RAPPORT DE PROJET

PROJET THÈME 2 : DÉVELOPPEMENT DU JEU MORPION EN C++ / SFML

Auteurs :
Othmane Haddou
Youssef Benharrous

Encadrant:
M. Dehos Julien

Jury:
Mme. Marion-Poty Virginie
M. Teytaud Fabien
M. Dehos Julien

Date de début : 06 Juin 2016
Date de fin : 24 Juin 2016

Année universitaire : 2015/2016

Table de matières :

Glossaire

Broadcast Réseau

En informatique, un Broadcast réseau désigne une méthode de transmission de données a l'ensemble des machines connectées sur un même réseau. Au contraire d'une connexion "point a point" (unicast) entre deux machines, qui utilise deux adresses IP (émetteur, destinataire), un Broadcast permet d'adresser des données a l'ensemble des machines d'un réseau, en utilisant une adresse IP spécifique au Broadcast. Cette dernière sera interceptée par toutes les machines du réseau local (ou sous- réseau).

SFML (Simple and Fast Multimedia Library)

La SFML est une bibliothèque multimédia écrite dans le langage de programmation C++. Elle permet d'afficher et de manipuler des éléments graphiques (images, formes, couleurs) au sein d'une fenêtre sur l écran de l'utilisateur. Elle permet également de capturer les évènements clavier et souris de l'utilisateur et d'y réagir. Enfin, la SFML dispose d'un module audio, permettant de jouer des sons ou des musiques, ainsi que d'un module réseau pour connecter plusieurs programmes entre eux.

Introduction

Dans le cadre de nos études pour la 3ème année en Licence Informatique, il nous a été demander de réaliser un projet sous l'encadrement de M. Dehos.

Tous deux attirés par la programmation orienté objet, nous avons choisi le projet de d'un « Jeu Morpion». Ce dernier nous permet en effet d'acquérir de nouvelles connaissances en C++, d'approfondir celles acquises durant les précédentes formations mais également de mesurer la difficulté de la mise en place d'un jeu qui sera réellement opérationnelle.

1 Présentation du projet

1.1 - Contexte général

Le jeu Morpion est un jeu avec une interface simple et intuitive qui est un jeu de pions se jouant a deux sur un damier de neuf cases (3x3). Chaque joueur pose a tour de rôle l'un de ses marqueurs. Le vainqueur est celui qui réussit a aligner trois de ses marqueurs horizontalement, verticalement ou en diagonale. Le but est de réaliser le jeu Morpion mais plus simple afin qu'on puisse effectuer une partie en Multijoueur, qui nous permettra d'afficher les scores de chacun des joueurs.

1.2 - Intérêt et contraintes du projet

Afin de créer le jeu qui réunit l'ensemble de ces fonctionnalités il nous est nécessaire de chercher des outils et de développer des scripts permettant d'analyser et de traiter tout sorte de fonctions nécessaires dans un jeu Morpion, en prenant en compte le fait que l'interface graphique doit être facile à manipuler.

1.3 - Besoins et priorités

Le besoin principal est de pouvoir développer une solution complète avec les outils et les connaissances acquises en C++. L'objectif est d'apprendre à utiliser la bibliothèque SFML étant une interface de programmation destinée à construire des jeux vidéo ou des programmes interactifs.

Ce jeu doit être doté des commandes de base :

- Faire jouer 2 joueurs humains au jeu du Morpion .
- Remplir la case correspondante après avoir vérifié qu'il s'agit d'un emplacement du plateau non encore coché.
- Passer la main a l'autre joueur.
- Le jeu se termine si un des joueurs gagne ou s'il n'y a plus de cases libres pour jouer.

2- Conception

Pour la conception de notre jeu nous avons eu recours au langage de modélisation objet unifié UML dans le but de bénéficier de ce standard incontournable.
Dans notre phase de conception nous nous sommes intéressés à :

- Diagramme de cas d'utilisation
- Diagramme de classes

2.2 - Diagramme des cas d'utilisation

La première information importante est l'existence d'un seul acteur dans le jeu, à savoir le joueur qui va effectuer la partie.

Définissons ainsi les cas d'utilisation possibles :

- Acteur : Joueur

- Cas d'utilisation :
  - Lancer une nouvelle partie.
  - Choisir le mode du jeu (Contre IA ou JoueurHumain);
  - Poser un pion sur la grille;
  - Voir le score du gagnant.

C'est ainsi que nous obtenons notre diagramme de cas d'utilisation suivant :

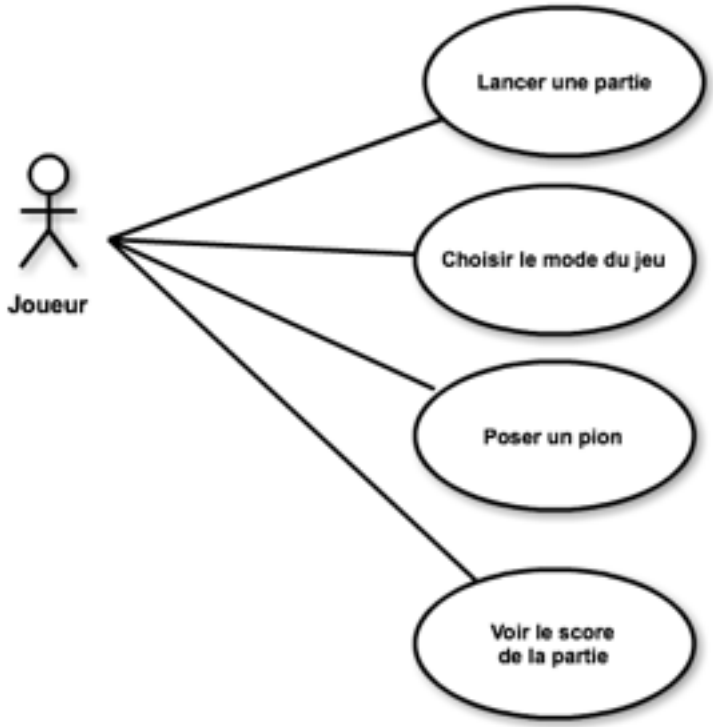


Figure 1.1 – Diagramme de cas d'utilisation du modèle

2.2 - Diagramme de classes

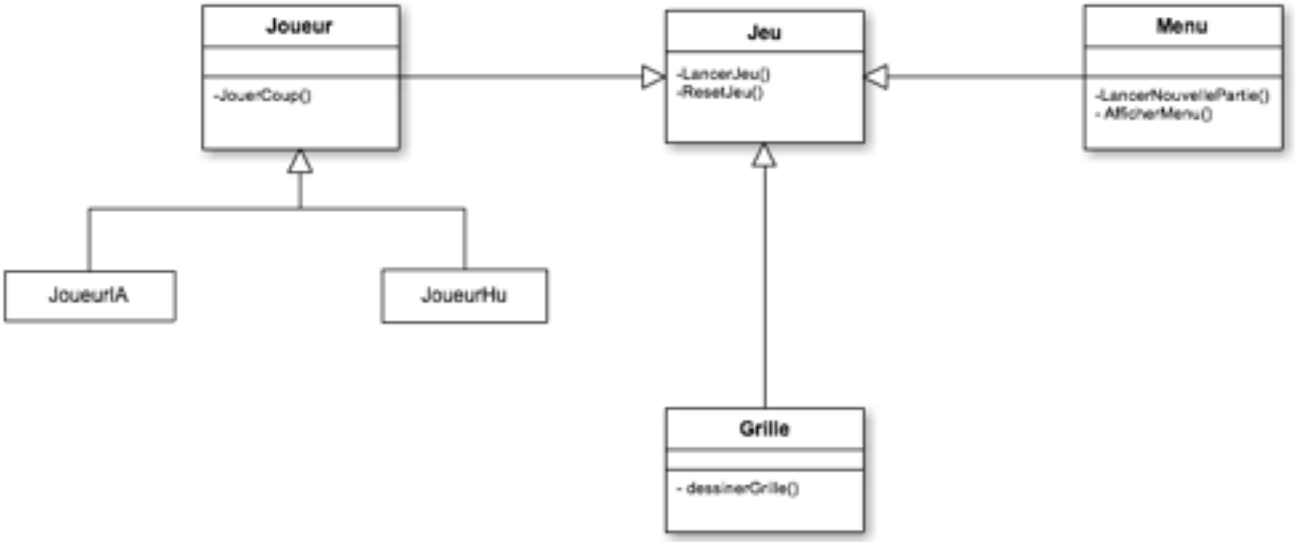
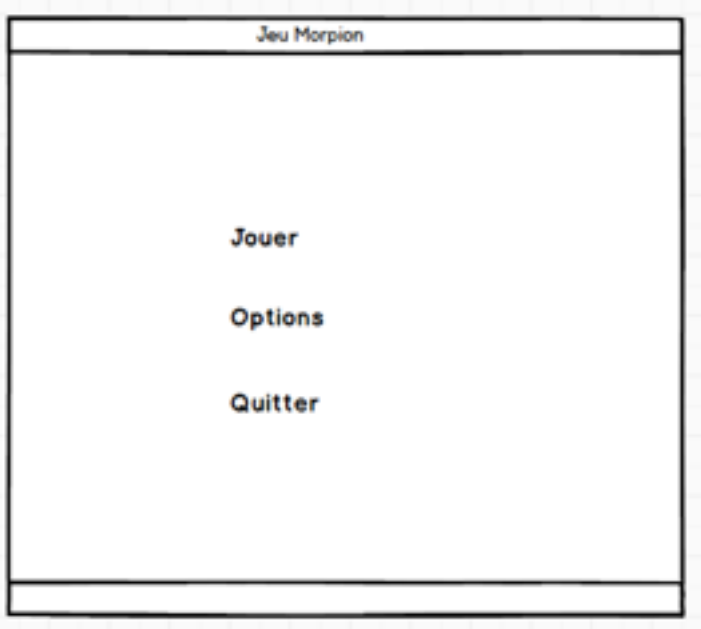


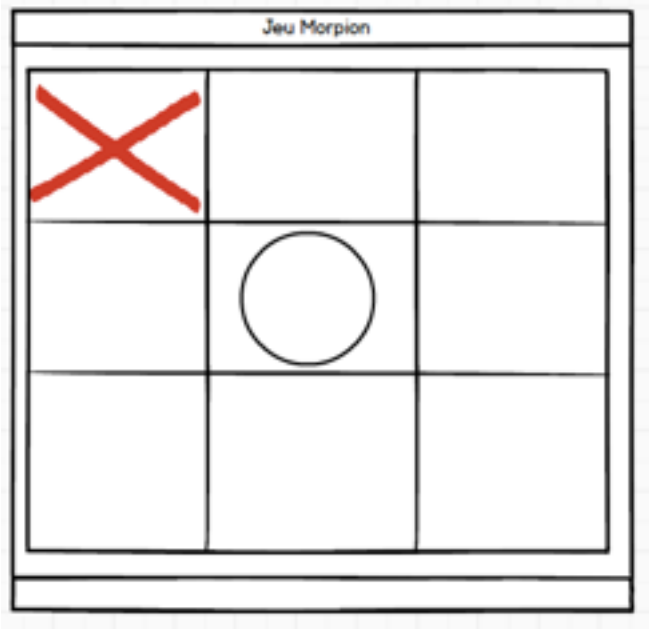
Figure 1.2 – Diagramme de classes du modèle

2.4 - Maquettage

De cette phase d'analyse a découlé la conception de notre jeu. Après avoir déterminé les scénarios d'usages nous nous sommes mis à créer la maquette de notre jeu afin de voir si toutes les possibilités décrites dans nos scénarios étaient bien réalisables dans notre application finale. Ainsi, à l'aide de l'outil Balsmiq, nous avons réalisé la maquette de notre jeu présentée ci-après.



Maquette 1.1 – Menu du jeu



Maquette 1.2 – Grille du jeu avec les pions

3 Choix techniques

3.1 - C++

Notre projet ,ayant la particularité d'être un jeu, et donc de devoir fonctionner en temps réel.la simulation du jeu doit idéalement être affichée plus rapidement. Par ailleurs, ce langage est celui que nous avons étudié lors de notre second semestre de Licence 3 Informatique: Le projet était donc un moyen complémentaire de mettre en pratique les apprentissages théoriques vus en cours.

3.2 - SFML

Le projet étant un jeu graphique, multi-joueurs, et multi-plateformes, le développement a été fait de plus que le langage C++ , a l'aide de la bibliothèque SFML (***Simple and Fast Multimedia Layer***), qui est particulièrement adaptée pour notre projet pour plusieurs raisons. Elle est écrite dans le langage de programmation C++, multi-plateformes qui permettra le fonctionnement du projet sur plusieurs systèmes d'exploitation, sans oublier l'exigence du client par rapport au choix du langage.

La SFML est constituée de **cinq modules** permettant de faire le lien avec votre hardware :

- Le module système permet notamment de gérer les variables temporelles, les flux de données et les threads.
- Le module de fenêtrage, comme son nom l'indique, s'occupe de la gestion des fenêtres mais également de tous les événements en général, notamment concernant les entrées (clavier, souris...).
- **Le module graphique** permet exactement ce que l'on imagine : dessins, sprites, textures, entités, shaders, il sera utiliser durant le projet et permettra d'afficher les éléments graphiques à l'écran(images, formes, couleurs), ou même les événements clavier et souris de l'utilisateur qui va interagir avec le jeu.

- Le module audio qui s'occupe du son.
- Le module réseau permet de faire communiquer votre programme : paquets, sockets, mais aussi FTP et requêtes HTTP, ce dernier servira à la connexion des joueurs entre eux.

3.3 - Git

Dans le développement de logiciels, Git est un contrôle de version distribué et système de gestion de code source (SCM). Nous avons utilisé Git pour synchroniser notre projet plus rapidement qu'avec d'autres supports tels que la clé USB et aussi pour pouvoir mettre des versions de l'avancement du projet.

En effet, nous avons passé beaucoup de temps pour effectuer les premières dépôts du projet, comme c'est un nouveau outil qui permet de mettre en place un suivi des projets avec des dépôts consécutifs.

4 Réalisation

4.1. Présentation du jeu

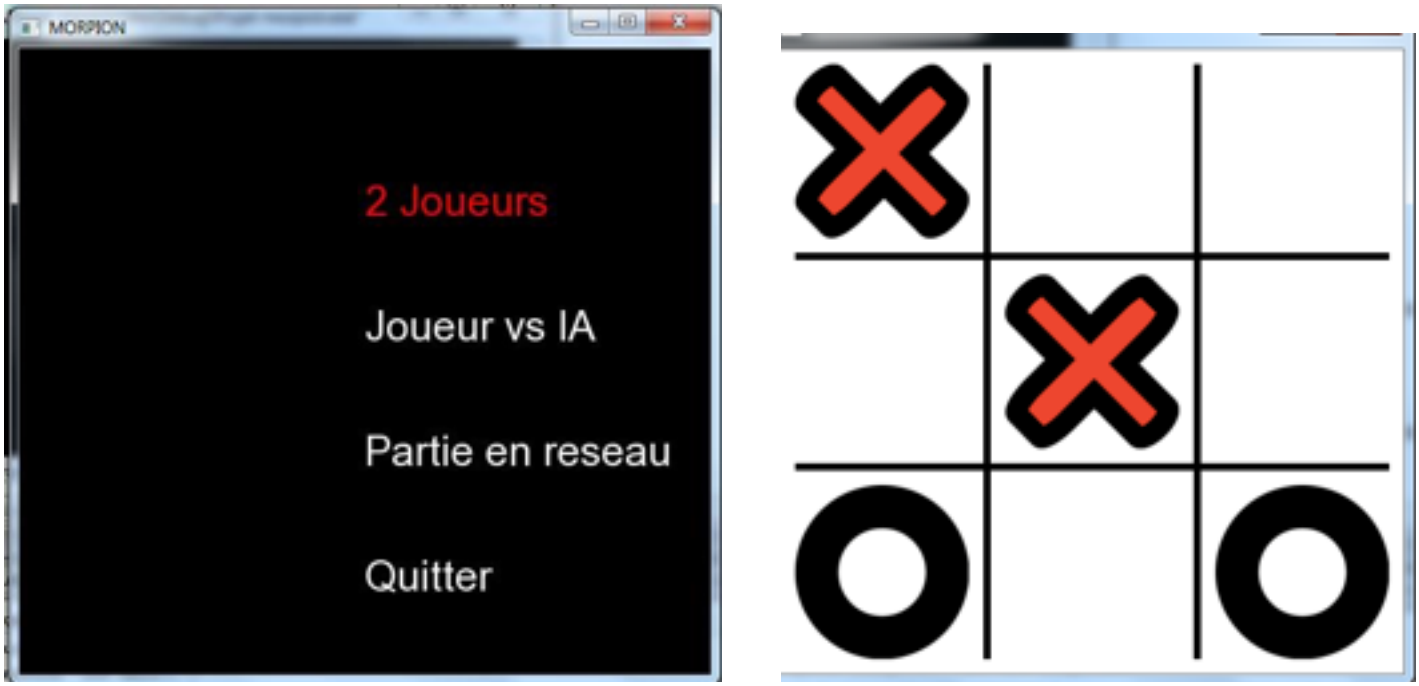


Figure 2.1 – Capture écran de l'interface du jeu

La but de la définition de l'interface est de doter une fenêtre contenant un menu dans lequel viendront se placer les composants d'une nouvelle interface qui est la grille du jeu. Cet emboitement de contenants les uns dans les autres permet de définir des interfaces aussi complexes que nécessaire. Le contenant définit la façon dont seront présentés les éléments . Le placement des éléments eux-mêmes dans le contenant est régi par un objet de placement que l'on associe au contenant. Après avoir créé une classe Menu, il est nécessaire de créer des composants du menu pour différencier le type de partie que ça soit contre un joueurHumain ou joueurIA.

Les deux premières phases de la réalisation de ce projet ont été le fruit de très longues réflexions et de plusieurs recherches qui ont servi à concevoir le jeu Morpion avec la bibliothèque SFML.

La classe de base est la classe Jeu qui permet de réaliser des interfaces en devenant le réceptacle ou viendront se placer les divers composants d'interface en ajoutant des méthodes propres à la gestion de notre fenêtre.

4.2 - Gestion des événements

Jusque là, nous avons pu créer une interface et réussit à intégrer un menu qui permet de choisir le type de partie que veut le joueur.Toutefois, il manque un aspect important au jeu : l'interaction avec le joueur.
Cette interaction se fait dans notre cas par le biais

- du clavier
- de la souris
- ou d'interactions avec l'interface graphique comme pour la fermeture de la fenêtre

C'est là qu'intervient la gestion des événements. comme SFML utilise un système de *polling* pour récupérer les évènements. et définit les types d'événements dans la structure sf::Event comme pour traiter les cliques de souris, les touches du clavier

4.3 - Partie Réseau : Multijoueur

SFML nous fournit à l'aide des sockets avec deux types de base: TCP et UDP. TCP. Ces deux protocoles sont capables d'envoyer et de recevoir des données, mais elles sont fondamentalement différents les uns des autres, dans notre cas, on a choisi de travailler sous le protocole TCP

Sockets TCP

TCP est un protocole basé sur la connexion, ce qui signifie que avant que les données peuvent être échangées, une connexion doit être établie en ayant une application qui tente de le lancer (un client) pour se connecter à une autre application qui est activement en attente pour les connexions

Tout d'abord, nous créons une instance de sockets TCP de la partie client. Sa méthode de connexion est appelé, avec trois arguments:
• Le premier argument est de type sf :: Connexion avec Adresse IP
• Le deuxième argument est le numéro de port.
• Enfin, nous avons le troisième argument, qui est totalement facultatif. c'est la valeur de délai après lequel la prise doit abandonner et jeter une erreur. Si cet argument est pas fourni, le système d'exploitation la valeur de temporisation par défaut est utilisé.

Pour la partie serveur Pour accepter une connexion sur le côté serveur en utilisant le protocole TCP, une classe spéciale est utilisée: sf :: TcpListener. Il doit être lié à un port spécifique et ne peut pas envoyer ou recevoir des données

Finalement, après avoir détaillée le fonctionnement la partie réseau du jeu, un nouveau problème se pose : faire fonctionner l'ensemble des composants du jeu de manière fluide et contrôlée. Pour cela, nous avons décidé d'utiliser un système de trends. Ces derniers permettent de faire fonctionner en parallèle la simulation de jeu et la partie réseau décrite précédemment., c'est le contrôleur qui va lancer la méthode loop du networkClient dans un nouveau thread. Des lors, le contrôleur va, dans sa boucle de jeu, afficher le jeu, le faire tourner, mais aussi récupérer les informations issues de la partie réseau (clientData et serverData) pour les intégrer au jeu dans le tour de boucle suivant.

4. Bilan

4.1 - Déroulement du projet

En fonction du temps qui nous était imparti, ainsi d'arrangement, il semble évident que nous n'avons pas pu réaliser tous les objectifs que nous nous étions fixés. Nous avons fait le choix de développer certains modules de façon rapide et non évolutive. Par exemple, nous aurions aimé réaliser que le joueur puisse choisir le pion de son choix, aussi pouvoir faire un bon affichage du score.

Nous aurions également apprécié de développer une fonctionnalité qui permettrait de traiter un joueur IA avec des différents niveaux de difficultés .Le manque du temps, et les taches attribuées étant supérieur à l'effectif du groupe , nous avons préféré se consacrer de plus sur la partie réseau pour que le jeu soit plus évolutive.

L'importante phase de conception s'est avérée bénéfique pour la réalisation de ce projet. En effet, elle a permis un développement plus rapide et une structuration importante des différents composants.
Cependant, certaines options prévues pendant la conception n'ont pas pu être implémentée. Toutefois, cela n'empêche pas un bon fonctionnement du produit.

Nous aurions souhaité faire une interface encore plus précise avec plus d'options accessibles.

4.2 - Résumé du projet

Dans le cadre du Travail d'Etude en Licence 3ème année informatique, on a eu comme projet, le développement d'un jeu . Cette plateforme, implémentée en C++ permet de créer un jeu exécutable en respectant toutes les règles du jeu, et en développant une interface graphique . L'autre finalité de cette plateforme est la possibilité de faire une partie multijoueur en réseau tant au niveau de la rapidité que de l'efficacité. Le fait de développer une telle plateforme s'est avéré utile car dans la plupart des cas, le développement d'une interface graphique sous C++ s'avère important qui aide à mieux maitriser les connaissances acquis en cours. On a donc étudié un ensemble d'outils nécessaire au développement pour le développement des procédures qui pourront gérer ce jeu.

4.3 - Réalisation des objectifs

Fonctionnalités	**Réalisation**
Création de l'interface du jeu	oui
Jeu fonctionnel	oui

Respect des règles du jeu	oui
Affichage du score	oui
Réalisation de la partie réseau	En cours
Réalisation du joueur IA	Non traité
Option : Retour au menu	En cours

4.4 - Conclusion pour les projets futurs

Ce projet bénéfique pour chacun des membres du groupe car elle nous a permis d'illustrer les notions acquises au cours de ce module.
Dans un premier temps, nous avons fait une étude les besoins fonctionnels du jeu, cerné les différentes contraintes et entamé la phase d'analyse et de conception. Ensuite nous avons réalisé notre jeu via le langage de programmation C++ en se servant du SFML comme bibliothèque graphique, pour finir nous avons testé et analysé notre jeu finale.

4.5 - Bibliographie / Webographie

Site	Type de site	Information recherchée
http ://www.sfml-dev.org/index-fr.php	Site officiel	Librairie SFML
http ://www.cplusplus.com/	Site d'information	Librairies standards C++
http ://www.github.com/	Site officiel	Dépot du projet