Rapport de projet

Hellvania

Projet Synthèse

420-C61-IN

Travail remis au

professeur Jean-Christophe Demers

Par Caroline Emond Serret

Cégep du Vieux-Montréal

26 mai 2021

Présentation générale

La version finale du projet est intitulé Hellvania, un jeu *platformer* 2D développé avec Unity. Le jeu contient un menu principal, un prologue en guise de mise en situation, ainsi qu'un niveau avec 3 ennemis de base et 1 *final boss.*

Résumé du développement

Le début du projet a été difficile puisque l'idée de départ était finalement trop complexe pour notre niveau d'expérience avec Unity/animation, ainsi que le temps disponible pour la réalisation du projet. Le temps passé pour la création d'animations en huit directions en vue isométrique est rapidement devenu insoutenable et la décision de modifier le jeu pour un style *platformer* a été prise. Une grande partie des objectifs dans la planification sont donc devenus obsolètes. La fin du projet fut également difficile, où un changement a priori banal crée un problème en cascade et avec l'émergence de *bugs* uniquement dans le build entre autres.

Sources utilisées

Les références au code utilisé se retrouvent également en commentaire au début des scripts.

1) Code reçu de Maël Perreault et code réalisé dans le cadre du cours C63 – Développement de jeux vidéos
2) Brackeys. 20 mai 2018. « How to Fade Between Scenes in Unity ». https://www.youtube.com/watch?v=Oadq-IrOazg
3) Brackeys. 23 juillet 2017. « How to make a Dialogue System in Unity ». https://www.youtube.com/watch?v=_nRzoTzeyxU&t=254s
4) GameDevBeginner. 4 avril 2020. « Coroutines in Unity (how and when to use them) ». https://gamedevbeginner.com/coroutines-in-unity-when-and-how-to-use-them/#how_to_write_a_coroutine
5) GeeksforGeeks. 28 février 2019. « C# Dictionary with examples ». https://www.geeksforgeeks.org/c-sharp-dictionary-with-examples/
6) GitHub Gist. 16 avril 2016. « quill18 / SimplePools.cs ». https://gist.github.com/quill18/5a7cfffae68892621267
7) Kaliko. 29 septembre 2013. « C# test if object or type implements interface ». https://kaliko.com/blog/c-test-if-object-or-type-implements/
8) Press Start. 1er mai 2019. « Parallax Effect in Unity ». https://pressstart.vip/tutorials/2019/05/1/94/parallax-effect-in-unity.html
9) Ray Wenderlich. 23 novembre 2016. « Object Pooling in Unity ».https://www.raywenderlich.com/847-object-pooling-in-unity
10) Unity Documentation. « JSON Serialization ». https://docs.unity3d.com/Manual/JSONSerialization.html

11) Unity Documentation. « Application.persistentDataPath ». https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html
12) WeeklyHow. 10 décembre 2020. « Unity tutorials : How to Make a Health Bar in Unity ». https://weeklyhow.com/how-to-make-a-health-bar-in-unity/
13) Forum Unity Answers. « OnMouseOver UI Button c# ». https://answers.unity.com/questions/1199251/onmouseover-ui-button-c.html

Stratégies explorées

L'implémentation d'un système de Factory était souhaité. Toutefois, le PrefabManager accomplissait déjà sensiblement la même tâche. L'idée a donc été abandonnée dû au temps de refactorisation que cela amènerait.

Du Object Pooling a été implémenté pour les objets légers et facilement réutilisables (projectiles, effets spéciaux). Souvent utilisé avec des Factories, des fonctions de création de prefabs spécifiquement pour le Object Pooling ont été ajoutées dans PrefabManager.

Des évènements/delegate (observers) ont été utilisés pour la détection de changement dans le niveau de vie et mana par exemple.

Fonctionnalités

1. GameManager: parfaitement fonctionnel
2. PrefabManager: parfaitement fonctionnel
3. MovementController: parfaitement fonctionnel
4. LevelManager: parfaitement fonctionnel
5. PoolManager: parfaitement fonctionnel
6. SaveLoadManager: semi-fonctionnel, problème avec InventoryManager, qté d'items sauvegardés ne correspond pas à la quantité réelle
7. InventoryManager: semi-fonctionnel, problème au niveau de la reconnaissance des items et/ou qté en inventaire
8. DialogueManager: parfaitement fonctionnel
9. SoundManager: parfaitement fonctionnel
10. Potions vie/mana: semi-fonctionnel, relié au problème d'InventoryManager
11. Main Menu: parfaitement fonctionnel
12. QuestManager: abandonné
13. SanityManager: abandonné
14. TriggerManager: abandonné
15. UIManager: abandonné

Améliorations possibles

- Un component séparé pour gérer les inputs du joueur, tel un InputManager, aurait été souhaitable au lieu de gérer chaque touche dans le Update du Player.
- Utiliser un effet miroir pour changer la direction des sprites gauche/droite, cela éviterait d'avoir chaque image/frame et chaque animation en double.
- Une structure de données plus étanche concernant les items d'inventaire, avec des constantes read-only par exemple

Auto-évaluation

Apprenant C# et Unity cette session-ci, ce projet reflètait ironiquement le contexte et pratique de chaque cours du programme intensif: apprendre le plus possible, le plus rapidement possible. La différence est que la ligne d'arrivée, soit le produit final, est bien plus nébuleuse et variable que celle d'un projet définit par le professeur. Je dirais que ce type de projet exacerbe et met rapidement en évidence ce qu'on nous répète depuis le début du programme: réfléchir longuement à ce que l'on tente d'accomplir avant de lancer son IDE. Est-ce une bonne idée d'établir telle ou telle structure? Est-ce qu'une structure plus optimale/sécuritaire serait souhaitable ou est-ce qu'une approche plus cowboy « tant que ça fonctionne » peut être satisfaisante dans certains cas? Idéalement, une structure optimale est ce qu'on souhaite, mais il faut également peser le pour et le contre s'il y a des priorités plus importantes ou pressantes. Bref, l'importance de s'interroger sur la tâche et son fonctionnement était de mise. Certaines structures ou design patterns peuvent être excellents dans une situation, mais nuire ou compléxifier la tâche inutilement dans une autre. Cela nous ramène bien entendu aux cours de gestion de projet, où il est important de savoir où, quand et quoi couper, réévaluant constamment notre cheminement et le produit final selon les avancées afin de pouvoir s'adapter.

Personnellement, ma satisfaction envers le projet est ambivalente. D'un côté, c'est incroyable de réaliser quelque chose de cette ampleur après seulement 16 mois, ayant passer la première session complète avec seulement un terminal comme *output*. C'est énormément gratifiant. D'un autre côté, c'est également difficile. Étant perfectionniste et ayant une vision claire et détaillée de ce qu'aurait pu être le jeu avec un peu plus de temps et d'expérience avec Unity, je suis très déçue de remettre quelque chose d'incomplet. Encore plus lorsque certains aspects sont à un détail près. J'alterne donc entre fierté et déception/honte. J'imagine toutefois que c'est sans doute commun en tant que débutante.

Somme toute, j'opterais pour 75% comme note d'évaluation. J'aurais aimé réaliser plus rapidement le cul-de-sac dans lequel j'étais avec la première version du projet, voire même avoir penser à une option plan B afin de ne pas être prise au dépourvu.