420-C61-IN

PROJET SYNTHÈSE

Fanid, Amine 1370770

Baril, Henrick 2195213

GEEK’S LEGACY

Jeu de plate-forme 2D, inspiré du jeu « Terraria »

Rapport de mi-mandat

Amine Fanid:

Au cours du Sprint 3, j'ai été confronté à un défi significatif lié à la distribution aléatoire des biomes. Après plusieurs tests, j'ai réalisé que l'algorithme que j'avais développé pour cette tâche n'était pas véritablement aléatoire, et qu'en plus, il souffrait d'une instabilité notable. Bien que j'aie initialement pensé que l'algorithme fonctionnait correctement, il est rapidement devenu évident que des ajustements étaient nécessaires. Cela a demandé un investissement de temps considérable pendant le Sprint 3, même si j'avais initialement prévu de finaliser cet algorithme dès le Sprint 2. Néanmoins, ces efforts en valaient la peine, car la distribution des biomes dans notre monde est désormais authentiquement aléatoire et semble stable pour le moment.

Après avoir résolu ce problème, j'ai dirigé mes efforts vers le développement du script C# intitulé ClickDetection.cs, permettant de supprimer les "Tiles" lorsque celles-ci sont cliquées. Ensuite, j'ai entamé le travail sur les différents objets du jeu, tels que les blocs de terre, de fer, de bois, ainsi que les épées, entre autres. Ainsi, lorsque je clique sur une "Tile" de terre, un bloc de terre apparaît comme prévu.

Le code pour l'algorithme de distribution aléatoire des biomes se trouve dans le fichier ProceduralGeneration.cs. La fonction BiomeNumAssignation() a été cruciale pour résoudre les problèmes rencontrés, et j'ai bénéficié de l'aide de ChatGPT pour sa mise au point. Quant aux objets du jeu, leur code est réparti dans les fichiers Item.cs, Axe.cs, Pickaxe.cs, Sword.cs et Materials.cs.

Henrick Baril:

La conception et l'implémentation d'une Finite State Machine (FSM) ont constitué l'un des principaux défis de mon travail. L'objectif était de généraliser les états possibles pour mes ennemis, afin de leur permettre d'adopter des comportements plus réalistes. Bien que la création de la FSM elle-même n'ait pas été particulièrement difficile, son implémentation et sa compréhension ont présenté des défis significatifs.

J'ai suivi les recommandations d'un site web spécialisé pour élaborer la FSM, dont le lien est fourni à la fin de ce document. L'aspect le plus significatif de mon travail a été l'implémentation concrète de la FSM. Mon objectif était d'assurer une transition fluide entre les états pour les ennemis, de manière qu'ils puissent réagir réalistiquement aux actions du joueur.

Pour ce faire, j'ai développé plusieurs scripts, notamment FSM.cs et State.cs pour la gestion de la FSM, Slime.cs pour la mise en œuvre de la FSM avec le premier ennemi, TileDetection.cs pour la détection des possibilités de saut lorsque l'ennemi est confronté à un obstacle (pour l'escalade), et Inventory.cs pour gérer les animations liées à l'inventaire dans l'interface utilisateur.

Bien que ma contribution ne puisse pas être définie par des lignes précises, les scripts FSM.cs, State.cs et Slime.cs offrent des exemples concrets de ma réalisation en matière de FSM.

En équipe:

La collaboration au sein de notre équipe a été limitée, chaque membre étant principalement concentré sur l'achèvement de ses tâches assignées. Toutefois, une exception a été faite pour la conception des sprites représentant les équipements et les objets du jeu, où nous avons travaillé ensemble. En cas de besoin, nous nous sommes mutuellement apporté notre soutien.

Dans le prochain cycle de développement, une collaboration plus étroite sera nécessaire, étant donné que les fonctionnalités que nous avons développées devront être intégrées de manière cohérente. Nous anticipons que cela constituera notre plus grand défi à venir, mais nous restons confiants quant à notre capacité à le relever.

À l'heure actuelle, nous estimons avoir accompli environ 50% de notre travail total, avec une marge d'erreur d'environ 10%. Cependant, cette estimation peut varier en fonction du temps nécessaire pour implémenter certaines fonctionnalités facultatives.

Bibliographie:

- https://faramira.com/finite-state-machine-using-csharp-delegates-in-unity/ (Consulté le 18 avril 2024)