Émile Gélinas  
Veille technologique  
420-1SH-SW

Le développement de jeux vidéo

Travail présenté à M. Nicolas Bourré

Collège Shawinigan  
2023-06-09

**Table des matières:**

[Introduction 1](#_Toc137195037)

[L’origine de Godot 1](#_Toc137195038)

[Différences entre Unity/Unreal 1](#_Toc137195039)

[Méthodologie 2](#_Toc137195040)

[Le Résultat 6](#_Toc137195041)

[Conclusion 6](#_Toc137195042)

[Médiagraphie i](#_Toc137195043)

# Introduction

L’un des loisirs le plus populaire de ces dernières années et le jeu vidéo. Avec autant de personnes s’intéressant à ce média grandissant, il y aura aussi beaucoup de personnes intéressées à participer et créer un jeu vidéo. Heureusement pour eux, il y a beaucoup d’outils pour les aider à accomplir leurs objectifs. Un de ces outil le plus simple d’utilisation est le logiciel nommé « Godot ». Cependant, avant de se lancer dans cette aventure sans connaissances sur le sujet, il est important de connaitre la méthodologie de base de ce cicle de développement : connaitre son logiciel, les fondations d’un jeu fonctionnel et les entités nécessaires et réutilisable pour la construction d’une preuve de concept d’un jeu vidéo

# Qu'est-ce que Godot?

Godot est un moteur de jeu, c’est-à-dire un logiciel de création de jeux vidéo, initialement développé par Juan Linietsky et Ariel Manzur. Le logiciel fut par la suite rendu publique en tant que logiciel *open source*, c’est-à-dire que tout programmeurs volontaires peut travailler sur ce logiciel pour le rendre meilleur, ce qui ne le relie à aucune compagnie et assure qu’il reste gratuit pour le grand public. Certaines de ses caractéristiques les plus connue est l’utilisation d’un langage informatique créé pour le logiciel, appelé *GDscript*, et de l’utilisation d’un system de *nodes* qui permettent d’assembler une entité à partir de plusieurs entités préexistantes de manière simple et efficace.

# Différences entre Unity/Unreal

Contrairement à d’autres moteurs de jeux comme les célèbres Unity et Unreal Engine, Godot est un moteur très léger pouvant être utilisé sur des ordinateurs moins performants. Il a certes moins de fonctionnalités, mais il est parfait pour des projets de plus petite envergure, pour des jeux en 2D ou tout simplement pour les développeurs débutants voulant apprendre les bases du jeu vidéo sur un moteur moins compliqué. En effet, Godot à une emphase sur la création de jeux 2D, ce qui est parfait pour débuter dans le milieu, car il y a moins de dimensions à gérer et aucune gestion de lumière et d’ombrage. Les nouvelles versions de Godot à venir essayeront de rendre la création de jeux 3D aussi compétitive que d’autres moteurs comme Unity, même si ça restera incomparable avec Unreal qui est spécialisé dans les graphismes réalistes, ce qui n’est pas nécessairement pertinent pour des studios indépendants qui veulent faire de jeux de plus petite envergure.

# Méthodologie

Le projet à comme base la série de tutoriels vidéo de [HeartBeast](https://www.youtube.com/@uheartbeast) nommé : [Godot Action RPG Series](https://www.youtube.com/playlist?list=PL9FzW-m48fn2SlrW0KoLT4n5egNdX-W9a). Le but originel de ce projet est de faire une preuve de concept d’un potentiel jeu vidéo, or le résultat final de ce tutoriel manque beaucoup d’éléments de complexité qui donnerais de l’intérêt au produit final. Il fut donc décider d’utiliser cette base et de construire de nouvelles fonctionnalités par-dessus celle-ci.

Cette base consiste des éléments suivants :

1. Une simple carte monde en vue du dessus, des buissons et collines possèdent des collisions et des herbes pouvant être coupé
2. Une interface utilisateur pour les points de vie
3. Un personnage jouable avec des mouvements dans 8 directions, une attaque à l’épée simple, une roulade, des animations et des *hitboxe*
4. Un ennemi qui ne fait que suivre le joueur pour le toucher et lui faire des dommages

La carte utilise une *tilemap* pour créer le terrain comme si on peinturait le sol et le sable sur une toile.

Une image contenant capture d’écran, texte, Logiciel multimédia, Logiciel de graphisme

Description générée automatiquement

Le personnage jouable est composé, entre autres, d’un *KinematicBody2D* qui lui permet d’interagir physiquement avec d’autres éléments comme des murs qui bloque ses déplacements, et d’un *Sprite* animé par un *AnimationPlayer* et un *AnimationTree* qui permet de gérer les différentes directions des animations.

Une image contenant capture d’écran, texte, Logiciel multimédia, Logiciel de graphisme

Description générée automatiquement

Cependant, les ennemies utilisent un *AnimatedSprite*, qui est moins modulable, car ils n’ont pas besoins d’animations complexes ou d’interactions avec des scripts. Une image contenant capture d’écran, texte, Logiciel multimédia, Logiciel de graphisme

Description générée automatiquement

La *Hitbox* (zone pour faire des dommages) et la *Hurtbox* (zone pour recevoir des dommages) sont des *nodes* séparés pour ainsi réutiliser du code spécifique à ce genre d’interaction.

Les nouvelles fonctionnalités implémentées sont :

1. Possibilité de changé de carte, pour créer de nouveaux niveaux et pouvoir y accéder
2. Des ennemies avec une meilleur intelligence artificiel, pour que le jeu soit plus difficile et complexe
3. Des attaques spéciales, pour diversifier la mécanique de combat
4. Faire en sorte que les ennemis vaincus dans une carte restent morts quand le joueur revient dans cette carte, pour éviter de réinitialiser la progression du joueur

Le changement de carte est effectué pas une zone de collision qui remplace la carte actuel avec celle désirer. Cette zone de collision est assignée le nom de la carte désirer et l’emplacement où le joueur dois être positionné quand il sera généréUne image contenant texte, capture d’écran, diagramme, Logiciel de graphisme

Description générée automatiquement

Les ennemis ont un ID, ce qui leurs permettent, quand ils sont vaincus, d’appeler une fonction globale et de lui donner leur ID et le nom de la carte qu’ils sont situé pour que cette fonction les enregistre dans un dictionnaire d’entités vaincus. Quand un changement de carte va les regénérer, ils vont regarder s’ils sont dans le dictionnaire. Si oui, ils sont immédiatement supprimés.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

L’attaque spécial utilise uniquement des *Area2D*, qui sont des zones permettant de détecter une collision. L’utilisation d’un *KinematicBody2D* n’est pas pertinent, car l’utilisation de collisions physiques, comme utilisé pour arrêter les mouvements du joueur face à un mur, n’est pas nécessaire. Simplement l’effacé à la collision d’un mur est suffisant. L’utilisation de deux zones de collision est utile pour agrandir celle qui interagie avec les ennemies et rétrécir celle qui interagie avec l’environnement, pour donner plus de chance au joueur de toucher un ennemi dans des zones plus restreintes. Une image contenant texte, capture d’écran, Logiciel de graphisme, diagramme

Description générée automatiquement

L’intelligence artificiel des ennemies a été amélioré en leur permettant d’inverser leur direction si le joueur est trop proche d’eux et d’utiliser un projectile reprenant la même logique que l’attaque spécial du joueur.

# Le Résultat

Une image contenant capture d’écran, Rectangle, diagramme, pixel

Description générée automatiquement

Au final, nous avons une base de jeu parfaitement fonctionnel, même si imparfaite. Le personnage jouable et l’interface utilisateur sont directement placé sur la carte, ce qui veux dire qu’à chaque changement de carte, le joueur et son interface dois être effacé puis replacé sur la nouvelle carte. Cela force la réutilisation de *nodes* de manière redondante, ainsi que la gestion des statistiques du joueur dans un script global pour ne rien réinitialiser dans le processus. Il y a encore beaucoup de code peu réutilisable qui pourront définitivement être optimiser dans un autre jeu qui pourra commencer sur de meilleures bases

# Conclusion

Pour conclure, la création d’un jeu vidéo est beaucoup plus vaste qu’une simple démo peut représenter. La simple initialisation du personnage jouable et d’un seul ennemi demande énormément de différentes variables qui interagisses entre elles, et mal optimiser une mécanique développer au tout début du cicle de création peut être la cause d’énormément de futurs maux de tête. C’est à se demander à quel point ce processus, déjà complexe, peut l’être encore plus si on ajoute une 3e dimension ou si on utilise un moteur de jeu encore plus demandant comme Unreal.

# Médiagraphie

* HeartBeast. (2020, March 12). *Make an Action RPG in Godot 3.2* [Video]. YouTube. <https://www.youtube.com/watch?v=mAbG8Oi-SvQ>
* *Godot Docs – 4.0 branch*. (n.d.). Godot Engine Documentation. <https://docs.godotengine.org/en/stable/>
* Wikipedia contributors. (2023). Godot (game engine). *Wikipedia*. <https://en.wikipedia.org/wiki/Godot_(game_engine)>
* How to make enemy follow until a certain length from player. (2019, February 8). Godot Engine - Q&A. <https://ask.godotengine.org/40465/how-to-make-enemy-follow-until-a-certain-length-from-player>
* Arith metic. (2021, March 7). Shooting a Bullet - Godot Tutorial [Video]. YouTube. <https://www.youtube.com/watch?v=2G41KECXXn4>