**MAXIME DESROCHERS, NOÉ BOUSQUET**

**PROJET SYNTHÈSE**

**420-C61-VM gr. 00001**

## **Rapport Final**

Travail présenté à

Pierre-Paul Monty

Cégep du Vieux Montréal

Le 23 Mai 2024

**Présentation générale du projet**

Notre projet est un jeu “puzzle platformer” ou le joueur doit se déplacer dans une quatrième dimension physique à l’aide d’une barre de défilement. Le jeu est composé de 6 niveaux augmentant en difficulté, ainsi qu’une librairie générique de “4D raymarching”.

**Résumé du développement pendant la session**

Au début du projet, nous voullions ajouter plusieurs objets que le joueur pourrait interagir avec pour résoudre les différents niveaux. Cependant, avec le développement du Raymarching, nous nous sommes rendu compte des limitations de ce qu’on pouvait faire avec nos ressources et le temps qu’il restait, nous avons donc dû laisser tomber les objets interagissables pour un seul objet modulable, qui permet de modifier l’état d’un objet 4D. Néanmoins, nous avons pu créer des niveaux intéressants avec cette mécanique et celle de la 4D en elle même.

**Fonctionnalités**

4DCamera : Fonctionnelle avec limitation (nécessite au moins un objet 4d

dans la scène)

4DShape : parfaitement Fonctionnelle

Mouvement du joueur : Parfaitement Fonctionnelle

Interaction avec les objets : Parfaitement Fonctionnelle

Collision du joueur avec les objets 4D: Parfaitement Fonctionnelle

Transition entre les scènes : Parfaitement Fonctionnelle

Menu et UI : Parfaitement Fonctionnelle

Gravité : Partiellement Fonctionnelle

Sauvegarde de savefile : Parfaitement Fonctionnelle

Ombres : Fonctionnelle avec Limitation (on ne peux pas toggle l’ombre par objet)

**Élément techniques**

- interface graphique utilisateurs:

Nous avons enlevé l’écran de démarrage car ce n’était pas très pratique ni utile.

Nous avons enlevé l’écran d’options, car il n’y a pas d’élément du jeu a changer.

Nous avons ajouté un écran de démo, permettant de visualiser les éléments 4d du jeu.

L’écran de sélection de niveau se fait avec des boutons simples.

- données persistantes:

Nous utilisons le format binaire pour stocker nos données, cette façon est plus efficace et encrypte nos données afin qu'elles ne puissent pas être modifiées facilement par l’utilisateur. Ces méthodes sont dans le script SaveSystem.cs et ensuite utilisées avec un SaveController dans le menu pour créer une savefile (write) et pour afficher les saves présentes (read), nous sauvegardons aussi le niveau le plus loin que le joueur a été pour débloquer les suivants dans le menu.

- structures de donnés:

*Struct*, qu’on utilise pour stocker les différentes shapes et leurs attributs dans RaymarchShader.shader

*Vecteurs*, utilisés presque partout pour les positions 3d et 4d, dans RaymarchCam.cs par exemple.

*Buffers*, utilisés pour stocker les shapes sous forme de shader, utilisé dans RaymarchShader dans une liste de Buffer.

*PlayerData*, créé dans PlayerData.cs et utilisé dans SaveSystem.cs pour stocker les différents joueurs.

- patrons de conceptions:

*Facade:* Le NameEntryController est une façade du SaveSystem car il permet de sauvegarder les méthodes simplement.

*Strategy:* Le patron de conception Strategy est utilisé par les différentes fonctions de distances signés dans DistanceFunctions.cs

*State:* le 4DObject utilise le patron de conception State car il change d’état en changeant de forme géométrique et d’opération booléenne

Bibliothèque: Notre libraire générique 4DRaymarching contient les prefabs pour la caméra raymarching 4d, le joueur 4d. Tout cela se trouve dans le dossier 4DRaymarching et est utilisé dans chacun des niveaux du jeu.

Algorithme: Notre algorithme est un algorithme de rendu, en utilisant un shader pour changer ce que voit la caméra. Le script est RaymarchCam, placé sur la caméra et utilise le shader RaymarchShader.

Mathématique: Le calcul de la vision des objets raymarched, situé dans le Shader RaymarchShader

Regex: Nous utilisons une expression régulière pour l'entrée du nom du joueur pour qu’il soit alphanumérique et contient de 1 à 16 caractères, dans NameEntryController.cs.

**Améliorations possibles**

Un problème apparent est le fait que le raymarching nécessite un ordinateur performant, nous pourrions essayer d’optimiser davantage l’algorithme pour permettre des scènes plus complexes avec plus d’objets.

**Auto évaluation Noé**

Je pense que ce projet montre à quel point mes compétences de programmation ont augmenté depuis le début de la technique. Je suis content de ce qu’on a réussi à créer. Développer une technique de rendu tel que le raymarching m’a appris beaucoup de choses sur Unity et apprendre à organiser un projet de 0 m’a beaucoup plu. Pour ce qui est de la valeur de mon travail, considérant le niveau d’effort et du temps passé sur le projet, je me donne une note de 90%.

**Auto évaluation Max**

Je trouve que ce projet a démontré que mon niveau de logique et de programmation a beaucoup augmenté depuis mon arrivée au CVM.

J'avais hâte d'apprendre à créer des jeux sur Unity, et je suis vraiment fier de pouvoir dire que je maîtrise maintenant la plateforme. Je ne suis pas un professionnel, mais je comprends beaucoup plus qu'avant. De plus, j'ai été surpris de réussir à créer un jeu utilisant la 4D. J'ai atteint mes objectifs et j'espère réaliser d'autres projets aussi difficiles à l'avenir.

Développer l’UI du jeu m'a beaucoup appris sur Unity. En pensant et en créant des niveaux en 4D, j'ai pu voir ce que je pouvais faire avec mon imagination et ma logique. De plus, commencer sans sujet précis, où c'était à nous de proposer une idée de projet, était enrichissant.

Je me mets 90%

**Auto évaluation d’équipe**

Noé: Je pense que la façon dont on a séparé le travail, en utilisant les connaissances, forces et faiblesses de chacuns a très bien marché, nous divisons notre travail mais nous gardions toujours un oeil sur ce que l’autre faisait, en proposant des idées ou en donnant des commentaires constructifs.

Maxime : Je trouve que la répartition de nos tâches a été très efficace. Noé connaissait beaucoup plus que moi la 4D et moi je connaissais bien Unity et son interface. Nous avons travaillé de manière autonome tout en restant en communication.