**Jeu: Vitrail**

**CSI2772 – Concepts avancés en C++**

**Automne 2019**

**Ecole de génie électrique et de science informatique**

**Université d’Ottawa**

Professor : Robert Laganière

Mathieu Marc Bellefeuille 6501369

Matthew Tran 300028206

Date de remise: Vendredi 06 Decembre 2019

**Class Joueur**

Cette classe nous permet de représenter les joueurs lors de la partie. Elle est composée de quatre variables. Une variable de type String *nom* pour stocker le nom, une de type Integer point pour stocker les points, une int position pour représenter la position du vitrier, et une dernière variable de type Vitrail appelée \*vitrail qui est un pointeur vers le vitrail associé au joueur.

Nous retrouvons également le constructeur paramétrisée de cette classe qui prend en argument le nom du joueur ainsi que sont vitrail, puis initialise les valeurs de position et de points à 0.

Donc nous avons les méthodes suivantes qui permettent de modifier l’état du joueur:

|  |  |
| --- | --- |
| **Nom de la méthode** | **Description** |
| getPoint | Nous retourne le nombre de point du joueur. |
| calculatePoints (int unusuedVitres) | Prend en argument le nombre de vitre non utiliser lors de la partie pour calculer le nombre de point à ajouter ou soustraire au joueur |
| getPosition | Retourne la position du vitrier du joueur |
| GetVitrail | Retourne un pointeur vers le vitrail de notre joueur |
| changerPoints | Prend en argument le nombre de point a ajouter au joueur et modifie la variable associer |
| Operator - -(int) | Vas déplacer le vitrier de un vers la droite lors de l’appel (position diminue de 1) |
| Operator -= (const int num) | Prend en argument le nombre de positions à déplacer à droite, soustrait la position de ce nombre. Lance une exception out\_of\_range si c’est hors la limite |
| Operator ~ | Remet le vitrier a l’extrême gauche du vitrail |
| Operator<< | Permet d’afficher tout le vitrail du joueur ainsi que ses identifiant et point. |

**Class Lots**

Cette classe est composeé d’un vecteur de type char permettant de stocker les caractères de surplus, un tableau (array) *couleur* de type char composer initialisée avec les valeurs possible des couleur soit G, J, B, R, O, et un double pointeur de type char qui stock les lots de vitrail que les joueur peuvent accéder.

La classe est également composé d’un destructeur qui libère la mémoire de lotFenetre ainsi que un constructeur par défaut qui nous permet d’initialiser par défaut les valeur de notre lot. Le constructeur crée un tableau dynamique à 2 dimensions composés de sélections aléatoires de couleur pour former un grid de 5 colonnes x 4 rangées.

Donc nous avons les méthodes suivantes:

|  |  |
| --- | --- |
| **Nom des method** | **Description** |
| getLot (char couleur, int col) | Prend en argument une couleur type char et une colonne de type int pour retourner vrais si la couleur est dans la colonne du lot |
| isEmpty | Retourne vraie si le lot et le surplus est vide |
| lotColonneEmpty(int col) | Retourne vraie si la colonne dans le paramètre est vide |
| surplusEmpty() | Retourne vraie si seulement le surplus est vide. |
| lotEmpty() | Retourne vraie si seulement le lot est vide. |
| ramasseVitre (char couleur, int numeroLot) | Prend en argument une couleur et un numéro de colonne du lot pour retourner un vecteur de char contenant les couleurs compris dans cette colonne. Ça va remplacer les char dans la colonne du Lot avec des ‘.’ Pour indiquer qu’ils ont été ramassés. |
| Reset\_L | Recharge le lot avec un surplus vide et d’autres valeurs de couleurs aléatoires |
| Operator<< | Surcharge l’opérateur qui permet d’afficher le contenu du lot |

**Class Vitrail**

Cette classe est composé de quatre variables dont un tableau de type char composé des couleurs prédéfinis soit G, J, B, R, O, un double pointeur de type char fenetre nous permetant de stocker notre vitrail comme tableau dynamique 2D, et deux variables de type int colonnes et rows qui nous permet de stocker le nombre de colonnes et de rangées dans notre tableau (pour donner à l’utilisateur le choix d’utiliser les settings standard de 5x7 ou de définir ses propres colonnes et rangées).

Nous avons également un constructeur paramétrisée qui prend en argument le nombre de colonne et de rangées pour créer le vitrail. Les colonnes et les rangées sont initialisées a 7 et 5 respectivement avec l’option pour l’utilisateur de les modifier au début de la partie.

Donc nous avons les méthodes suivante:

|  |  |
| --- | --- |
| **Nom des method** | **Description** |
| construireVitrail (vector<char> vires, int col) | Prend en argument un vecteur de type char qui représentent les couleurs pris du lot sélectionné par le joueur, et un int de colonne. Ceci permet à la fonction de modifier la colonne du vitrail en question et retourne le nombre de vitrails (soit les éléments de vecteur) non placer. |
| estComplete(int colonne) | Prend un argument un type int colonne et qui retourne vrais si celle-ci est pleine |
| EstEnConstruction | Prend en argument un type int colonne et retourne vrais si au moins une vitre correspondant à la colonne a été modifi/ |
| getColonne | Retourne le nombre de la colonne |
| Operator<< | Nous permet d’afficher le matrix vitrail |

**Class projetCSI2772**

Cette classe contient la fonction principale (main) de notre programme et nous permet d’exécuter notre jeu. Nous débutons par demander à l’utilisateur s’ils veulent choisir les conditions de jeu par défaut (le vitrail 5x7) ou définir leur propres rangées et colonnes. S’ils choisissent l’option de le faire eux-mêmes ils vont être demandés d’entrer des nombres de rangée et de colonne avant de passer à la prochaine étape, ce qui va appeler le constructeur approprié.

Ensuite le programme demande les noms des deux joueurs, et affiche l’état du joueur 1 et des lots (le vitrail du joueur 1 est généré aléatoirement et de même avec le lot). Comme spécifié, il n’y a aucune autre méthode dans la classe, mais nous utilisons un tableau de Joueurs pour alterner entre joueur1 et joueur2 lorsque c’est leur tour. Le programme les donne le choix entre 2 options : 1- déposer des vitres à la droite ou 2- le déposer et revenir à l’extrême gauche. De là ça leur demande de choisir une colonne de lot et une couleur et le programme enlève les couleurs du lot et le met soit dans un vecteur ou dans le surplus. Dépendamment de ce que le joueur à choisit il pourrait choisir une colonne du vitrail ou déposer les vitres ou ça se fera déposer automatiquement.

Le calcul des points pour le joueur se fait automatiquement à la fin de chaque ronde lors de l’affichage de l’état du joueur. Ensuite l’utilisateur peut peser sur n’importe quelle clé pour continuer, et ce sera maintenant le tour du joueur 2, qui effectuera les mêmes opérations. Chaque phase du jeu se déroule jusqu’à temps que le lot est vide, et à ce temps là le counter de jeu va s’incrémenter et le lot va se réinitialiser avec de nouvelles valeurs. Le joueur 2 pourra commencer avec un lot frais cette fois (les vitraux restent inchangés). Ceci va continuer jusqu’à temps que le programme vient à exécuter 4 phases, et à ce temps-là va comparer les points et déclarer un joueur gagnant.