Cyril Li
Adrien Lebron

FISE1-E

Mini-Projet : 2048

Rapport

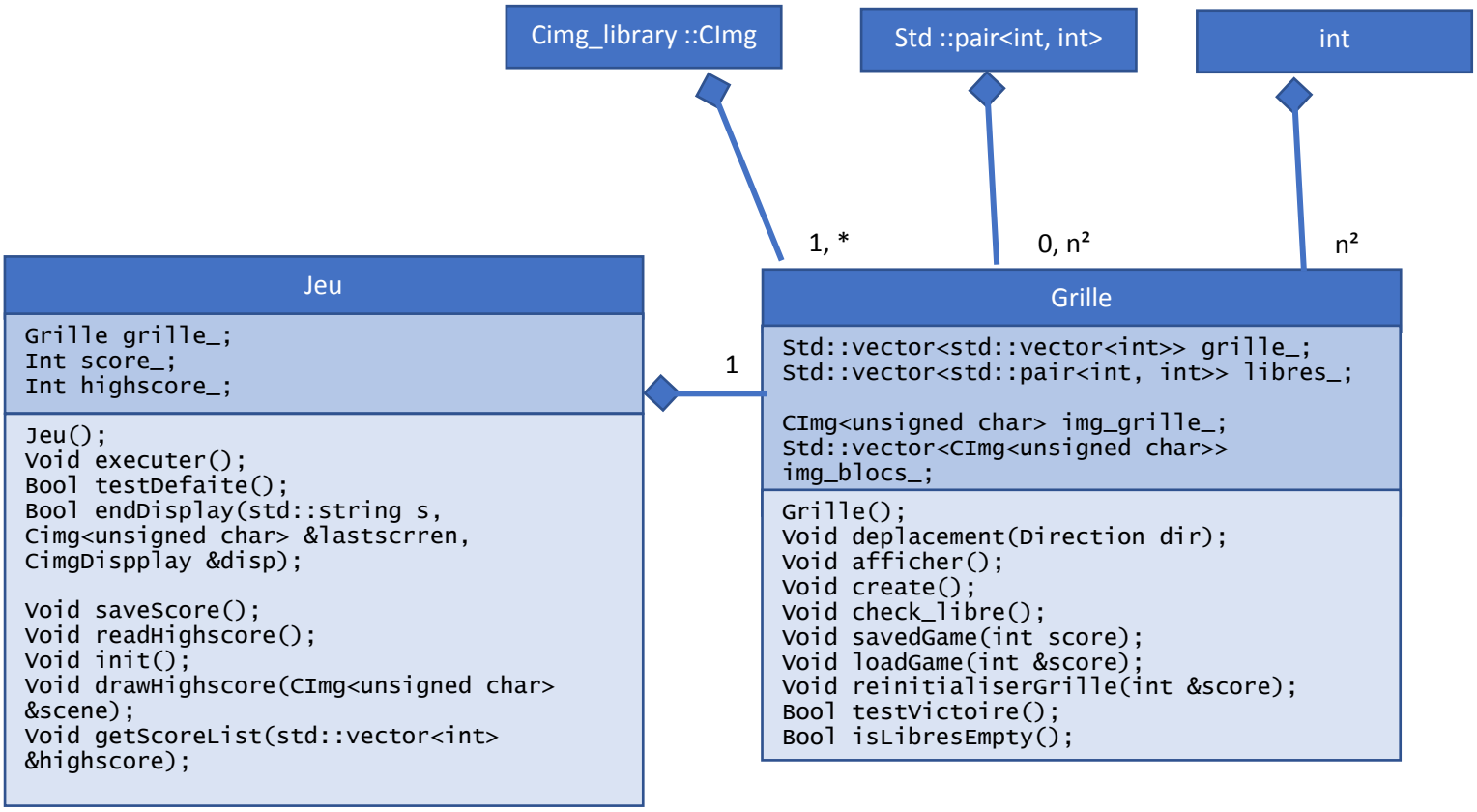
1 TABLE DES MATIERES

2 SPECIFICATIONS

- Génération d'une grille de 4*4 carrés
- Initialisation de la grille (2 blocs placés aléatoirement de valeur 2)
- Génération dans la grille des tuiles à un emplacement aléatoire dans les case disponibles (de valeur 2 ou 4)
- Affichage de la grille à l'aide de la bibliothèque CImg
- Contrôles avec les flèches : translate l'intégralité des tuiles dans la direction choisie
- Fusion des tuiles adjacentes de valeurs égales lors de la translation dans la direction choisie
- Détection quand plus aucun mouvement est possible.
- Incrémentation du score de la valeur des tuiles crées par la fusion.
- Le joueur gagne si une tuile à la valeur 2048.
- Sauvegarde de la partie quand le joueur quitte l'application
- Réinitialisation de la grille lors de l'appui d'une touche
- Chargement de la partie sauvegardée lors du lancement de l'application
- Sauvegarde des meilleurs score et affichage des 3 meilleurs scores.

3 DIAGRAMME DE CLASSE

n correspond à la taille du carré.



La classe Jeu gère la partie, il possède une grille, et il va gérer les différentes opérations que l'on effectuera sur la grille. La classe Grille gère la Grille, elle utilise un tableau d'entier à deux dimensions pour gérer les cases du tableau.

Nous n'avons pas créé de menu au démarrage car notre menu ne comporterait qu'un bouton reprendre la partie et un bouton commencer une nouvelle partie, nous n'avons donc pas jugé nécessaire d'alourdir notre code pour si peu de fonctionnalités. Par défaut, notre programme récupère la partie précédente, et si l'utilisateur souhaite commencer une nouvelle partie, il n'a qu'à appuyer sur R, ce qui est bien plus rapide que de saisir un chiffre puis appuyer sur Enter.

Nous n'utilisons pas non plus de classe Case ou Tuile, car l'utilisation d'un tableau d'entiers à deux dimensions permet de gérer notre grille de manière simple et efficace.

La fonction testVictoire() ne se trouve pas dans la classe jeu, car il est bien plus simple de parcourir la grille directement dans la classe prévue à cet effet que de la transmettre à la classe Jeu.

Pour les mêmes raisons, les fonctions SaveGame() et LoadGame() se trouvent dans la classe Grille.

La fonction readHighscore() permet d'avoir une évolution de l'highscore en temps réel, elle va lire le meilleur score en début de partie, et si le score actuel est supérieur à l'highscore du fichier highscore.txt, il affichera la plus grande valeur.

4 DETAIL DES FONCTIONS DES CLASSES GRILLE ET JEU

En commentaire les actions réalisées par les fonctions :

```
Classe Grille (Auteur Principal : Cyril Li)
/*
	Classe : Grille
	Résumé :  Objet qui va gérer le tableau de jeu (gestion et affichage)
*/
class Grille
{
	private :
		// Position dans le fenêtre de la grille
		size_t pos_x_, pos_y_;

		// Image de la grille
		cimg_library::CImg<unsigned char> img_grille_;
		// Images de tous les blocs
		std::vector<cimg_library::CImg<unsigned char>> img_blocs_;

		// Tableau 2 dims contentant l'état de la grille
		std::vector<std::vector<int>> grille_;
		// Vecteur de toutes les cases vides
		std::vector<std::pair<int, int>> libres_;

	public:
		// Constructeur par défaut
		Grille();
		// Gère le déplacement des blocs
		bool deplacement(Direction dir, int &score);
		// Affiche le tableau
		void afficher(cimg_library::CImg<unsigned char> &scene);
		// Ajoute des images blocs au cas où l'on dépasse la valeur maximale
		void addBlocs();
		// Créer de nouveau blocs dans la grille
		void create();
		// Recherche toute les cases libres
		void check_libre();
		// Enregistre la partie à la fermeture de la fenêtre
		void saveGame(int score);
		// Chargement de la partie précédente
		void loadGame(int &score);
		// Réinitialise la grille pour une nouvelle partie
		void reinitialiserGrille(int &score);
		//Teste la victoire (affichage d'une case à 2048)
		bool testVictoire();
		// Teste s'il reste des cases libres
		// Retourne Vrai s'il n'y a plus de de case libre, Retourne Faux sinon
		bool isLibresEmpty() { return libres_.empty(); };


};
```

Classe Jeu (Auteur principal : Adrien Lebron)

```
/*
Classe : Jeu
Résumé :  Objet qui va gérer le tableau de jeu (gestion et affichage)
*/

class Jeu {
	private:
		// Grille de jeu
		Grille grille_;
		// Score actuel
		int score_;
		int highscore_;

	public:
		// Constructeur par défault
		Jeu();
		// Exécuter la boucle de jeu
		void executer();
		// Vérifie si le jouer a perdu
		bool testDefaite();
		// Affiche l'écran de fin
		bool endDisplay(std::string s, cimg_library::CImg<unsigned char>
&lastscreen, cimg_library::CImgDisplay &disp);
		// Sauvegarde le score
		void saveScore();
		// Lis le plus gros score dans un fichier txt
		void readHighscore();
		// Initialise une partie
		void init();
		// Affiche les meilleurs scores à la fin d'une partie
		void drawHighscore(cimg_library::CImg<unsigned char> &scene);
		//Récupère les meilleurs score et les stocke dans un vector
		void getScoreList(std::vector<int> &highscore);

};
```