Rapport P2 Qt

Damian Petroff, Sergiy Goloviatinski, Raphaël Margueron

16 janvier 2018

Table des matières

Chapitre 1

Introduction

Chapitre 2

Analyse

2.1 Spécifications

Chapitre 3

Conception

Chapitre 4

Développement

4.1 Génération des sols

Pour la gestion du terrain, nous avons décidé d'avoir deux types de sol, l'eau et l'herbe. Sur l'herbe, il est possible d'ajouter des bâtiments et sur l'eau non.

Pour pouvoir avoir des cartes auto-générée, nous avons décide d'utiliser un algorithme de bruit de Perlin. Nous n'allons pas aller dans les détails du fonctionnement du bruit de Perlin, dans ce rapport. Mais principe est le suivant, il permet d'avoir des nombres aléatoires qui sont proche les uns de autres en function d'une seed(initialisé au début) et de 1, 2 ou X dimensions (trois dans celui que nous avons utilisés, mais que 2 utilisés). Vu que les outputs de la function de bruit de Perlin sont les mêmes quand nous utilisons les mêmes inputs. Nous pouvons la considéré comme une fonction à plusieurs variables (qui sont les dimensions et la seed).

Les nombres aléatoires que nous récupérons de la function sont des nombres flottant entre 0 et 1. Leurs distribution sont sous la forme d'une distribution normal.

$$y = e^{-x^2}$$

Pour pouvoir générer la carte nous parcourons toutes les cases de la carte (selon x et y). Et "nous déplaçons sur le function de Perlin" d'un certain offset (qui n'est pas le même que les index des cases).

4.2 Gestion du bonheur

4.3 Routes et bâtiments

Pour la gestion des routes et des bâtiments nous sommes parti sur une approche plutôt simple qui est d'avoir divisé en trois classe qui chacune des rôles spécifique. Une qui décrit les caractéristiques propre à chaque bâtiments : les constantes -> ConstantBuilding. Une seconde classe qui décrit une bâtiment spécifique à la partie -> Building. Et une troisième qui sert de gestionnaires de building, 'classe tableau' -> Building Manager.

Constant Building : Constant Building contient une function de construction d'un tableau d'objet de 'soit même'. Chacun de ces objet représente un bâtiment avec toutes ces caractéristiques internes, par exemple : le prix de construction, la liste des bâtiments nécessaires pour sa construction ou encore sa catégorie dans l'affichage de l'HUD.

Liste de caractéristique de base de chaque bâtiment :

— Nom à l'affichage
— Catégorie
— Prix
— Largeur (au sol)
— Hauteur (au sol)
— Une liste de case ignoré (non-implémenté, pour avoir la possibilité d'avoir de bâtiment non rectangulaire)
— Type de pré-requis (Ou / Et)
— Une liste d'identifiant de bâtiment pré-requis.
— La sommes des pré-requis nécessaires.

Nous avons choisi d'avoir des objets au lieu d'une liste de constante, pour pouvoir avoir la possibilité d'y ajouter des fonctions pour ces objets ce qui nous a permis d'avoir un autre set de valeur basé sur les caractéristiques de base, voir ci-dessous. Cela nous permet ajuster facilement le jeu et d'avoir une sorte de progression linaire du jeu, voir ci-dessous.

Liste des caractéristique dérivé :

— Prix par secondes = Prix / 4
— Efficacité = $(Prix/10)^{1.4} + 10$ (arrondi au multiple de 25 le plus proche)
— Rayon d'action = $Log10((PrixParSecondes * Efficacit + 1))$ (Arrondi au multiple de 5 le plus proche)
— Poids du bâtiment dans les pré-requis = Prix / 10

On remarque donc que le prix du bâtiment influe directement sur tous les caractéristique qui définisse si le bâtiment est performant ou non.

Fonctionnement des pré-requis : Les pré-requis sont un aspect que nous considérons comme l'élément primordiale permettant au jeu d'avoir une progression. Le principe est le suivant.

Pour qu'un bâtiment soit accessible à la construction il faut qu'il respect les règles suivantes si son type de pré-requis est 'ou' : La somme des poids de tous les bâtiments qui sont actuellement posé qui sont dans la liste des bâtiments en pré-requis du bâtiment désiré doit être supérieur ou égal à la somme des pré-requis nécessaires. Exemple : Pour poser un Laboratoire médical : Il faut au moins une somme de 6. La liste des bâtiments pré-requis et des poids pour le laboratoire est la suivante :

— Clinique : 1
— Hôpital : 5

Il y a donc au moins trois possibilités pour arrivé à remplir cette demande :

— 6x Clinique
— 1x Clinique et 1x Hôpital (Meilleur solution, au niveau du prix par seconde de le efficacité sur un grand rayon)
— 2x Hôpital

Pour les pré-requis en 'et' : Il faut en plus que pour les pré-requis en 'ou', avoir au moins un bâtiment de chaque type de la liste des bâtiments pré-requis. Exemple : Pour poser une Tour Eiffel : Il faut au moins une somme de 11.5 La liste des bâtiments pré-requis et des poids pour le laboratoire est la suivante :

— Hôpital : 5
— Caserne de pompier : 1.5
— Quartiers généraux : 5

Il y a donc au moins une possibilité pour arrivé à remplir cette demande, qui est d'avoir une bâtiment de chaque. A noter que dans ce cas, la somme de pré-requis n'influence pas pour ce bâtiment, mais on pourrai imaginer qu'un bâtiment aille une somme plus grande que la somme de chaque bâtiment pré-requis et la la somme aurai une influence.

Building : La classe building décrit un bâtiment posé dans une partie. Il est décris les caractéristiques suivantes :

— Un identifiant unique
— Un identifiant du type de bâtiment (de ConstantBuilding)
— Une position X/Y
— Un angle de rotation (De quel côté est posé le bâtiment, non-implémenté)
— Une population

On peut donc noter que cet objet est rendu assez 'léger' car la plus part de ce qui le décrit est géré par l'identifiant constant building. Nous pouvons donc avoir beaucoup de bâtiment et avoir une emprunte mémoire peu conséquente.

Building Manager : Cette classe permet de gérer une liste de bâtiment de la classe Building. On peut exécuté les actions suivante :

— Ajouter un bâtiments (qui est ajoutable selon les pré-requis)
— Supprimer des bâtiments
— Récupérer la population total de la ville.
— Récupérer le bonheur moyen de la ville.
— Récupérer la somme des prix par seconde.

Les fonctions de récupération retournent un valeur stocké mais la valeur stock est recalculé si le besoin l'est, par exemple le bonheur moyen ne varie pas selon le temps mais si un nouveau bâtiment est placé on doit donc recalculer le bonheur moyen.

4.4 Gestion du son

Pour la gestion du son nous avons décider de pouvoir traiter facilement les musiques et les bruitages du jeu avec un mixeur. Nous avons donc crée une classe. L'HUD supérieur permet de la piloté à l'aide de trois sliders : Un pour le contrôle du volume de la musique, un pour celui des bruitages et un troisième englobant les deux, appelé master.

4.5 Launcher et gestion des sauvegardes

Chapitre 5

Tests

Chapitre 6

Bilan

Chapitre 7

Conclusion

Chapitre 8

Annexes