THE BINDING OF DOOM

GENERATION PROCEDURALE DE NIVEAU DE DOOM

FEATURES

- Utilisation de prefabs pour toutes les salles et portes
- Utilisation de Json pour la configuration des salles
- Algorithme de Prim pour la génération des nœuds du labyrinthe
- Algorithme de Dijkstra pour obtenir la distance de chaque salle par rapport au Start
- Lois normales pour tout ce qui se rapporte aux Things (nombres, montres, bonus, armes, difficultés, …)
- Placement de clefs imbriquées

INSTALLATION

- Copier le dossier GenerativeDoom dans votre projet Doom Builder à l'emplacement : "doombuilder/Sources/Plugins/"
- Copier le dossier Data dans votre dossier de Build
- Pour permettre l'instanciation de prefab par le code :

Rendre public la function **InsertPrefabStream** dans CopyPasteManager et la variable **General.Map.CopyPaste**

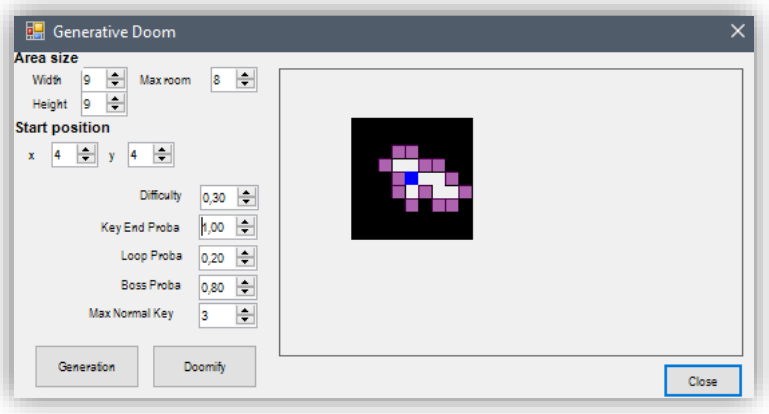
- Pour permettre le reset de la map dans le code :

Rendre public la fonction **General.Map.Map.Dispose()** dans MapSet et le **constructeur** de MapSet

- Pour la désérialisation du Json, j'utilise la librairie Newtonsoft.Json, il est nécessaire de rajouter Newtonsoft.Json.dll dans le dossier du Build

UTILISATION

TYPE DE MAP : DOOM 2



- X et Y en -1 équivaut à un random
- Max Room en -1 génère toute la taille du labyrinthe
- Difficulty (0 à 1) : influence le choix et nombre de monstres « difficiles »
- Key End Proba (0 à 1) : permet de choisir de verrouiller la salle de fin avant le placement aléatoire d'autres clefs
- Loop Proba (0 à 1) : probabilité que l'algorithme choisisse de créer une boucle (entre les salles) à chaque fois qu'il le peut
- Boss Proba (0 à 1) : probabilité de placer une salle de boss à la fin du niveau
- Max Normal Key : Nombre max de clef à placer aléatoirement sur les portes (indépendant du Key End Proba, mais cette dernière compte pour une si elle est placée)

Le bouton Generation génère le labyrinthe dans une prévisualisation (à droite), puis le bouton Doomify lance le choix et l'instanciation des prefabs et des Things.

N'oubliez pas d'appuyer sur entrée à la fin de la génération pour valider le dernier prefab.

Attention : chaque ancre personnalisée sur un prefab doit avoir son opposée (possibilité de se coller) dans au moins un autre prefab de chaque type de salle. En effet, l'algorithme génère de gauche à droite et de haut en bas, sans revenir en arrière si aucune salle compatible n'est trouvée pour le prochain index.

AMELIORATIONS

- Rajouter une préférence pour la position des Things (par exemple, ENUM : ALL, WEAPON, AMMO), et ainsi plus maitriser le placement des objets dans les salles
- Système de probabilité sur la sélection des salles, pour favoriser les moins choisies
- La version Doom 2 des maps ne permet pas d'Action sur les monstres et donc de provoquer la fin par la mort du boss, le type de map Doom in Hexen Format permet de cela, mais l'index des actions change et donc il serait nécessaire de mettre à jour les prefabs actuels (portes notamment)

EXEMPLE DE CONFIGURATION DE SALLE EN JSON

```
{
    "file": "normal/tetrisRight",
    "type": "NORMAL",
    "width": 608,
    "height": 512,
    "origin": { "x": 96, "y": 0 },
    "things": [{ "x": 48, "y": 48 },
        { "x": 48, "y": 240 },
        { "x": 176, "y": 256 },
        { "x": 464, "y": 464 },
        { "x": 464, "y": 48 },
        { "x": 64, "y": 144 },
        { "x": 192, "y": 144 },
        { "x": 320, "y": 160 },
        { "x": 320, "y": 256 },
        { "x": 320, "y": 368 }
    ],
    "doors": [{
        "side": 0,//2 up, 3 left, 0 down, 1 right
        "anchor": { "x": 192, "y": 512 }
    },
    {
        "side": 3,
        "anchor": { "x": 128, "y": 352 }
    },
    {
        "side": 2,
        "anchor": { "x": 192, "y": 0 }
    },
    {
        "side": 1,
        "anchor": { "x": 512, "y": 192 }
    }
    ]
},
```

WEBOGRAPHIE

https://randcode.wordpress.com/2012/10/07/dijkstras-algorithm/

https://github.com/garyng/Maze