UFR Sciences et technique
Université Marie et Louis PASTEUR
Projet
L3 Informatique

Jeu de plate-formes avec génération procédurale

Rapport de projet

Kilian Jelic
Laura Jacqueson
Théo Pariney

avril 2025

Table des matières

Table des figures

1 Introduction

Dans le cadre de notre projet semestriel en L3 Informatique à l'Université Marie et Louis Pasteur, nous avons travaillé sur le développement d'un jeu de plate-formes avec génération procédurale en C++.

Les jeux de plate-formes sont un genre qui repose sur le contrôle d'un personnage avec des mécaniques comme les sauts et les obstacles, l'objectif étant généralement de rejoindre une sortie pour terminer le niveau. L'ajout de la génération procédurale introduit une caractéristique supplémentaire : plutôt que de concevoir chaque niveau à la main, un algorithme est chargé de créer les niveaux du jeu, permettant ainsi une expérience unique à chaque partie.

L'objectif principal de ce projet était de concevoir un jeu de plate-formes dont les niveaux seraient générés de manière procédurale, offrant ainsi une rejouabilité infinie. En parallèle, nous avons dû concevoir un jeu intégrant plusieurs mécaniques de jeu, incluant des déplacements, des sauts, un dash, ainsi qu'un objectif centré sur la récolte d'objets, tout en assurant la gestion des collisions et la physique du jeu.

Dans ce rapport, nous commencerons par une présentation générale du jeu, en exposant les objectifs du joueur et les scènes du jeu. Ensuite, nous détaillerons la conception des blocs et entités qui composent le monde et leur gestion dynamique. Par la suite, nous aborderons la génération procédurale, en expliquant la méthode utilisée ainsi que les détails d'implémentation. Nous discuterons ensuite des contrôles du personnage et des différentes mécaniques de jeu, telles que le saut, le dash, etc. La partie suivante sera consacrée à la physique, notamment la gestion des collisions et des propriétés physiques comme la résistance de l'air. Enfin, nous reviendrons sur l'organisation du travail, la répartition des tâches et les outils utilisés pour mener à bien ce projet. Le rapport se conclura par un bilan du projet et les améliorations possibles.

2 Présentation du jeu

2.1 Présentation générale

Ce projet consiste à développer un jeu de plate-forme en 2D intégrant un système de génération procédurale de niveau. L'objectif est d'offrir une expérience où chaque partie est unique : au lieu d'avoir des niveaux prédéfinis, ceux-ci sont générés aléatoirement à chaque nouvelle partie. Contrairement aux jeux de plate-forme classiques où l'on peut apprendre les niveaux par cœur, ici, aucun parcours ne peut être rejoué à l'identique. Cette approche permet non seulement une rejouabilité infinie, mais aussi un défi constant, obligeant le joueur à s'adapter à chaque nouvelle configuration de niveau.

Le joueur incarne un petit robot, dont la mission principale est de récolter un maximum d'écrous avant d'atteindre la sortie du niveau. Ces écrous sont disposés aléatoirement dans l'environnement, incitant le joueur à explorer avant de pouvoir terminer la partie.

Pour évoluer dans le monde, le joueur dispose de plusieurs mécaniques lui permettant de se déplacer librement et d'interagir avec son environnement :

- **déplacement** : le personnage peut se déplacer latéralement, à gauche et à droite, pour parcourir le niveau.
- **saut** : le personnage peut sauter pour franchir des obstacles ou atteindre des plateformes situées en hauteur.
- **double saut** : après un premier saut, lorsqu'il est encore en l'air, le personnage peut effectuer un second saut lui permettant d'atteindre des zones plus élevées.
- **dash** : le personnage dispose également d'un dash, une impulsion rapide dans une direction (gauche ou droite), utile pour traverser de grands espaces vides ou esquiver des obstacles.

Ces mécaniques offrent une grande liberté de mouvement, permettant aux joueurs d'adopter différentes stratégies. Certains privilégieront une approche prudente, optimisant chaque saut pour éviter la mort, tandis que d'autres tenteront des enchaînements rapides et fluides pour terminer le niveau efficacement.

Comme mentionné précédemment, tous les niveaux sont générés de manière procédurale, cela signifie qu'à chaque lancement de partie c'est un algorithme qui se charge de créer un environnement en plaçant des blocs, plateformes, échelles, écrous, piques et une sortie. Cette méthode permet de générer de manière presque infinie des mondes différents.

Ces différent niveaux sont composés de salles générées de manière contiguës, et les salles sont reliées entre elles par des échelles et des plateformes, de façon à ce que chaque niveau soit possible, en partant du début jusqu'à la fin. Des blocs et plateformes de glace et de slime sont également générés, et le joueur doit donc faire attention aux changements que cela implique sur ses déplacements.

2.2 Les objectifs du joueur

2.2.1 Les écrous

2.2.2 La sortie

2.3 Les scènes

3 Les blocks et entités du monde

3.1 La définition dynamique des blocks

3.2 Le stockage des blocks

4 La génération procédurale

4.1 Aléatoire et graine de génération

À des fins de débuggage, la possibilité de pouvoir générer le même monde plusieurs fois est importante. Pour cela, les niveaux sont générés à l'aide d'un générateur de nombres pseudo-aléatoire. Ce générateur

peut utiliser un entier codé sur 64 bits, et générera la même séquence de nombres aléatoires tant que la même graine est utilisée.

Nous avons donc implémenté deux moyens de générer les mondes : avec ou sans graine. Si une graine est donnée au générateur, alors cette graine sera utilisée lors de la création du niveau. Sinon, une graine aléatoire est choisie, et est affichée dans la console. Cela permet, en cas de problème, de facilement noter et partager une graine, permettant de reproduire le problème plus facilement.

4.2 Étapes de génération

4.2.1 Génération des salles

Les salles sont générées de manière à ce qu'elles soient connectées entre elles, sans jamais se chevaucher. Pour cela, l'algorithme suivant est appliqué :

- Générer une salle de départ.
- Tant que le nombre de salles voulu n'est pas atteint :
  - Choisir aléatoirement la taille de la salle.
  - Choisir une direction (haut, bas, gauche ou droite)
  - Créer une nouvelle salle, placée à côté de la salle précédente, aléatoirement le long du mur correspondant à la direction choisie.
  - Si la salle chevauche une autre salle, la regénérer.
  - Si, au bout d'un certain nombre d'essais, la salle n'a pas réussi à se générer, réessayer en changeant la taille de la salle.
  - Si la salle n'arrive toujours pas à se générer, supprimer la salle ainsi que la salle précédente, et regénérer la salle précédente.

Certains cas spéciaux se présentent avec cet algorithme :

- **La salle de départ** : La salle de départ, étant la première salle, ne peut pas se générer à la suite d'une autre salle. Elle se génère donc simplement aux coordonnées (0, 0)
- **Retour en arrière** : Dans certains cas, une salle peut se retrouver encerclée par d'autre salles. Dans ce cas, aucune salle ne pourra se générer par la suite. Afin d'éviter ces situations, l' algorithme décide, si une salle n'arrive pas à se générer, de réessayer d'abord avec une salle de différente taille, car une salle plus petite peut éventuellement réussir à se générer. Mais, au bout d'un certain nombre d'essais, l'algorithme décide de retourner en arrière. Dans ce cas, la salle précédente est supprimée, et, en prenant un autre chemin, le générateur réussira à générer l'ensemble des salles.
- **Taille minimale d'entrée** : La génération de plateformes et de pièges pourrait rendre l'entrée dans une salle impossible dans des cas où l'entrée d'une salle serait trop petite. Afin d'éviter cela, lorsqu'elles se placent le long d'un mur, les coordonnées aléatoires ont été restreintes afin que l'entrée vers la salle suivante aie toujours une taille minimale. Cette taille a été définie à 2 blocs. Dans la figure 1, on peut voir que peu importe où se génère la salle, l'entrée de la salle verte vers la salle bleue aura toujours une taille d'au moins 2 blocs.

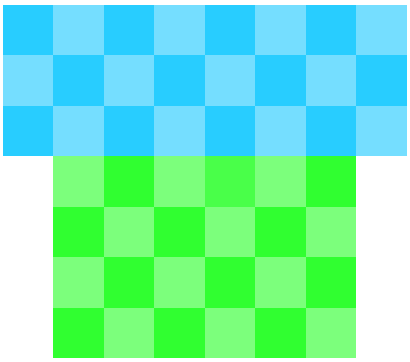

FIGURE 1 – Un exemple de placement de salle.
Une salle est représentée en vert.
Si la direction haut est choisie, une salle de taille 3x3 peut
se générer n'importe où dans la zone bleue.

8 Conclusion

8.1 Points d'amélioration

8.1.1 Ce qu'on aurait voulu ajouter

8.1.2 Wave Function Collapse

8.1.3 Pathfinding amélioré

8.1.4 La physique