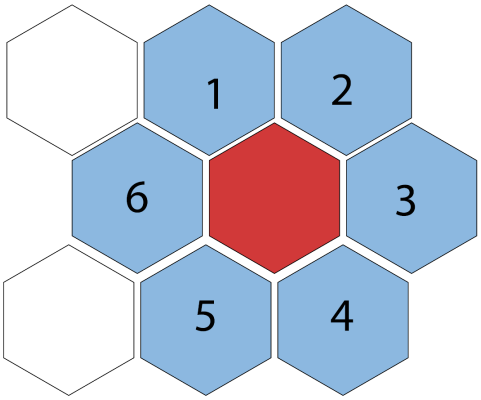
Rapport technique sur les stratégies et autres

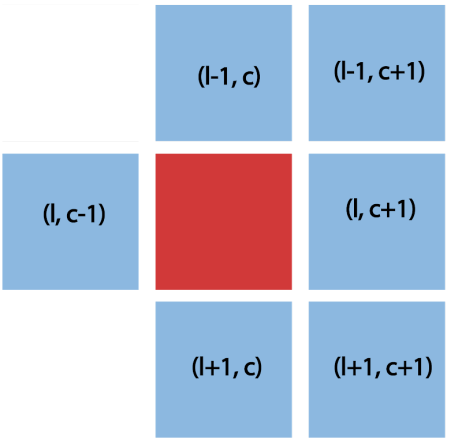
Création de la map

Génération

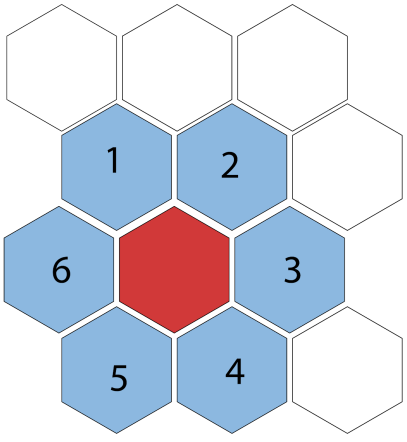
Pour implémenter une map, il faut interpréter des positions d'une grille de cellules hexagonales en matrice. Cette interprétation va différer selon un décalage d'une ligne de cellules paire ou impaire.

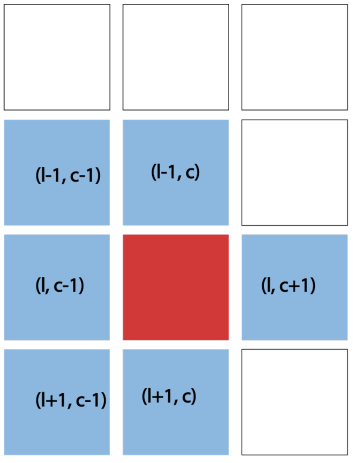
Pattern de voisinage - ligne impaire





Pattern de voisinage - ligne paire





Pour répondre aux spécifications du jeu et de l'interface graphique, on implémente un type *Coordinates* qui est une paire d'entier (*pair<int,int>*). Le premier argument correspond à l'axe y (ligne), le second à l'axe x (colonne). Par ailleurs, nous créons un type *Matrix* (une matrice d'entier), qui permet la simulation de grille de cellules hexagonales et d'empêcher les confusions textuelles avec la Map.

L'initialisation de la map commence par le placement aléatoire d'un nombre de cellules égal au nombre de territoires souhaité. A ce stade-ci, chaque cellule représentera chaque futur territoire. On expand chaque territoire cellule par cellule, puis territoire par territoire.

La matrice résultante doit être traduite sous forme de structures utiles pour le déroulement du jeu (tels que *SMap*, *SCell*, …) et permettant une génération compatible d'une interface graphique (*SRegions*, *SRegion*, *SRegionCell*). Afin d'obtenir une map parsemée et contenant une seule composante connexe, la condition d'arrêt est donc la connexité et le remplissage total des cellules à 75%.

La connexité est évaluée par la fonction *getMaxConnexité(int,SMap)*, développée à l'origine pour une stratégie. Le premier paramètre est un *IdPlayer*, nous nous appuyons sur le principe de coloration :

1. Création d'un vecteur du nombre total de cellule, la valeur associée à chaque id représente une même composante. On raffine en accédant aux voisins de chaque cellule.
2. Déclaration d'une map qui va permettre de compter le nombre de SCell associé à chaque composante. La fonction renvoie ensuite la valeur de la plus grande.
3. Pour finir, si la valeur retournée est égale au nombre de SCell, on a bien une même composante connexe.

La difficulté intervient quand il faut créer le modèle (SMap et SCell, à la volée), tout en gérant l'expansion, notamment l'ajout des voisins au modèle de données, via la fonction *addNewNeighborsSCell* (qui l'effectue pour une SCell) :

1. Récupération d'une liste de coordonnées étrangères non vides autour d'une SCell.
2. Le principe de cette fonction consiste en l'instanciation des voisins d'une SCell en ayant pour initialisation une liste de coordonnées.
3. Si une des cellules créée n'a pas de liste de voisins instanciée, la fonction le fait. Sinon elle effectue une "réallocation mémoire" pour pouvoir étendre les capacités des tableaux de SCell voisines.

Cette instanciation est réciproque car la relation "être voisin" l'est aussi, il fallait donc penser à instancier d'une part la SCell étudiée à l'origine mais aussi réitérer l'opération pour chacune de ses SCell voisines en ajoutant la SCell étudiée dans leur liste de voisins.

L'utilisateur détermine la forme de sa map et le nombre de territoire qu'il souhaite poser grâce aux paramètres *nbLignes*, *nbColonnes* et *nbTerritoires*.

Attributions

Pour l'attribution des territoires (SCell) d'une map, on opte pour une distribution dans l'ordre d'apparition des territoires. Ainsi, pour 2 joueurs, territoire 0 ira au joueur 0, territoire 1 ira au joueur 1, territoire 2 ira au joueur 0.

Pour l'attribution des dés, étant donné le manque de règles précise sur l'attribution de dés à l'initialisation de la partie, nous avons décidé d'attribuer un même nombre de dés (4) par territoire.

Stratégies

Stratégie de Dijkstra

La Stratégie Dijkstra se base sur l'une des règles du jeu. En effet, la redistribution des dés s'effectue, pour le joueur qui vient de jouer, à la fin de son tour. Le nombre de dés redistribués est égal au nombre de territoires appartenant à la plus grande composante connexe possédée. Donc son but principal est de rejoindre la plus grande composante connexe aux autres plus petites. Pour cela, un calcul de la distance du plus court chemin (le poids étant représenté par les dés à combattre) est réalisé. On essaie donc de rejoindre coup par coup, un territoire appartenant à une autre composante connexe. Une fois le tour terminé, si la deuxième composante connexe a été rejointe, alors cela garanti un nombre de dés gagnés plus important. C'est ce qui fait la force de cette stratégie. Le principal problème, vient quand il ne nous reste qu'une seule composante connexe sur le terrain. En effet, on ne possède donc aucun autre territoire à rejoindre et donc le calcul du plus court chemin pour rejoindre une autre composante connexe devient inutile. Pour pallier ce problème, une attaque reprise de la stratégie Bourgeoise (ci-dessous), qui attaque en fonction de la différence de dés de la cellule attaquante et de la cellule attaquée est réalisée. L'attaque est effectuée entre les deux cellules qui ont la plus grande différence.

Stratégie Bourgeoise

Cette stratégie repose sur le calcul de différences de dés entre la cellule attaquante et la cellule attaquée. Pour chaque territoire appartenant au joueur, un calcul de différence de dés est calculé. La différence la plus grande est priorisée. Si deux différences sont égales c'est la cellule attaquante qui comporte le moins de dés qui a la priorité car selon des probabilités, une attaque à 2 dés contre 1 a plus de chance d'aboutir qu'une attaque à 3 dés contre 2. Si la différence de dés est égale à 0, un calcul, afin de savoir les ressources nécessaires pour obtenir 8 dés sur chaque territoire, est fait. Si cette ressource nécessaire est compensée par les dés en stock et le nombre de dés redistribués au tour suivant alors la stratégie tente l'attaque sans risque de se retrouver à 1 seul dé sur la cellule.

Exemple :

Dans le cas où le joueur 0 (moi) a 6 territoires avec 8 dés avec un stock de 7 dés. Si une des cellules souhaite attaquer une autre cellule à 8 dés alors la différence est de zéro mais si notre attaque échoue notre stock de dés sera capable de compenser la perte des 7 dés perdus lors de l'attaque.

Stratégie du chaos

Cette stratégie est une stratégie aléatoire. Un premier tirage aléatoire est effectué avec le choix des cellules attaquantes et attaquées. Le joueur a une chance sur deux d'attaquer. Si le joueur décide d'attaquer alors toutes ses cellules sont répertoriées dans un vecteur, l'indice du vecteur est tiré au sort. Puis parmi sa liste des voisins, un autre est également tiré au sort. La cellule attaquante est forcément possédée par le joueur mais la cellule attaquée n'est pas soumise à une condition particulière. Cette stratégie peut donc proposer des coups invalides.
Cette stratégie n'est pas destinée à être performante mais nous as permis de construire notre boucle de jeu, notre distribution de dés et notre validation d'attaque.