EISTI - École Internationale des Sciences du Traitement de l'Information

SOULAGE Tom VAUXEL Paul

Projet Semestre 2 : Qwirkle

Table des matières

1.1 Organisation du fichier

Le fichier source est composé de 7 fichiers :

- **QwirkleJeu.pas** : programme principal, exécutable.
- **UmanipListe.pas** : regroupe les fonctions utilisées pour manipuler des listes (supprimer, ajout en tête...).
- **Upion.pas** : fonctions/procédures liées aux pions (création, affichage...).
- **Umain.pas** : fonctions/procédures liées à la main (création, affichage, manipulation...).
- **Ugrille.pas** : fonctions/procédures liées à la grille (création, affichage, insertion de pion...).
- **UregleQwirkle.pas** : fonctions/procédures liées aux règles du jeu ainsi qu'au comptage des points.
- **UnitType.pas** : tous les types et constantes utilisés par le jeu.

1.2 Plateforme

Plateau du jeu

Le plateau du jeu est modélisé par une grille. Cette grille est un tableau fixe à deux dimensions, dont la taille est modifiable, en changeant la constante "M" du fichier.

Pion

Nos pions sont modélisés par 6 chaînes de caractères :

- -
- +
- =
- #
- ¤
- ~
- **o**

Les couleurs des pions du jeu sont les suivantes :

- Bleu
- Vert
- Cyan
- Rouge
- Magenta
- Marron

Chaque case "vide", de la main et de la grille, est représenté avec un pion de la forme "**.**" de couleur noir.

Main

La main des joueurs est un tableau fixe à une dimension, de 6 pions. Dès lors qu'un pion est ajouté dans la main, il s'agit du pion présent en début de pioche.

Pioche

La pioche du jeu est une liste contenant des pions. Cette pioche est mélangée dès le début et pendant la partie.

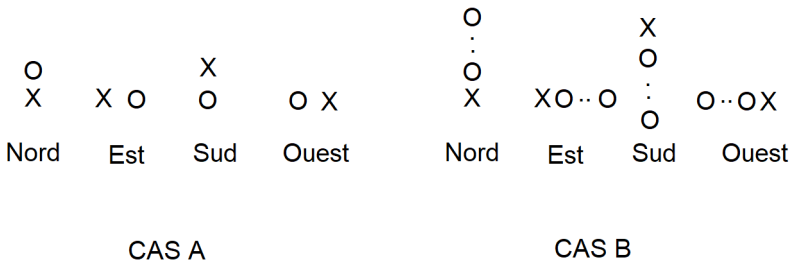
1.3 Règles du jeu et points

Concernant les règles du jeu, nous avons étudié tous les cas possibles du plus simple au plus complexe et du plus probable au plus rare.

Lorsque nous jouons au Qwirkle, nous pouvons placé des pions uniquement horizontalement et verticalement. Les diagonales sont donc interdites. De ce fait, nous avons conclu qu'il fallait étudier les 4 directions possibles autour d'un pion. Nous avons donc créer un type pour vérifier ces cas. Il se présente comme un tableau à une dimension composé de 4 booléens où le premier représente le nord du pion, le second l'ouest etc. On tourne dans le sens des aiguilles d'une montre.

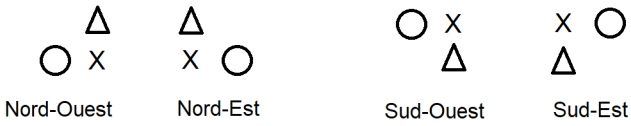
Premièrement, nous avons établi une fonction vérifiant que le pion que l'utilisateur souhaite placer est bien voisin d'un pion déjà présent sur la grille. C'est la fonction *verifposition*.

Ensuite, nous nous sommes attaqué au cas où le pion qu'on souhaite placer a un seul voisin. C'est le cas le plus courant dans le jeu. Cependant, nous nous sommes aperçu que deux autres cas apparaissaient. En effet, le cas où le pion est entouré d'un seul voisin (cas A) et le cas où le pion est entouré d'au moins deux pions sur la même ligne (cas B)



Pour le cas B, le nombre de pions déjà placé n'étant pas fixe, nous avons établi une fonction calculant le nombre de pions sur la ligne. C'est la fonction *comptepion*.
On établit également une fonction qui vérifie que le pion qu'on souhaite placer n'est pas déjà posé sur la grille. C'est la fonction *verifdoublon*.
On obtient ainsi la fonction *verifattribut1* qui vérifie, en fonction de l'orientation du pion voisin, si le pion qu'on souhaite placer est valide.

Ensuite nous arrivons au cas où le pion qu'on souhaite placer à deux voisins sur la grille. On distingue encore 2 cas : un cas simple et un cas compliqué.



Cas simple

Cas compliqué

Le cas compliqué est plus difficile puisqu'il fusionne deux lignes pour en former une seule. De ce fait, là où pour le cas facile, on doit simplement appeler deux fois la fonction *verifattribut1* (une pour chaque direction), nous devons pour le cas compliqué compter le nombre de pions dans les deux directions (N/S et O/E) pour ensuite faire une vérification sur une seule ligne.

Nous arrivons maintenant sur les cas quasi-inexistants. Premièrement, le cas à 3 voisins. Il n'est pas très complexe puisqu'il s'agit d'un mélange de la fonction *verifattribut2complexe* et de la fonction *verifattribut1*.
Le cas à 4 voisins est également facile puisque ce sont les 2 cas de la fonction *verifattribut2complexe* réunis.

Nous avons ensuite été confrontés au cas où le joueur souhaite jouer plusieurs pions dans le même tour. En effet, à partir du moment où le joueur souhaite rejouer dans le même tour, il est dans l'obligation de jouer sur la même ligne que son pion précèdent.
Pour vérifier ceci, nous avons eu l'idée d'enregistrer dans une variable toutes les positions possibles en fonction du premier pion posé par le joueur.
Nous avons ainsi créé un nouveau type qui se présente comme un tableau à deux dimensions. La première dimension prendra comme valeur uniquement 1 et 2 (qui correspondent respectivement à la position i et j). La seconde prendra comme valeur 1, 2, 3 ou 4 (car on aura au maximum quatre positions à enregistrer peu importe les cas).

Nous avons ensuite créé une variable globale du type en question pour enregistrer les positions. On utilise cette variable dans chaque cas. Grâce à ça, au moment où le premier pion est posé, nous obtenons les uniques positions possibles si le joueur veut jouer un autre coup dans le même tour.
Il ne restera ainsi qu'à comparer avec les valeurs que l'utilisateur insèrera.

Pour terminer, la question des points. Nous avons là aussi créé une variable globale qui compte le nombre de points pour chaque cas. Cela n'a pas été une grande difficulté puisque nous avions déjà le nombre de pions présents sur la ligne grâce à la fonction *comptepion*. Nous avons simplement dû ajouter une condition en cas de Qwirkle.

1.4 Déroulement du Jeu

Début

Les joueurs entrent leur nom et leur âge, on leur demande de rentrer le nombre le plus élevé de pions ayant un attribut commun.

Le 1er joueur choisit son premier pion à insérer dans la grille, celui-ci est placé automatiquement au milieu de la grille afin d'exploiter au mieux la taille de celle-ci, pour éviter que les pions soient, dès le début, placés en bordure de grille.

Tour

A chaque tour, à l'exception du premier, il est proposé aux joueurs de remplacer ses pions ou d'en insérer dans la grille.

Si dès le début de son tour, le joueur choisit de ne pas poser de pions sur la grille, alors il peut échanger ses pions. Afin, d'accélérer la partie, nous avons décidé de laisser la possibilité au joueur de changer plusieurs fois, au maximum 6 fois en accord avec les règles, le même pion. Nous demandons alors au joueur, combien il veut changer de pions, puis, en fonction de ce nombre (n), il peut échanger le(s) pion(s) (n) fois de son choix. Ensuite, il passe automatiquement son tour à l'autre joueur.

Sinon, le joueur insère dans la grille autant de pions qu'il souhaite, dans la mesure du possible (nombre de pions qu'il a dans la main), et des règles, qui lui autorise ou non de placer un pion dans la grille.

Fin

La partie se termine dès lors que la pioche est vide et qu'une main d'un des deux joueurs est vide. Pour cela, on initialise une variable qui compte le nombre de pions dans la pioche et qui diminue au fur et à mesure de la partie. On utilise également une fonction vérifiant si la main d'un des joueurs est vide.

1.5 Remarques

Le jeu n'est pas paramétrable pour le nombre de joueurs : il est conçu uniquement pour 2 joueurs. Il n'y a pas de mode "Vs Ordinateur".