RAPPORT DE PROJET :

IHM

Mbaye Thiam

2022-2023

Programmation du Jeu **Bejeweled** en MFC Visual C++

Enseignant : Monsieur Olivier DERRIEN

Plan :

I. INTRODUCTION

A. Contexte du projet

Le projet consiste à développer un jeu appelé "Bejeweled". Il s'agit d'échanger des pierres « précieuses » pour créer des lignes ou des colonnes de trois ou plus de pierres de la même couleur. Le but est de marquer le plus de points possibles en supprimant le plus de pierres possibles.

Le choix de programmation pour le développement du jeu est l'utilisation du langage C++ et du Framework MFC. Cela permet de tirer parti des fonctionnalités de dessin et de gestion de la fenêtre pour une interface utilisateur plus conviviale notamment avec utilisation des boites de dialogue.

B. Cahier des charges

Le cahier des charges établit les fonctionnalités de base et avancées du jeu Bejeweled. Dans un premier temps, j'ai conçu la structure de classes du jeu et implémenté les méthodes de base (l'affichage du plateau et l'interversion des pierres). Le plateau de jeu est de taille variable, avec une valeur par défaut de 8x8, le nombre de catégories de pierres est également variable, avec une valeur par défaut de 6, mais aussi le nombre de pierres à l'initialisation du plateau. J'ai ajouté une boîte de dialogue pour personnaliser la taille du plateau, le nombre de pierres. L'affichage se fait par dessin dans la vue à base de cercles colorés, et l'initialisation du plateau de jeu est statique.

Dans une seconde étape, j'ai implémenté les fonctionnalités avancées du jeu. Les alignements de pierres provoquent la disparition des pierres, suivie par un décalage des pierres vers le bas pour combler les cases vides. J'ai ajouté un processus de contrôle pour afficher les différentes étapes de l'évolution au moyen d'un Timer. L'initialisation du plateau de jeu et l'insertion de nouvelles pierres est désormais aléatoire, et j'ai ajouté un comptage des points. Ensuite avant la partie image, j'ai ajouté plusieurs paramètres telles que la couleur du plateau, la taille des pierres(en pixel), l'espace entre les pierres, la forme des pierres, la position du plateau. Enfin, nous avons remplacé les cercles colorés par des images bitmap pour une meilleure esthétique visuelle.

II. Chapitre 1

A. Objectif

L'objectifs du projet est de développer un jeu de type "Bejeweled" en utilisant le langage C++ , le Framework MFC et nos connaissances en POO(programmation orientée objet). Le jeu doit fonctionner avec toutes les spécificités définis dans le cahier des charges.

B. Choix de programmation

Comme mentionné dans l'introduction, le choix de programmation pour le développement du jeu Bejeweled est l'utilisation du langage C++ et du Framework MFC. Cela permet de mettre en pratique nos connaissances en programmation orientée objet et en MFC Visual C++.

III. Chapitre 2 :

A. Échéancier de réalisation

Le projet Bejeweled a été achevé en totalité, en un temps total de 3 semaines. J'ai travaillé seul sur ce projet, et la liste des tâches réalisées ainsi que leur durée estimée sont présentées dans le calendrier ci-dessous :

Analyse et conception : 1 jour

- Analyse du sujet
- Conception de l'architecture du système et des classes
- Établissement du diagramme de classes (UML)

Programmation : 2 semaines

- Codage des classes (CPlateau et CBoitePlateau)
- Intégration des différentes fonctionnalités de base
    - Génération d'un plateau de jeu aléatoire
    - Interversion des pierres
    - Détection de pierres identiques
    - Suppression des pierres identiques
    - Descente des pierres restantes
    - Combler les cases vides
- Tests unitaires
    - Chaque fonctionnalité est testée avant de passer à la suivante.

Correction et optimisation : 4 jours

- Débogage des erreurs et des bugs
- Amélioration des fonctionnalités et des performances
    - Déplacement du plateau dans la vue
    - Effets visuels lors des clics
    - Choix de la forme des pierres (version 1)
    - Choix du nombre de Couleur …

Documentation : 1 jour

- Documentation du code avec Doxygen

B. Fonctionnement des fonctionnalités de base du jeu

- Interversion des pierres : Lorsque l'utilisateur clique sur deux pierres adjacentes pour les intervertir, l'application stocke temporairement les positions de ces deux pierres dans une variable. Ensuite, les positions des deux pierres sont interverties dans la structure de données contenant le plateau de jeu. Le programme vérifie ensuite si l'échange a permis de créer une suite de trois pierres identiques horizontalement ou verticalement. Si c'est le cas, ces pierres sont marquées pour suppression et un compteur de score est incrémenté.

- Suppression des pierres identiques : Après l'interversion des pierres, le programme examine le plateau de jeu à la recherche de groupes de trois pierres identiques alignées horizontalement ou verticalement. Les pierres de chaque groupe sont alors marquées pour suppression dans un vecteur. Si plusieurs groupes sont détectés, ils sont tous marqués pour suppression. Ensuite, le programme passe en revue toutes les pierres du plateau de jeu et supprime toutes les pierres marquées pour suppression.

- Descente des pierres restantes : Après la suppression des pierres, les pierres restantes sur le plateau de jeu descendent d'un nombre de cases égal au nombre de pierres supprimées. Les cases vides laissées par la suppression des pierres sont comblées par des pierres aléatoires générées par le programme. Le programme vérifie alors si la descente des pierres a créé un nouveau groupe de trois pierres identiques. Si c'est le cas, ces pierres sont également marquées pour suppression et le compteur de score est incrémenté. Le programme répète ces étapes jusqu'à ce qu'aucun groupe de trois pierres identiques ne puisse être créées par une interversion ou une descente de pierres.

- Combler les cases vides : Après la suppression des pierres et la descente des pierres restantes, le programme vérifie s'il reste des cases vides sur le plateau de jeu. Si c'est le cas, le programme comble ces cases vides en générant de nouvelles pierres aléatoires pour remplir les cases vides. Le programme vérifie alors si la nouvelle disposition des pierres a créé un nouveau groupe de trois pierres identiques. Si c'est le cas, ces pierres sont également marquées pour suppression et le compteur de score est incrémenté. Le programme

En résumé, l'interversion de pierres, la suppression des groupes de pierres identiques, la descente des pierres restantes et le comblement des cases vides sont les principales fonctionnalités du jeu Bejeweled. Ces fonctionnalités ont été implémentées en utilisant des structures de données appropriées et en combinant plusieurs algorithmes pour garantir le bon fonctionnement du jeu.

C. Gestion de projet

Comme je travaillais seul sur ce projet, la gestion de projet était assez simple. J'ai décidé de structurer le travail en fonction du cahier de charges, et de suivre les différentes étapes de développement de manière méthodique. J'ai également utilisé GitHub pour travailler en branches et sauvegarder régulièrement mes versions. Le processus de développement était assez itératif, avec des tests réguliers pour m'assurer que chaque fonctionnalité fonctionnait correctement avant de passer à la suivante. Globalement, le processus de gestion de projet a été assez simple, car le projet était de taille relativement modeste.

III. Chapitre 3

A. Travail réalisé

Le projet a été réalisé dans son intégralité en suivant les étapes définies dans le cahier de charges. Le jeu Bejeweled est fonctionnel et les fonctionnalités demandées ont été implémentées, ainsi que des fonctionnalités supplémentaires telles que le déplacement du plateau dans la Vue, des effets visuels lors des clics… Une documentation complète du code a été générée à l'aide de Doxygen, facilitant ainsi sa compréhension et son utilisation pour de futures améliorations. Le travail a été réalisé de manière autonome, en respectant l'échéancier prévu. Les résultats obtenus ont répondu aux attentes du projet et ont permis d'atteindre les objectifs fixés.

Deux versions du jeu ont été développées : une première version où le plateau est dessiné avec des formes géométriques, et une seconde version où le plateau est affiché avec des images bitmap.

Dans la première version, j'ai commencé par implémenter la partie définie dans le cahier de charges (l'implémentation des fonctionnalités de base du jeu telles que la sélection et l'interversion des pierres, la suppression des pierres identiques, le déplacement des pierres vers le bas pour combler les cases vides, etc.) et en y ajoutant autres fonctionnalités telles que la possibilité de changer la taille des cercles, leur forme (losange, diamant, etc.), et la couleur du plateau. J'ai également ajouté des effets visuels lors des clics.

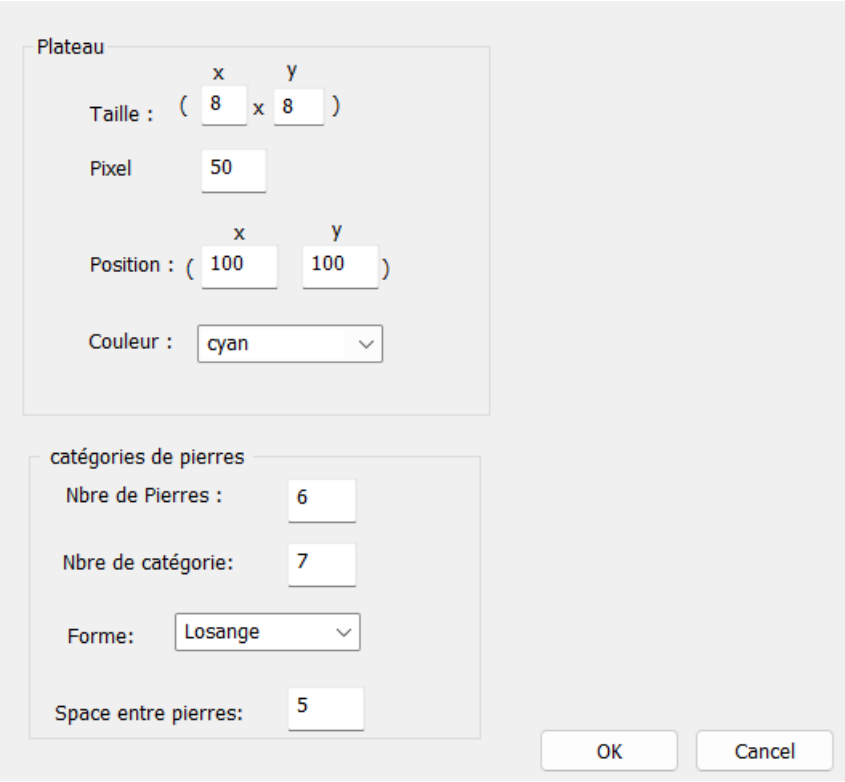

Figure 2 : **Boite de Dialogue de la version 1**

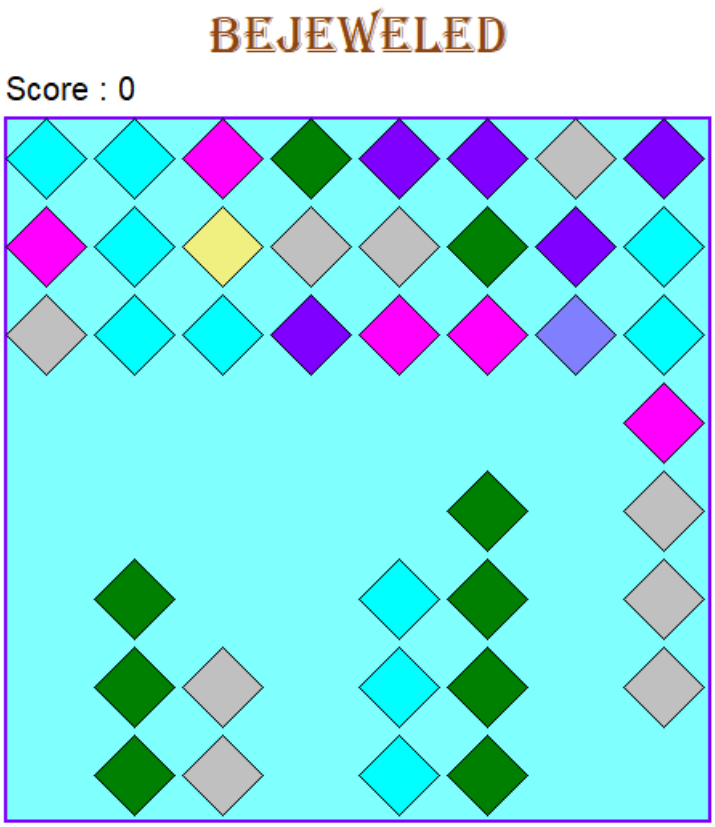

Figure 3 : Version 1 du Jeu

Dans la seconde version, j'ai travaillé sur l'implémentation de l'affichage du plateau avec des images bitmap. J'ai rencontré plusieurs difficultés pour cette partie. J'ai finalement réussi à résoudre ces problèmes en étudiant des exemples de code. Par contre j'ai supprimé plusieurs paramètres du jeu qui n'étaient pas compatible avec la version 2.

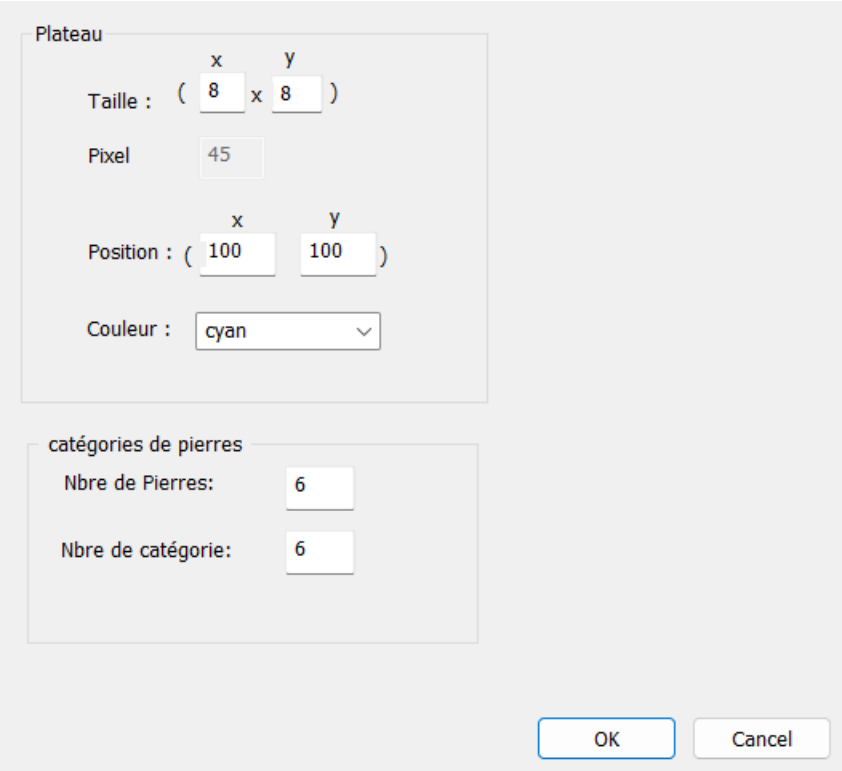

Figure 4 : **Boite de dialogue de la version 2**

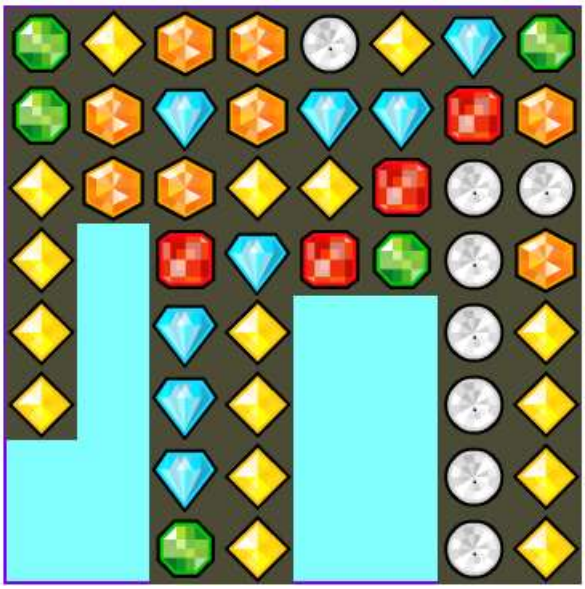

Figure 5 : **Version 2 du Jeu**

Dans les deux versions, j'ai utilisé une structure de données pour stocker la position et la couleur de chaque pierre, ce qui m'a permis de réutiliser cette structure dans toutes les fonctionnalités du jeu.

Enfin, une documentation complète du code a été générée à l'aide de Doxygen, facilitant ainsi sa compréhension et son utilisation pour de futures améliorations.

B. Description de la structure du programme

Le jeu Bejeweled est construit autour de quatre classes principales : **CPlateau**, **ProjetDoc**, **ProjetView** et **CBoitePlateau**.

- La classe CPlateau gère la logique du jeu, comme la gestion du plateau, l'interversion des pierres, la suppression et la descente des pierres, ainsi que la gestion des données pour chaque pierre (position et couleur). Elle est responsable de la mise à jour des données du plateau à chaque étape.

- La classe ProjetDoc fait le lien entre CPlateau et ProjetView. Elle contient le gestionnaire d'événements de la boîte de dialogue, qui permet d'initier la classe CPlateau avec les variables de la boîte de dialogues et ses méthodes. Cette classe permet également de charger et de sauvegarder les données du jeu.

- La classe CBoitePlateau est la classe de la boîte de dialogue et contient les paramètres variables de la boite de dialogue qui permet aux utilisateurs de sélectionner les options de jeu telles que la taille du plateau, le nombre de catégories de pierres, etc.

- La classe ProjetView est responsable de l'affichage du plateau, des pierres et du compteur de score. Elle contient les méthodes pour dessiner les éléments graphiques à l'écran et pour interagir avec l'utilisateur, notamment pour intervertir les pierres (`OnLButtonDown`), pour observer le dynamisme (`OnTimer`) , pour déplacer le plateau (`OnMButtonDown`), pour initier les images (`OnInitialUpdate`)…

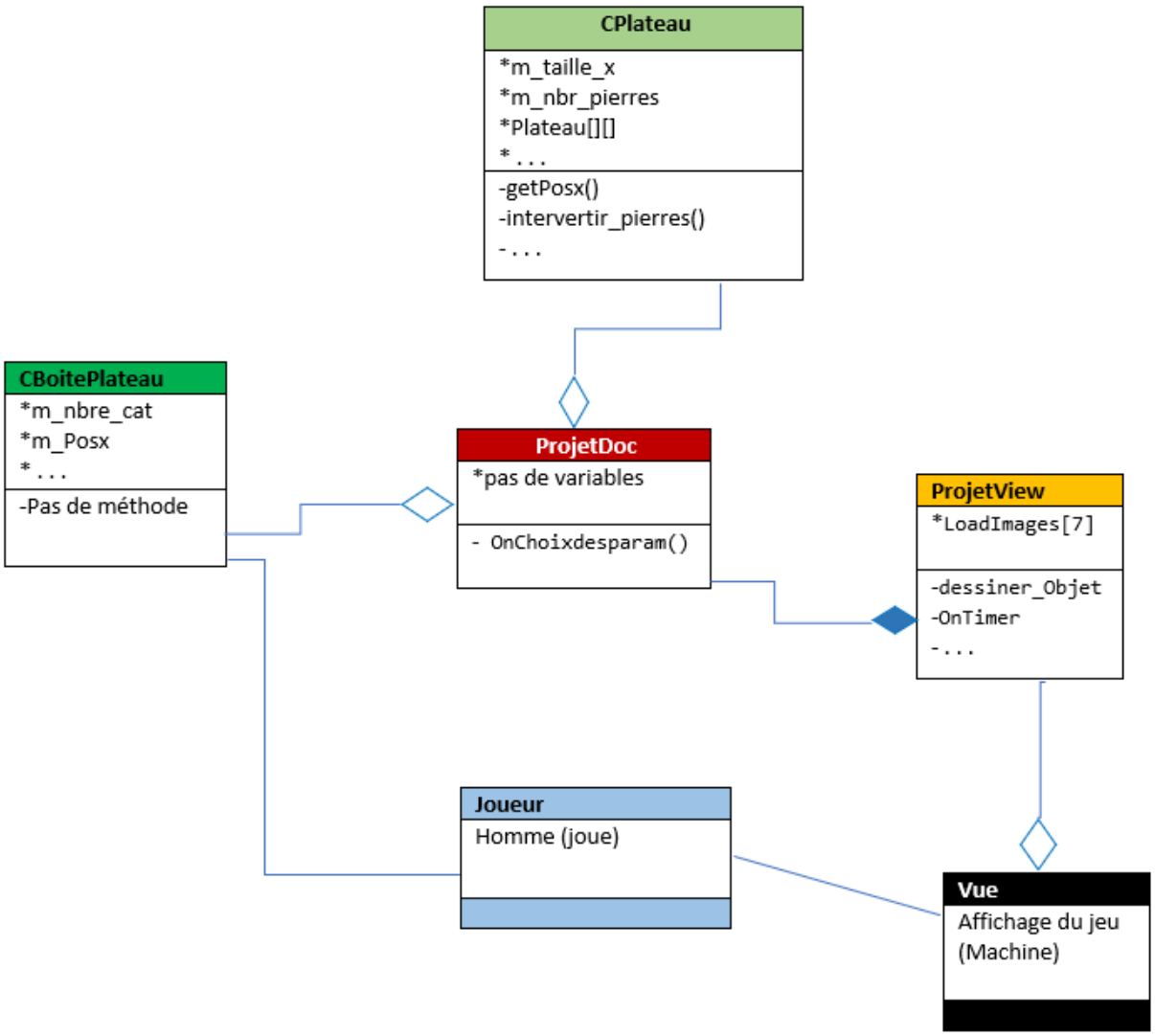

Figure 1 : UML du Projet (Unified Modeling Language)

Cette structure est cohérente avec l'approche de la programmation orientée objet, où les fonctionnalités du jeu sont encapsulées dans des classes qui interagissent entre elles pour produire le résultat final. La classe CBoitePlateau est importante, car elle est responsable de la gestion des événements de la boîte de dialogue, ce qui permet aux utilisateurs de sélectionner les options de jeu telles que la taille du plateau, le nombre de catégorie de pierres…

C. Problèmes rencontrés

Au cours du développement du projet, j'ai rencontré plusieurs problèmes. Tout d'abord, j'ai eu des difficultés à faire fonctionner correctement les fonctionnalités du jeu comme l'interversion des pierres, car je n'avais pas de structure de pierres, (quand je voulais intervertir que 2 pierres, toutes les pierres changeaient de couleur en même temps). J'ai résolu ce problème en créant une structure de données qui stocke la position et la couleur de chaque pierre, et en utilisant cette structure dans toutes les fonctionnalités du jeu (suppression, descente, comblement des cases vides, etc.), je n'ai plus eu ce genre de problèmes.

Ensuite, j'ai rencontré un problème de clignotement lorsque j'ai ajouté des «RedrawWindows » dans le programme, notamment dans la partie « OnButton » pour l'interversion et la suppression des pierres. J'ai résolu ce problème en utilisant le « OnTimer » de MFC pour la suppression et en enlevant les « RedrawWindows » . Ça redessine les pierres les unes par-dessus des autres, c'est dans l'idéal, mais le rendu final est propre et ça marche. J'ai voulu aussi utiliser le « buffering » pour l'affichage mais comme j'avais de connaissance solide sur le buffering, j'ai pas réussi.

Enfin, j'ai eu de gros problèmes avec les images Bitmaps. J'ai rencontré des bugs où les images ne s'affichaient pas ou j'avais le bug « PointBreak ». J'ai essayé d'utiliser CImage pour importer les images directement dans le code, mais cela n'a pas résolu le problème. J'ai finalement demandé de l'aide à ma professeure qui m'a montré un exemple de code bitmap, et cela a finalement fonctionné. Bien que cela m'ait fait perdre beaucoup de temps et m'ait énervé, j'ai finalement réussi à résoudre le problème.

En raison des difficultés rencontrées avec les images Bitmap, j'ai passé un temps considérable dans le développement de la première version du projet. Je n'étais pas certain de pouvoir le mener à bien la version améliorée.

Malgré ces problèmes rencontrés, j'ai réussi à terminer le projet en respectant toutes les étapes du cahier des charges.

D. Tests pour la validation du code

Plusieurs tests ont été effectués pour valider le bon fonctionnement du code. Tout d'abord, des tests unitaires ont été effectués pour chaque fonctionnalité implémentée dans le jeu, afin de s'assurer que le résultat obtenu est celui attendu.

Ensuite, des tests de validation ont été réalisés pour vérifier le comportement global du programme. Des parties ont été jouées dans différentes configurations (taille de plateau, nombre de couleurs, etc.) pour vérifier que le jeu fonctionne correctement et que toutes les fonctionnalités sont opérationnelles.

Des tests de performance ont également été effectués pour vérifier que le programme est capable de gérer les opérations de manière efficace, même pour des configurations de jeu très larges. Ces tests ont été réalisés en observant le temps de réponse du programme pour différentes tailles de plateau et de combinaisons de couleurs.

Tous les tests ont été concluants, ce qui indique que le programme fonctionne correctement et répond aux spécifications du cahier des charges.

V. Chapitre 4

A. Points Forts et Points faibles

Le projet Bejeweled a présenté de nombreux points forts, tels que la capacité de développer un jeu fonctionnel avec des fonctionnalités supplémentaires en utilisant des techniques de programmation orientée objet en C++. La structure MFC a été respectée et le jeu a pu être développé sans bug, avec une fluidité satisfaisante. L'utilisation de Doxygen pour générer la documentation du code permettra une meilleure compréhension du code pour les utilisations futures.

Cependant, certains points faibles ont été identifiés, tels que l'importance du temps consacré pour résoudre certains problèmes, en particulier ceux liés aux images Bitmaps. La conception de l'affichage des images a posé des problèmes, notamment avec l'affichage des images les unes par-dessus les autres, en l'absence de la méthode RedrawWindows.(ce qui peut être coûteux en mémoire). De plus, l'incapacité à vérifier si certaines méthodes sont coûteuses en mémoire a constitué un défi supplémentaire pour la qualité globale du projet. Enfin, l'absence de vérification scientifique pour les performances du programme a été un autre point faible.

B. Améliorations possibles

Pour améliorer ce projet, nous pourrions ajouter d'autres fonctionnalités telles que des effets sonores pour les mouvements et les combinaisons de bijoux, des animations pour les pierres qui tombent lorsqu'elles sont supprimées, des bonus de temps ou de score pour les combinaisons spéciales...

De plus, il est possible d'améliorer la performance globale du jeu en optimisant le code, en réduisant les opérations redondantes et en améliorant l'utilisation de la mémoire.

C. Conclusion

En conclusion, la réalisation de ce projet a été une expérience enrichissante en matière de programmation orientée objet et de gestion de projet. La résolution des problèmes rencontrés tout au long du projet a également renforcé mes compétences en résolution de problèmes et en recherche de solutions. Ce projet a également souligné l'importance de la documentation du code et de l'optimisation pour garantir un code de qualité.