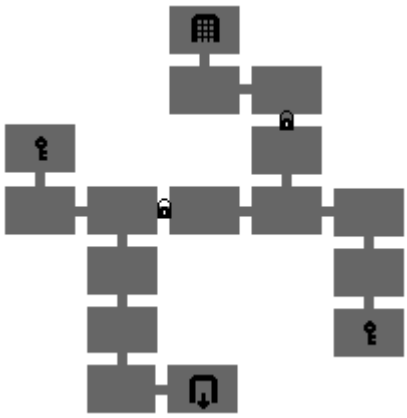
GENERATION PROCEDURALE DE DONJONS

I - Objectifs :

Ce TP aura les objectifs suivants :

- Code : Générer un donjon type Zelda de manière procédurale.
- Game-design : Définir les règles et le contenu qui permettront de rendre l'expérience procédurale maîtrisée. Réaliser le design des salles du donjon.



II – Règles de génération :

Le niveau généré devra répondre au moins aux règles imposées suivantes :

- Le donjon est découpé en salles rectangulaires de taille fixe reliées par des portes ou des passages.
- Le donjon commence et finit par des salles prédéfinies reliées par un chemin principal de X salles de long.
- Le chemin principal est barré par au moins un obstacle placé entre deux salles (i.e. porte cadenassée)
- Pour chaque obstacle, un chemin secondaire partant du chemin principal donnera la récompense permettant de continuer le chemin principal (i.e. clé)
- Chaque salle possède des caractéristiques qui doivent être utilisées pour rythmer l'expérience. Par exemple : Salle de début, salle de fin, positions des portes, difficulté, type de challenge, type de récompense, sens unique, pattern, ...
- Une salle cachée est ajoutée au donjon à un endroit permettant au joueur de déduire sa position. La règle doit être définie par les game-designers.
- Au moins une règle supplémentaire de génération doit être ajoutée par les game-designers.

A faire :

- GD : Réfléchir aux caractéristiques de salles qui permettront de rythmer la progression du joueur et à leurs utilisations dans l'algorithme de génération.
- GD : Réfléchir à la règle de placement de la salle secrète.
- GD : Réfléchir à une règle supplémentaire venant compléter les règles précédentes.

III - Mise en place du projet :

Pour réaliser ce TP, vous disposez d'une base de ressources :

- Créer un nouveau projet 2D sous Unity 2020.3 ou une version supérieure,
- Importer le package "TP_Procedural_Package.unitypackage" dans votre projet,
- Ouvrir la scène SampleScene.

A faire :

- GD / Prog : Se familiariser avec les outils mis à disposition : tilemap, prefabs ("player", "enemy", ...), scripts de base.
- GD / Prog : Mettre en place une méthode de travail collaboratif et les outils associés.

IV - Génération d'un chemin simple :

Le développement de ce projet sera réalisé en "bottom-up", en commençant par les fonctionnalités les plus primitives pour finir par les plus complexes.

La première étape est de définir un algorithme permettant de générer un graphe de chemin simple, sans embranchements, partant d'une salle de début pour arriver à une salle de fin.

Dans cette étape, le graphe du chemin est représenté par un arbre de données simples ! Définir les classes Node et Connection. Le graphe sera alors une succession d'instances de Node et de Connection.
Aucun GamObject ni component ne doit être instancié à ce stade ! La conversion du graphe en niveau jouable composé de GameObjects se fera à l'étape suivante.
L'algorithme doit être réutilisable ! La méthode servira à générer les chemins secondaires.

A faire :

- GD / Prog : Définir les règles relatives à la génération d'un graphe représentant un chemin simple.
- Prog : Implémenter une méthode de génération de graphe correspondant aux règles définies ci-dessus. L'algorithme a le droit à l'échec et sera relancé plusieurs fois jusqu'à obtention d'une solution, avec toutefois un nombre d'essai limité.

V - Génération d'un donjon simple :

Cette seconde étape vise à convertir le graphe de chemin généré ci-dessus en donjon.

A faire :

- GD : Réaliser le design de plusieurs prefabs de salles simples.
- Prog : Implémenter une méthode permettant d'instancier le donjon correspondant au graph en utilisant les prefabs fournis par les game-designers.

Au cours de cette étape, ranger tous les prefabs de salles dans une base de données (List) et filtrer cette base de données pour instancier une salle correspondant à la configuration de chaque nœud.

VI – Définition des règles de génération procédurale :

Maintenant que la génération d'un donjon simple est validée, vous pouvez définir l'algorithme relatif à la construction du donjon final.

On utilisera la même fonction de génération de graphe de chemin pour générer le chemin principal et les chemins secondaires (refactoriser le code si nécessaire).

A faire :

- GD / Prog : Définir les règles relatives à la génération d'un graphe représentant un donjon complet.
- Prog : Développer étape par étape les fonctionnalités définies précédemment.

VII – Design et features :

Pour ce TP, c'est aux game-designers d'implémenter l'ensemble du LD ainsi que les mécaniques « simples » du donjon, programmation comprise si possible. Soyez créatifs tout en restant dans la simplicité.

Autant que possible, les mécaniques de gameplay ajoutées doivent consolider l'expérience procédurale du jeu.

Dans cet exercice, il est très important d'optimiser la création de contenu. Pensez à un moyen d'adapter vos salles au maximum de situations possibles.

A faire :

- GD / Prog : Penser à un moyen d'optimiser la création de contenu.
- GD : Réaliser le design des salles et de leur contenu.
- « Au moins » 15 configurations de salles différentes doivent être produites.
- « Au moins » 3 tuiles de tilemap et/ou ennemis supplémentaires doivent être ajoutés.