cs** : Ce script gère l'effet de fondu appliqué aux objets lorsqu'ils sont récupérés par le joueur. Lorsqu'une clé est acquise, l’objet clé devient progressivement transparent, ajoutant une transition visuelle agréable lors de la collecte.

**5. Contrôle des Panneaux de Victoire et de Mort**

Ces scripts assurent l'affichage correct des panneaux de victoire et de mort en fonction des interactions du joueur avec les objets clés et les portes.

**-DeathPanelController.cs** et **VictoryRetour.cs** s’assurent que les panneaux appropriés sont affichés et que les boutons fonctionnent correctement pour permettre au joueur de relancer le niveau ou de retourner au menu principal.

**6. Navigation et Chargement des Scènes**

**MenuController.cs** centralise la navigation entre les différentes scènes du jeu. En utilisant le système de gestion des scènes de Unity, ce script permet au joueur de sélectionner et de charger les niveaux faciles, intermédiaire et difficile via l’interface du menu principal.

1. Difficultés rencontrées

Difficulté à faire fonctionner les animations du player et à gérer certains bugs

Etant donné que nous avions moins vu cela en cours, nous avons pris du temps à faire fonctionner les animations de notre player et à prendre en main son fonctionnement. Pendant longtemps, les animations s’exécutaient de façon aléatoire et nous ne comprenions pas pourquoi. De plus, lorsqu’on utilisait beaucoup les commandes saut, gauche et droite, cela arrivait que le jeu se bug et qu’il faille sortir du mode jeu et recommencer la partie pour pouvoir débugger le personnage.

Difficulté à faire comprendre au player quelles étaient les surfaces au sol et les surfaces verticales correspondant à des blocs à pousser pour le comportement des animations

Nous avons eu du mal à faire différencier les contacts verticaux (avec des murs) et horizontaux (avec des sols) pour les animations de notre player. C’est-à-dire que lorsque notre player rentrait en contact avec un mur il se bugait en croyant que c’était le sol. Il a fallu faire comprendre, dans le script du player, qu’il fallait adopter un comportement différent lorsque le player était sur le dessus d’une surface horizontal et lorsqu’il touchait le côté vertical d’une surface.

Difficulté à coordonner l’ouverture de la porte et la récupération de la clé dans le code

Nous avons parfois eu du mal à nous y retrouver lorsqu’il y avait plusieurs déclencheurs à effectuer pour une même action. Par exemple lorsqu’on prend la clé, la faire disparaitre mais aussi ouvrir la porte. Puis lorsqu’on traverse la porte, déclencher le bon menu et bien configurer les boutons cliquables, etc. Il fallait parfois bien structurer notre pensée pour que toutes les actions se coordonnent sans souci, ne pas faire d’erreur dans les tags et dans les layer, attribuer les bonnes propriétés aux éléments, etc.

1. Pour aller plus loin

Pour aller plus loin, nous aurions beaucoup d’idées d’améliorations supplémentaires :

-Travailler sur la diversité des fonds et des décors de jeu. Rajouter des éléments décoratifs comme des panneaux, des pioches, des pierres précieuses etc.

-Rajouter un menu afin de choisir son avatar et donner un pseudo

-Rajouter plus de fonctions au player comme la possibilité de se baisser ou bien de courir plus vite.

-Complexifier les parcours de jeu avec plus d’obstacles, d’ennemis etc.

-Rajouter une bande son avec des bruits de pas, grognements pour le monstre, musique de fond, bruitages des animations, etc.

-Rajouter un mode multijoueur s’inspirant du jeu déjà existant : fire boy and water girl.

Une fois que nous avons bien pris en main la configuration de jeux 2D sur Unity, il devient très plaisant d’apporter de multiples améliorations et de prendre plaisir à perfectionner le design et les fonctions des niveaux.