**Carnet de Bord**

**Jour 1 :**

La première étape de la journée a été consacrée à la recherche d’un projet utilisant la génération procédurale. Plusieurs vidéos abordant le sujet de la génération procédurale de terrain entre autres ont été regardées.

Après être tombé sur une vidéo abordant le principe des algorithmes de « Wave Function Collapse » j’ai décidé d’utiliser cette méthode de génération procédurale dans mon projet. J’ai donc commencé à chercher d’autre référence sur ce sujet et à imaginer les possibilités que m’offrait cette méthode.

Après mes recherches j’ai décidé de partir sur un programme de génération procédural de terrain en deux dimensions en utilisant un système de Tile map. J’ai alors commencé à chercher des assets de terrain en pixel art pouvant être utilisé dans une Tile map.

Après avoir trouvé et importer les assets dans Unity, j’ai commencé à créer le Scriptable Object TileSO contenant une tile qui constitue un visuel, et quatre liste de TileSO représentant les tiles qui pouvait être placées au Nord, au Sud à l’Est et à l’Ouest de cette tile.

A screenshot of a computer

Description automatically generated

**Jour 2**

Le deuxième jour a été consacré à la réalisation du Script GameManager qui contient l’ensemble de la logique de génération du terrain. Plusieurs fonctions comme GenerateMap, GenerateCell et GetPossibleTile ont été réalisé le deuxième jour. Après quelques itérations sur le programme et visionnage de vidéos portant sur le Wave Function Collapse, je me suis rendu compte que j’était parti dans la mauvaise direction. Suite à cette réalisation, j’ai créé le Script SuperpositionCell qui contient 5 listes de TileSO représentant les possibilité d’un cellule, et les possibilités des cellules se situant au Nord, au Sud, à l’Est et à l’Ouest de cette cellules, ainsi que des méthodes de permettant d’actualiser ces listes. La méthode UpdateAllNeighbors a également été créer. Grace à toutes ces fonctions implémentées, il était désormais possible de générer un map à l’aide de tiles créer précédemment montées ci-dessousA black background with white text

Description automatically generatedA black background with white text

Description automatically generated

. Un premier problème est apparu : La génération de map causait une Stack overflow. Pour résoudre se problème, la logique récursive a dû être repenser. Avant modification, le programme actualiser une cellule en récupérant les tile qui n’entrait en contradiction avec aucun des voisins dans une liste, et si cette liste correspondant aux possibilités qu’avait déjà la cellule avant modification, la cellule été alors catégoriser comme ayant déjà été actualiser et serait ignorer pour les itérations suivantes. Désormais dès lors qu’une cellule a été actualisée une fois elle ne sera plus actualisée par ses voisins jusqu’à la prochaine boucle de génération. Suite à cette modification, la map pouvait se générer et est alors apparu un deuxième problème, la map présentait des problèmes de génération comme montrer sur l’image ci-dessous :

A screenshot of a video game

Description automatically generated

Le problème venait du fait que lorsque tous les voisins avaient été actualiser une fois et que la map n’était pas définie, une cellule au hasard était choisie pour être définie et continuer la génération de la map. Le problème à était résolu en intégrer la prise en compte de l’entropie (le nombre de possibilité que peut prendre une cellue). Désormais, la cellule avec le moins de possibilité restante, ou une cellule au hasard parmi les cellules ayant le moins de possibilité restante si plusieurs cellules ont la même entropie, est choisi pour continuer la boucle.

**Jour 3**

Le début du jour 3 a été consacrer à l’ajour de nouvelles cellules afin de diversifier la map, notamment les zone d’eau qui était trop linéaire et ne former que des zones rectangulaires. Des coins de terre ont donc était rajouté pour permettre aux zones d’eau d’être moins linéaires.

**A black background with white text

Description automatically generated**

La suite de la journée a été dédié à l’implémentation d’une fonctionnalité, celle de la modification d’un terrain déjà existant. Pour ce faire, j’ai implémenté la fonction SetCell qui permet d’assigner à une cellule une tile particulière et d’actualiser la map en fonction de la cellule qui a été ajoutée. Dans l’Update, lorsqu’un clic de la souris était détecté, la position de la souris était récupérée puis converti en une position sur la tile, et si cette position correspondait à une cellule existante, la cellule était alors modifiée pour être soit ne case d’herbe, ou une case d’eau dépendant de si le clic était un clic droit ou un clic gauche. Suite à cette implémentation un problème est apparu, la carte ne se regénérer pas. Le problème venait de l’algorithme actualisant les possibilités d’une cellule. A ça création, une cellule peut être n'importe quelle tile existante, et pour actualiser les possibilités, on passe une liste de cellules qui respectent les contraintes de tous les voisins puis on vérifie pour toute les tiles du jeu si la liste de cellule respectant les contraintes contient cette tiles. Si ce n’est pas le cas, on enlève cette tile des possibilités. Cette méthode par soustraction à l’inverse d’une méthode par ajout à l’avantage d’empêcher d’ajouter des possibilités qui été déjà été écarté précédemment. L’inconvénient est que lors de la modification du map déjà existante, le liste de tile possible peut ne contenir aucun tile en commun avec la liste des possibilités. Dans ce cas il faut réinitialiser les possibilités de la cellule pour pouvoir lui donner de nouvelle possibilité. Une fonction ResetPossibilities à donc été implémenter dans la class SuperpositionCell. Un autre problème s’est alors posé. Après la modification d’une cellule, il est possible qu’un ou plusieurs de ces voisins se retrouve sans tile possible car ces voisins sont incompatibles. Pour régler ce problème, il faut alors réinitialiser les voisins de cette cellule qui n’ont pas encore été modifé, afin que cette cellule puisse se concentrer sur les possibilités induites par les cellules ayant été modifié.

**Jour 4**

Suite au nombreuses modifications apportées la veille, il était désormais possible de modifier une map, un problème se posait cependant, lorsque l’on placé une nouvelle cellule sur la map, celle-ci ce regénérait en intégralité, ce qui n’était pas le résultat désiré puisque les case qui restaient valide dans la nouvelle configuration ne devaient pas être modifié. La quatrième journée a donc été consacrée à tenter de corriger ce bug. Un problème a été identifié dans la fonction UpdateAllNeighbors. Le problème étant qu’après avoir vérifié la compatibilité de voisins et réinitialiser les possibilités si nécessaire, on actualisait les possibilités d’une cellule sans vérifier si celle-ci était toujours valide dans le contexte actuel. Cependant la résolution de ce bug a amené un autre problème, le rendu présenatit parfois des rendu impossible similaire au problème survenu au jour 2. Diverses tentatives ont été effectué pour régler ce problème comme introduire une fonction de vérification pour s’assurer qu’aucun case n’avait de tile qui ne respectait pas les contraintes de ses voisins, et réinitialiser les problématique dans le cas contraire, mais c’est tentatives ont aménes d’autres problème comme des StackOverflow. Face à la complexité de résolution du problème et le temps qui commencer à manquer, il a été décidé d’abandonner les modifications et de revenir à la dernière version sable du programme dans laquelle la map se regénérait entièrement. Une nouvelle tile de haute herbe a été par la suite implémenté pour ajouter de la diversité.