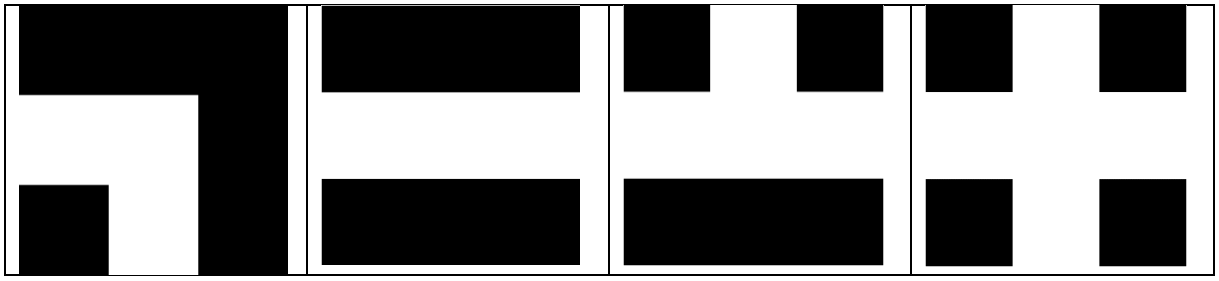
BARSONI Dorian

Projet Info4A : Partie 2

Dans la partie 2 du projet il était demandé de réaliser un labyrinthe dit « intéressant ». Je vous détaille dans ce compte rendu la manière dont je m'y suis pris.

L'idée qui m'est venue était de créer plusieurs paternes récurrent que l'on peut retrouver dans un labyrinthe dit « intéressant ». Les paternes utilisés sont les suivants (les carrés noirs sont des murs, les blancs sont du vide) :



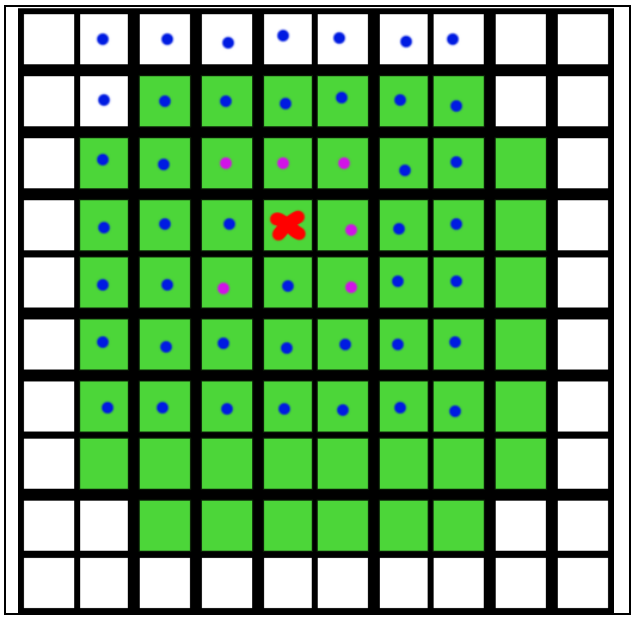
Une fois ces paternes créés sous forme de tableaux de taille 9 je les ai tous regroupés dans un tableau de pointeur. J'ai également créé une fonction rotate() permettant d'effectuer une rotation de 90° dans le sens horaire des paternes. La fonction selec_pattern() permet, lorsqu'on lui donne un tableau de taille 9, de sélectionner un de ses 4 paternes de manière aléatoire, puis de lui effectuer un nombre aléatoire de rotation (compris entre 0 et 3).

Quand bien même la rotation n'a pas d'effet sur le paterne 4, et 2 rotations correspondent à aucune rotation pour le paterne 2, je n'ai pas fait de cas particulier pour simplifier le code.

Maintenant qu'on a ces paternes on souhaite les appliquer sur la grille mais pas n'importe comment. Il ne faut pas que les paternes empiètent les uns sur les autres (c'est-à-dire qu'en positionner un en affecte un autre, ou encore que cela rende le labyrinthe non connexe). Je prends l'exemple d'un labyrinthe 10x10 pour expliquer le raisonnement :

- En vert nous avons les cases du labyrinthe ou, initialement, les paternes peuvent êtres positionnés
- La croix rouge représente le point d'application du paterne
- Les points magentas représentent les murs
- Les points bleus représentent les cases interdites pour que 2 paternes n'empiètent pas l'un sur l'autre

Projet Info4A : Partie 2



Les cases vertes sont ainsi sélectionnées pour éviter qu'un paterne soit coupé et ne sorte du tableau. De plus les coins du carré vert sont exclus car dans certain cas la fonction selec_patern() peut retourner le paterne 1 avec une rotation qui rendrait le labyrinthe non connexe.

On parcourt donc les cases vertes une à une en ayant à chaque fois une chance sur 2 d'y placer un paterne. Une fois qu'un paterne doit être placé on le copie sur la grille. Le point sélectionné représente le centre du paterne. Puis on réalise un carré 7x7 autour du paterne dans lequel on remplace tous les 0 par des 3 (également sur la croix rouge). Cela permet d'espacer les paternes de sorte qu'il y ait toujours une ligne de vide entre eux.

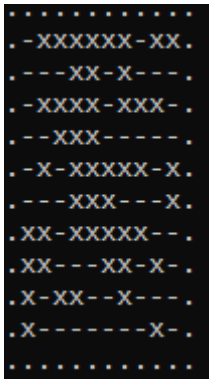
À chaque fois qu'on place un paterne on compte le nombre de mûrs présents sur la Grille. Si jamais le nombre de cases vides est inférieure ou égale à k (nombre de cases vides souhaité dans le labyrinthe) on arrête la construction du labyrinthe.

Enfin, une fois que les paternes sont placés, pour terminer la construction du labyrinthe on utilise une boucle while. Tant que le nombre de cases vides est différents de k et tant que le critère de sécurité est différent de 5 (choisit arbitrairement), on parcourt la Grille de la seconde case à l'avant dernière. Si la valeur de la case vaut 0 on y place un mûr et on vérifie si le labyrinthe est connexe. S'il ne l'est pas on annule la modification et on passe à la case suivante. S'il l'est alors on a une chance sur 2 d'effectivement remettre la case à 0. Ensuite on décompte le nombre de mûrs pour en déduire le nombre de cases vides. S'il est inférieur ou égale à k alors on sort de la boucle. On termine le while en incrémentant le critère de sécurité de 1.

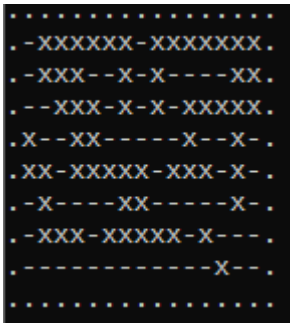
Toutes ces étapes nous donnent donc les résultats suivants (la valeur de k vaut, pour chaque labyrinthe, la moitié des cases que présente la grille) :

BARSONI Dorian

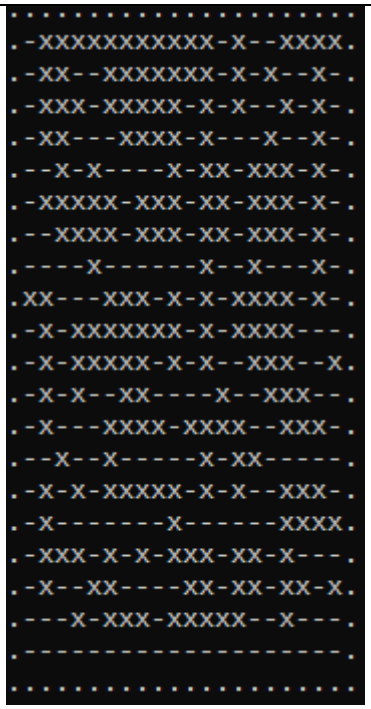
Projet Info4A : Partie 2



Labyrinthe 10x10



Labyrinthe 15x8



Labyrinthe 20x20

On obtient donc un labyrinthe connexe. Cependant je remarque que ma méthode laisse apparaitre parfois de grandes lignes droites ainsi que de grandes zones remplis de mûrs, rendant la résolution du labyrinthe plutôt facile. Je n'ai pas trouvé d'autres méthodes pour rendre mes labyrinthes plus « intéressant ».

(Je tiens à préciser ici que dans la zones des tests automatiques on peut lire « | !| Le labyrinthe ne contient aucune case blanche | !| ». Ceci est dû au fait que lorsque l'on test la fonction connexe on vérifie également le cas où le labyrinthe est uniquement composé de mûrs. Cela renvoie, comme demandé, un message d'erreur)