Chapitre 3

Réalisation et mise en œuvre

Ce chapitre concerne l'implémentation de la solution adoptée. La première section est consacrée à la description des outils de développement utilisés. On décrit, par la suite, les principales fonctions du module contrôleur du dialogue. On termine par la présentation de quelques interfaces graphique du jeu.

3 Réalisation et mise en œuvre

3.1 Présentation des Outils :

3.1.1 Qt :

Qt est une bibliothèque logicielle orientée objet et développée en C++ par la société Trolltech. Elle permet la portabilité des applications qui n'utilisent que ces composants par simple recompilation du code source. Les environnements supportés sont les Unix (dont Linux) qui utilisent le système de fenêtrage X11, Windows et Mac OSX. Qt est notamment connu pour être la bibliothèque sur laquelle repose l'environnement graphique KDE. Qt est sous licence GPL (General Public License) lorsque l'application développée était également sous GPL, pour le reste, c'est la licence commerciale qui entre en application. Cette politique de double licence a été appliquée uniquement pour Unix dans un premier temps, mais depuis la version 4.0 de Qt, elle est appliquée pour tous les systèmes.

3.1.1.1 Modèle objet de Qt :

La classe QObject est la classe de base de tous les objets de Qt utilisant des signaux et/ou des slots. Tous les objets Qt héritant de QObject possèdent un paramètre dans le constructeur appelé parent. Les QObject s'organisent dans des arbres d'objets. Quand vous créez un QObject avec un autre QObject comme parent, l'objet parent ajoute l'autre à sa liste d'enfants. Lorsque le parent sera détruit, il supprimera automatiquement ses enfants dans son destructeur. La gestion de la mémoire est, grâce à ce mécanisme grandement facilitée et les risques de fuites mémoire sont moindres. En résumé, les seuls objets à détruire avec delete sont ceux ayant été créés avec new et ne possédant pas de paramètre parent renseigné.

3.1.1.2 Qt Creator

Qt Creator est un environnement de développement intégré multiplateforme faisant partie du Framework Qt. Il est donc orienté pour la programmation en C++.

Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git et Mercurial ainsi que la documentation Qt. L'éditeur de texte intégré permet l'auto complétion ainsi que la coloration syntaxique. Qt Creator utilise sous Linux le compilateur gcc; il peut utiliser MinGW ou le compilateur de Visual Studio sous Windows.

3.1.1.3 Qt Designer

Qt Designer est une interface intuitive permettant de dessiner rapidement une fenêtre ou une boite de dialogue en utilisant la bibliothèque Qt. Cette application a également été réalisée par la société Trolltech.

3.1.2 OpenGL

OpenGL (Open Graphics Library) est un ensemble normalisé de fonctions de calcul d'images 2D ou 3D lancé par Silicon Graphics en 1992. Cette interface de programmation est disponible sur de nombreuses plateformes où elle est utilisée pour des applications qui vont du jeu vidéo jusqu'à la CAO en passant par la modélisation.

OpenGL permet à un programme de déclarer la géométrie d'objets sous forme de points, de vecteurs, de polygones, de bitmaps et de textures. OpenGL effectue ensuite des calculs de projection en vue de déterminer l'image à l'écran, en tenant compte de la distance, de l'orientation, des ombres, de la transparence et du cadrage.

L'interface regroupe environ 250 fonctions différentes qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives géométriques. Du fait de son ouverture, de sa souplesse d'utilisation et de sa disponibilité sur toutes les plates-formes, elle est utilisée par la majorité des applications scientifiques, industrielles ou artistiques 3D et certaines applications 2D vectorielles. Cette bibliothèque est également utilisée dans l'industrie du jeu vidéo où elle est souvent en rivalité avec la bibliothèque de Microsoft : Direct3D. Une version nommée

OpenGL ES a été conçue spécifiquement pour les applications embarquées (téléphones portables, agenda de poche, consoles de jeux...).

Plusieurs bibliothèques sont développées à partir d'OpenGL afin d'apporter des fonctionnalités qui ne sont pas disponibles dans la bibliothèque OpenGL elle-même :

- GLU
- GLUT
- GLUI
- GLEW
- GLEE
- GLFW
- GLM

3.2 Les principales fonctions du module contrôleur du dialogue

Le contrôleur gère le séquencement de l'interaction entre le Noyau Fonctionnel et l'interface. Il maintient un état lui permettant de gérer les modes d'interaction et les enchaînements d'écrans.

3.2.1 Le mécanisme des signaux et des slots :

Un principe propre à Qt qui est clairement un de ses points forts. Il s'agit d'une technique séduisante pour gérer les évènements au sein d'une fenêtre.

Par exemple, si on clique sur le bouton ''Nouvelle Partie '' de l'interface, on voudrait qu'une fonction soit appelée pour réagir au clic cette fonction crée une nouvelle partie au niveau du noyau fonctionnel.

Description :

Les signaux et slots sont une implémentation du patron de conception observateur. L'idée est de connecter des objets entre eux via des signaux qui sont émis et reçus par des slots. Du point de vue du développeur, les signaux sont représentés comme de

simples méthodes de la classe émettrice, dont il n'y a pas d'implémentation. Ces "méthodes" sont par la suite appelées, en faisant précéder "emit", qui désigne l'émission du signal. Pour sa part, le slot connecté à un signal est une méthode de la classe réceptrice, qui doit avoir la même signature (autrement dit les mêmes paramètres que le signal auquel il est connecté), mais à la différence des signaux, il doit être implémenté par le développeur. Le code de cette implémentation représente les actions à réaliser à la réception du signal.

C'est le MOC qui se charge de générer le code C++ nécessaire pour connecter les signaux et les slots.

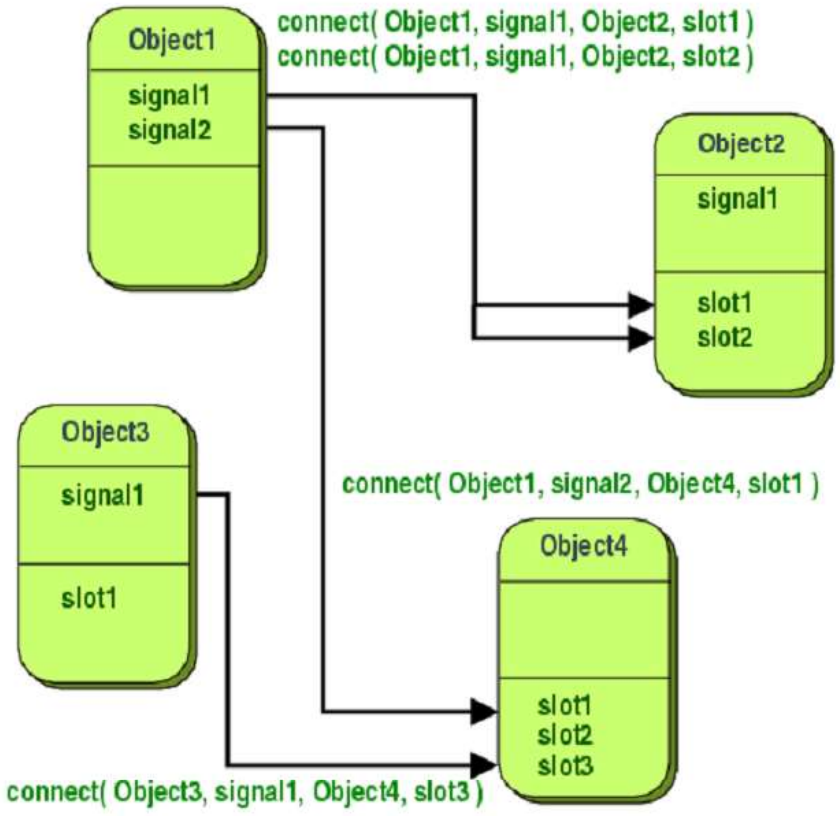

Figure 4 : Principe des signaux et slots

3.2.2 Composants :

- Variables :
  - Partie P ;
  - Fenetre_principal MainWindows;
  - Joueur Jr ;

- Jeu J;
- Jcourant

- Les SLOTS :
  - Creer_partie() : lors d'un clic sur nouvelle partie de l'interface cette fonction sera appeler pour créer une instance de Partie p.
  - Creer_Joueur(pseudo,clr) : lors d'un ajout de joueur avec son nom et couleur de pion depuis l'interface cette fonction sera appeler pour le créer au niveau de l'instance p qui vas appeler la fonction creerJoueur(name,clr) du noyau Fonctionnel.
  - PreparerPlateau() : après 2 joueur ou plus (max 4) lors d'un clic sur commencer la partie depuis l'interface cette fonction sera appeler pour la préparation et le commencement au niveau du Noyau Fonctionnel(NF) à travers l'instance P. A la fin elle émit le SIGNAL Interne joueurCourant().
  - Joueur_Courant() : cette fonction sera appeler à chaque fin de tour pour définir le joueur suivant par l'appellation de la fonction getJoueurs()[Jcourant] depuis NF.A la fin elle émit le SIGNAL S_envoyerJCourant(int) à la fenêtre Principal MainWindows.
  - LancerDe() : cette fonction sera appeler à chaque fois que le buttons Lancer dé est cliquer elle permet de récupérer la valeur du dé du NF après émit le SIGNAL S_valeurDe(int) à la fenêtre Principal MainWindows.
  - FinAnimationDe() : cette fonction sera appeler à la fin de l'animation du dé permet d'effectuer le jeu au NF et d'incrémenter ou initialiser à 0 la variable Jcourant après elle émit le SIGNAL S_valeurDeplacement(position,clr) à la fenêtre Principal MainWindows.
  - FinDepNormal() : à la fin du déplacement normal cette fonction test la case si elle est ordinaire alors émit le SIGNAL Joueur_Courant() ou une case (échelle ou serpent) qui permet un déplacement supplémentaire alors émit le SIGNAL S_DeplacementSupp(position,clr) à la fenêtre Principal MainWindows.
- Les SIGNALS :
  - S_joueurCourant()
  - S_envoyerJCourant(int)
  - S_valeurDe(int)
  - S_valeurDeplacement(int,int)
  - S_DeplacementSupp(int,int)

3.3 Présentation des éléments de l'interface

Lors du lancement du jeu la fenêtre principale affiche l'écran suivant :

Figure 5 : menu d'accueil du jeu

Cet écran contient un menu avec des boutons offrant à l'utilisateur le choix de jouer en multi-joueurs, ou consulter les Règles du jeu. L'arrière plan représente un damier animé dessiner en OpenGl.

Le choix de créer une nouvelle partie affiche à l'écran un nouveau panneau permettant à l'utilisateur de saisir les informations concernant les joueurs (leurs pseudo et la couleur de leur pion), le panneau est montré dans la figure ci dessous :

Figure 6 : panneau création des joueurs-vue1

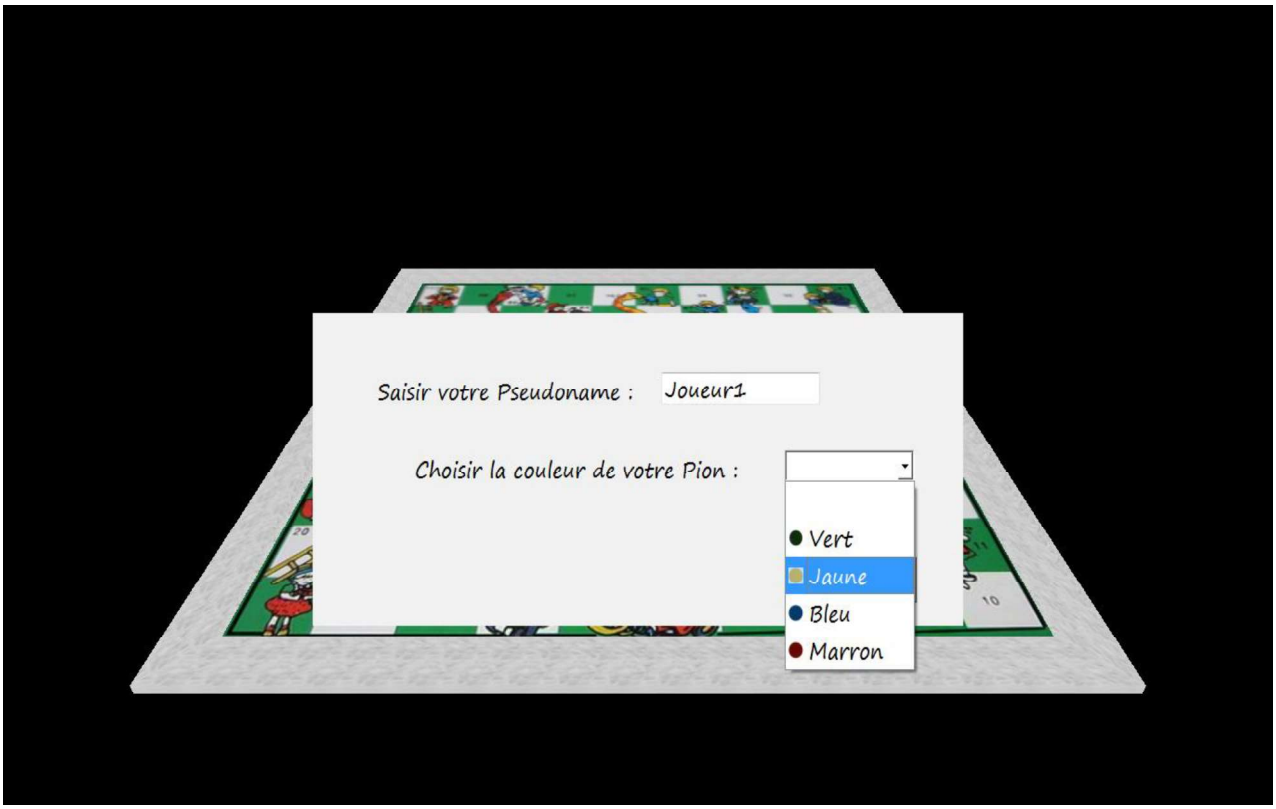

Figure 7 : Panneau creation des joueurs - vue2

Une fois un joueur créé, un nouveau panneau s'affiche à droite de l'écran contenant une fiche concernant le joueur nouvellement créé, cette fiche est montrée dans la figure suivante :

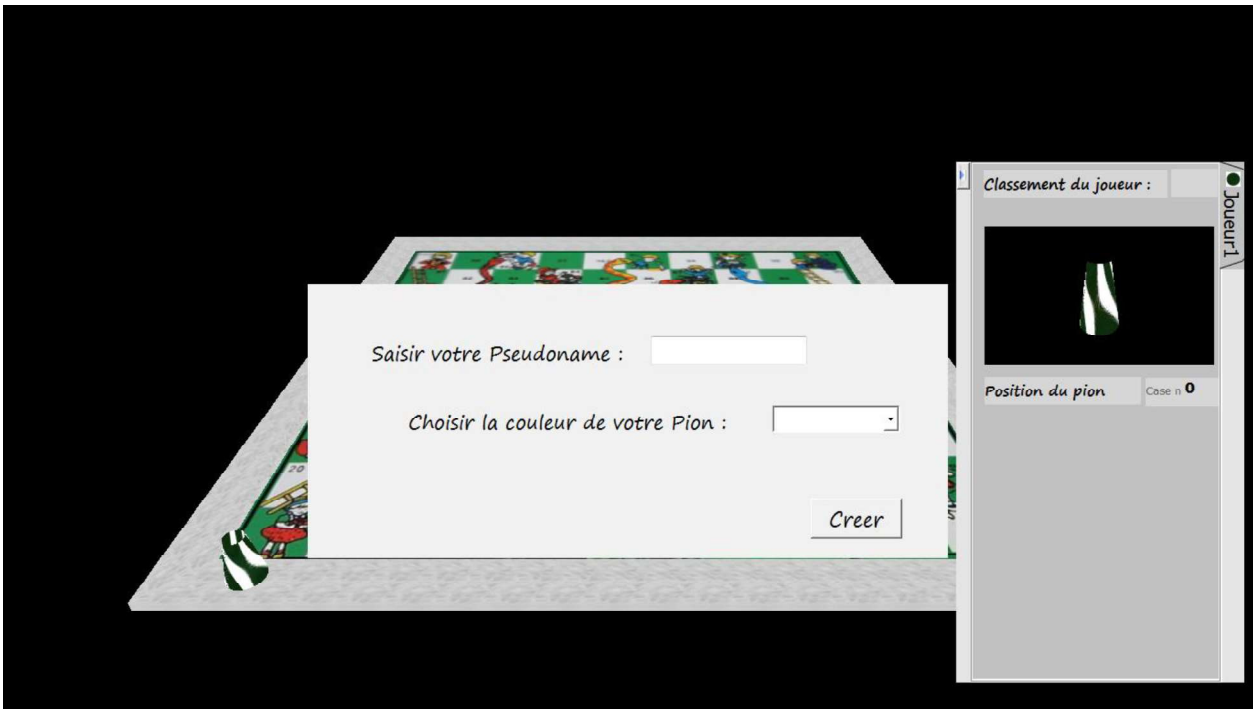

Figure 8 : Panneau informations joueur - vue1

Le Panneau à droite représente la liste des joueurs participant au jeu, c'est une liste qu'on peut masquer et afficher selon le besoin.

Figure 9 : panneau informations joueur - vue2

Après la création d'au moins deux joueurs, un bouton 'Commencer' s'affiche permettant à l'utilisateur de commencer la partie.

Quand l'utilisateur décide de commencer le jeu, une horloge calculant la durée de la partie apparait, ainsi qu'un bouton permettant au joueur courant de lancer le dé. La figure ci dessous donne un aperçu sur l'interface pendant le jeu :

Figure 10 : plateau du jeu

Au cours d'une partie le panneau à droite contenant la liste des joueurs porte des informations pour chaque joueur : son classement, la position de son pion. Comme montré sur la figure ci-dessous :

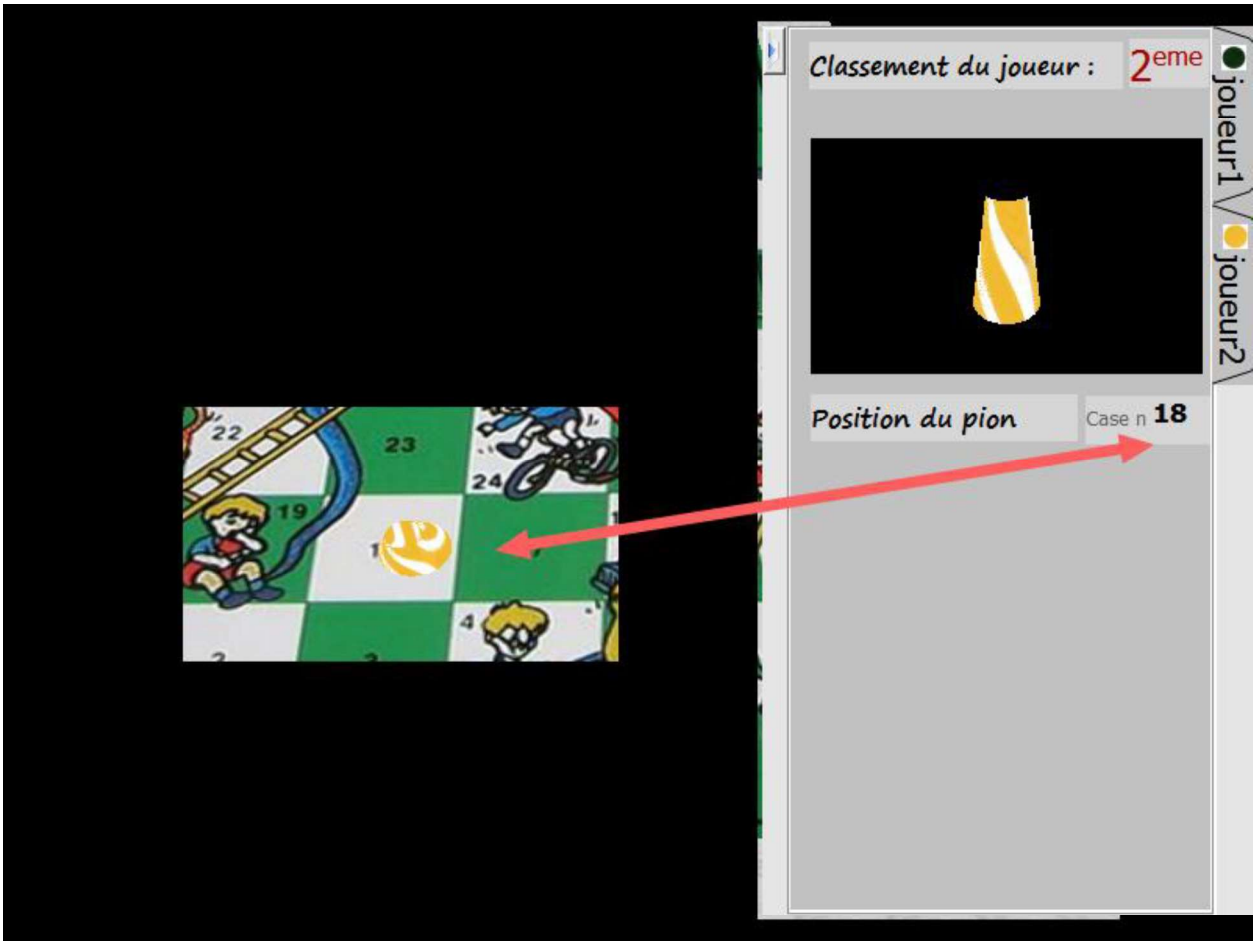

Figure 11 : panneau informations joueur - vue3

Conclusion

Ce chapitre a décrit la dernière étape du développement qui est celle de la réalisation et la mise en œuvre du projet. On a pu décrire l'architecture de la couche intermédiaire du jeu en présentant les principales méthodes et composantes. Et pour finir, nous avons donné un aperçu de l'application réalisée.