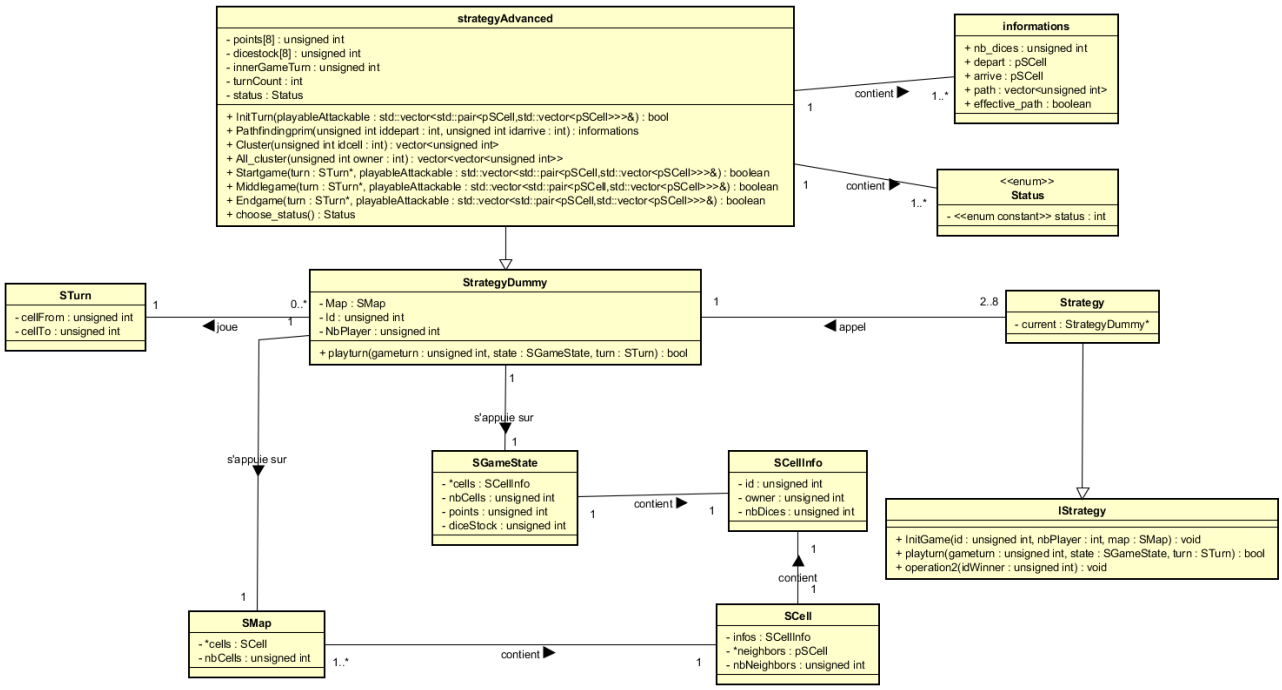
Rapport Mini-projet C++

Notre stratégie

Modèle objet



Description

Notre stratégie contient 3 états différents.
D'abord, nous réunissons toutes nos régions en un seul groupe pour gagner un maximum de dés à chaque tour.
Ensuite, nous ne jouons que les coups qui ont de grandes chances de remporter la région de l'adversaire.
Enfin, lorsque nous possédons un avantage de territoire sur nos opposants, nous attendons pour stocker des dés. Une fois que nous en avons assez, nous les attaquons de nouveau.
Le but de cette manipulation est de garder toutes nos régions imprenables. En effet, ces dernières possèdent 8 dés chacunes en fin de tour.

1. **StartGame**

Cette stratégie se déclenche tant que toutes nos régions ne sont pas réunies.
Son but est de trouver le plus gros groupe de régions et ensuite de relier les autres groupes à celui-ci. Pour cela, cette stratégie calcule tous les plus courts chemins entre une région déterminée et toutes les autres régions atteignables.
Pour cette même région, elle choisit parmi tous les chemins, celui qui est le moins

coûteux en terme de dés et elle répète l'opération pour chaque région qui n'appartient pas au plus gros groupe.
Enfin, elle choisit le chemin optimal, c'est-à-dire le chemin pour lequel le premier coup est le plus simple à remporter. Ce premier coup peut provenir de la région isolée ou du groupe le plus important.
Cette stratégie répète l'opération à chaque tour qui lui est donnée. À la fin du tour, si cela est possible, nos régions forment une composante connexe.
Il faut souligner que si la stratégie ne peut pas rejoindre toutes les régions dans le même tour à cause d'une région trop éloignée ou d'un nombre de dés trop important sur le chemin, elle passera la main à la stratégie la mieux adaptée pour la situation : la stratégie Middlegame ou Endgame. Lorsqu'elle passe la main, la stratégie restreint les déplacements possibles à ceux du plus gros groupe.

2. **MiddleGame**

Cette stratégie se déclenche lorsque les autres stratégies ne sont pas adaptées à la situation. En terme de conditions cela se traduit par : toutes les régions sont regroupées et le stock de dés est inférieur à 8 dés.
Cette partie de la stratégie globale est plutôt simple. Nous jouons ici à chaque coup l'action qui a le plus de chance de réussir. Tant que nous possédons un avantage d'au moins 1 dé sur la région d'un opposant, nous attaquerons. Il faut noter que durant notre tour de jeu nous ne pouvons pas quitter la stratégie Middlegame.

3. **Endgame**

Cette stratégie se déclenche lorsque nous sommes le joueur qui engrange le plus de dés à la fin du tour ou lorsque nous possédons un stock d'au moins 8 dés.
La stratégie ne joue pas tant que toutes nos régions ne possèdent pas 8 dés et que nous ne possédons pas 8 dés en stock. Elle joue seulement lorsque au prochain tour, son stock de dés et son gain de dés lui assurent de remplir toutes ses régions jusqu'à 8 dés.

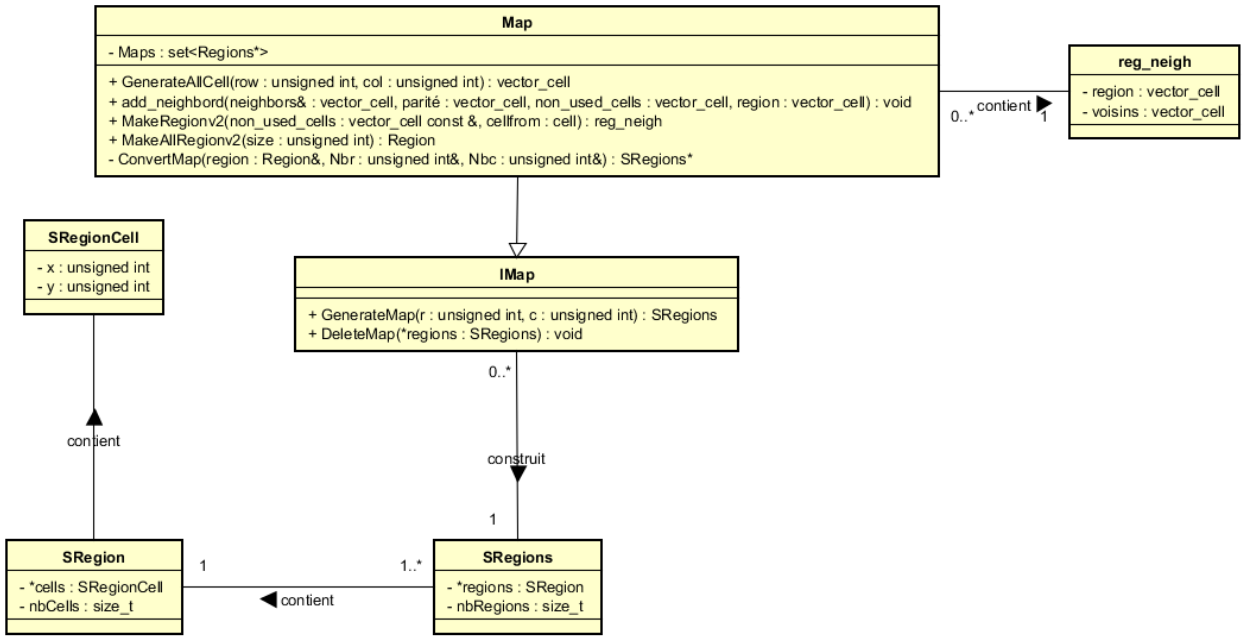
Pathfinding

Le pathfinding est un élément crucial de la stratégie de début de partie. Il est basé sur l'algorithme de dikjstra. Cet algorithme a pour but de trouver le chemin le moins coûteux entre deux sommets d'un graphe.
Ici, les sommets du graphe sont les régions. Les arêtes du graphe sont représentées par la relation de voisinage entre les régions. Deux régions voisines forment deux arêtes orientées. Le poids de l'arête est le nombre de dés présents sur la région vers laquelle on se dirige. Par exemple, si les nombres de dés des régions 4 et 5 sont différents alors le poids des deux arêtes orientées sera différent. La première prendra la valeur du nombre de dés de la région 5 (on se dirige de la région 4 vers la région 5). La deuxième prendra la valeur du nombre de dés de la région 4 (on se dirige de la région 5 vers la région 4).

L'algorithme se base sur deux éléments principaux pour trouver le plus court chemin. La distance parcourue depuis la région de départ pour arriver jusqu'à la région que l'on veut annexer et la région par laquelle il faut passer pour ajouter une nouvelle région. Nous appellerons ses éléments "distance" et "prédécesseur". Par exemple, si la région 3 est joignable depuis la région 4 et la région 5, le prédécesseur de 3 sera la région avec la plus petite distance.
L'algorithme cherche à aller d'une région à une autre. Lors de l'initialisation, il fixe les distances pour rejoindre chaque région à l'infini (représenté par 1000 dans le code). Lors de la première étape, l'algorithme ajoute la région de départ. Il actualise la distance des régions voisines avec leurs dés. Le prédécesseur de ces régions sera la région de départ. Lors des prochaines étapes, l'algorithme ajoute aux régions visitées la région avec la distance la plus faible. Il parcourt alors les voisins de cette région. Pour chaque voisin, il compare la distance enregistrée et la nouvelle distance. La nouvelle distance est l'addition de la distance pour arriver à la région ajoutée et la distance pour rejoindre le voisin depuis la région ajoutée. Si cette nouvelle distance est inférieure à l'ancienne alors la distance est mise à jour et le prédécesseur remplacé par la région ajoutée.
L'algorithme s'arrête lorsqu'il a ajouté la région d'arrivée ou lorsqu'il n'a plus de région à visiter. Grâce à la mémoire des prédécesseurs, en partant de la région d'arrivée il peut remonter directement jusqu'à la région de départ et renvoyer ainsi le chemin le plus court reliant les deux régions.

Notre générateur de cartes

Modèle objet



La génération de la carte se compose de deux étapes principales. La première

consiste à sélectionner les cellules disponibles. La deuxième étape consiste à relier les cellules pour créer des régions.

Description

1. **Génération des cellules**

Tout d'abord, toutes les cellules de la carte sont générées.
Les dimensions de la carte sont fixes. Elle contient 900 cellules soit une carte de 30 de largeur et 30 de longueur. Ensuite, pour créer des disparités dans la carte nous supprimons certaines de ses cellules aléatoirement.
En effet, nous supprimons 15 groupements d'en moyenne 10 cellules (le nombre est défini de manière aléatoire). La première cellule de ce groupe est choisie aléatoirement parmi toutes les cellules restantes. Au cours de la suppression des cellules, l'algorithme utilisé garanti que toutes les cellules restantes sont connexes (l'algorithme empêche les bords de la carte de se rejoindre).

2. **Génération des régions**

La première cellule pour générer toutes les régions est choisie au centre de la carte parmi les cellules restantes. À partir de cette cellule est générée une région d'en moyenne 20 cellules (le nombre est défini de manière aléatoire). Les cellules voisines de cette région sont stockées.
La cellule de départ de la prochaine région est tirée au hasard parmi les voisins actuels. Lorsque la deuxième région est construite on ajoute les voisins de cette dernière aux voisins encore disponibles de la première région et ainsi de suite jusqu'à obtenir un nombre de région entre 25 et 30. Dans le cas où l'algorithme n'arriverait pas à générer une carte avec les cellules manquantes, il prend alors toutes les cellules disponibles (les 900 évoquées au départ).
Il faut noter que l'exemple donné ici est spécifique à une carte de 30×30 mais cette valeur peut être modifiée avec un attribut (size). La modification de cet attribut modifiera aussi d'autres valeurs comme le nombre de région ou le nombre de groupements de cellules retiré de la carte.