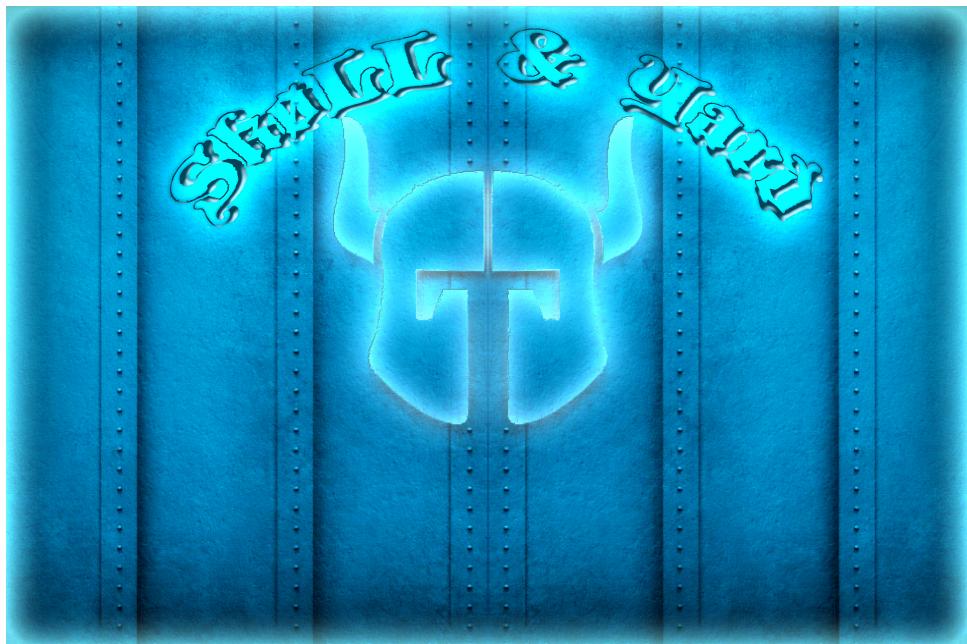


Projet Skøll&Yard - Rapport



Étudiants : PREVOT Clément MAES Benjamin TRABELSI Nadir BRIMEUX Benoit

Encadrant : DEHOS Julien

Date de début : 01 Juin 2015

Date de fin : 19 Juin 2015

Objet du projet : Projet Skøll & Yard : Le Puissance 4 en C++



19 juin 2015

Table des matières

1 Présentation du projet	2
1.1 Contexte du projet	2
1.2 Analyse de la demande	2
1.3 Spécifications	2
2 Réalisation	3
2.1 Présentation	3
2.1.1 Installation et lancement du jeu	3
2.1.2 Détail des menus	3
2.1.3 Règle du jeu	4
2.1.4 Spécifications	5
2.2 Architecture générale	6
2.3 Gestion du réseau	7
2.3.1 Quelques explications	7
3 Bilan	9
3.1 Déroulement du projet	9
3.1.1 Plannification	9
3.2 Réalisation des objectifs	9
3.3 Conclusion pour les projets futurs	10

1 Présentation du projet

1.1 Contexte du projet

Dans le cadre du projet de fin d'année de Licence 3 Informatique, et pour la validation du diplôme, il nous a été proposé trois grandes thématiques. Nous avons choisi le thème n°3, à savoir développer un jeu en réseau, de type morpion, dames, pong etc, à l'aide des technologies C++, pour le langage informatique, et SFML, une librairie complète et puissante pour ce genre de projet, qui concerne l'interface graphique ainsi que des éléments de réseaux.

1.2 Analyse de la demande

Après mûres réflexion nous avons choisi d'implémenter un jeu de Puissance 4. Ce choix ce justifie par le fait qu'il ne fallait pas un jeu trop complexe à coder sachant le travail qu'apporte l'interface graphique et le développement réseau dans un court laps de temps.

1.3 Spécifications

Les spécifications concernant le projet sont très concises et sont énumérées de la façon suivantes :

1. Développer un jeu en réseau : notre choix s'est donc tourné sur un jeu de Puissance 4
2. Architecture réseau avec au choix un modèle client/serveur ou peer to peer : ici nous avons avantage le modèle client/serveur connu pour être plus simple à implémenter que le peer to peer, mais c'est aussi dans ce modèle que nous avons le plus d'expérience.
3. Développer une interface graphique complète : il faut qu'elle soit simple et ergonomique
4. Utiliser les technologies C++ et SFML

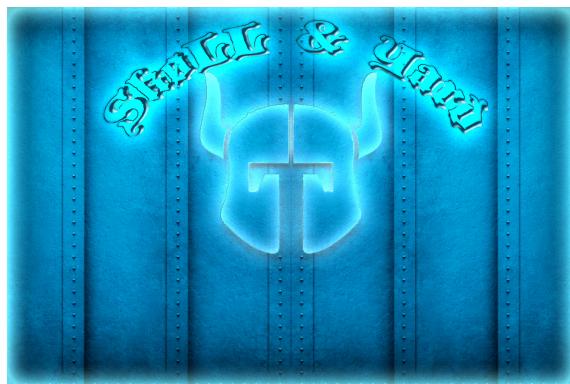
2 Réalisation

2.1 Présentation

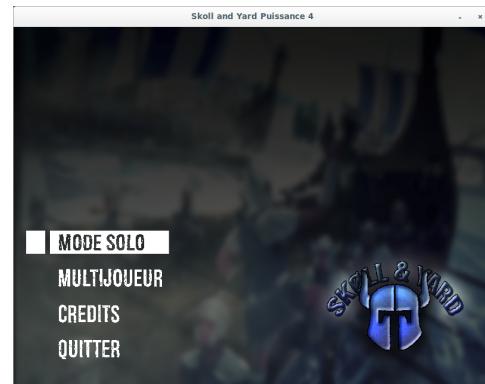
2.1.1 Installation et lancement du jeu

Pour installer le jeu, rien de plus facile :

- Après avoir téléchargé l'archive du jeu, placez vous dans le répertoire où vous souhaitez l'installer et décompressez l'archive
- Ouvrir un Terminal et faites un simple "make" et "make serveur"
- Lancer le serveur sur un terminal annexe au moyen de la commande suivante : ./bin/LaunchServer.out
- Lancer le jeu au moyen de la commande suivante : ./bin/Game.out Vous arriverez à l'écran d'accueil, appuyer sur n'importe quelle touche de votre clavier afin d'accéder au Menu du jeu.



(a) écran d'accueil



(b) menu principal

FIGURE 1 – Accueil du jeu

2.1.2 Détail des menus

Pour valider l'accès aux différentes parties du menu, cela se fait au moyen d'un clic gauche. Le menu se compose de la manière suivante :

1. **Mode Solo** : lancement d'une partie contre une IA "random" (joue au hasard). Une partie est gagnée en deux ou trois manches (exemple : si le score est de 2-0, la partie s'arrête). On peut quitter la partie à tout moments en appuyant sur echap.
2. **Mode Multijoueur** :
 - créer une partie local : lance une partie joueur contre joueur sur le même ordinateur.
 - rejoindre une partie : permet de se connecter au serveur en entrant l'adresse et le port du serveur (xxx.xxx.xxx.xxx/port), qui sont indiqués sur le terminal annexe où le serveur a été lancé. C'est ce mode qui permet de jouer une partie sur deux pc distincts, connectés sur le même réseau (ne fonctionne pas sur le réseau de l'Université, mais fonctionne très bien sur un réseau type "domicile"). Testé ici au moyen de la fonction modem d'un smartphone.
 - retour à l'écran titre : permet de retourner au menu principal.
3. **Crédits** :
 - défilement des noms et adresses mails de l'équipe Skoll&Yard ayant développé ce jeu.
 - retour à l'écran titre
4. **Quitter** : permet de quitter le jeu.

2.1.3 Règle du jeu

- Le Puissance 4 se joue à deux et la règle du jeu est très simple :
- Nous faisons face à un plateau composé de six lignes et sept colonnes (voir la figure 2)
 - Chaque joueur possède des pions d'une couleur qui le définit
 - Chacun leur tours, ils laisseront glisser leur pions dans l'une des sept colonnes de leur choix
 - Le premier des deux joueurs ayant réussi à aligner quatre de ses jetons horizontalement, verticalement, et en diagonale, à gagner ! (voir la figure 3)

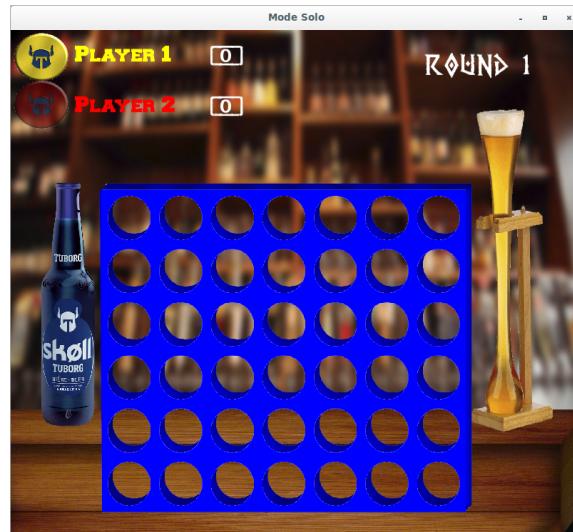


FIGURE 2 – plateau vide de départ

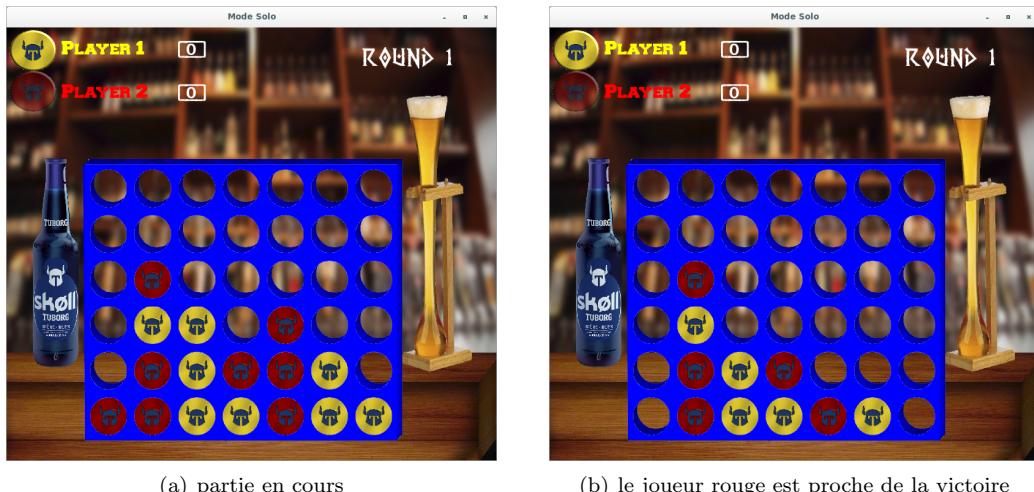


FIGURE 3 – Une partie de Puissance 4

Dans le cas où un joueur sort victorieux de la partie, une fenêtre mentionnant victoire apparaîtra, à l'inverse, le joueur perdant aura une fenêtre mentionnant défaite. (voir la figure 4)



(a) écran Victoire



(b) écran Défaite

FIGURE 4 – Victoire / Défaite

2.1.4 Spécifications

Pour répondre aux spécifications demandées nous avons construit et implémenté le jeu de la façon suivante :

- **Le code source du jeu "pur"** : il se résume en trois grands fichiers
 1. Player.hpp/cpp : un joueur est identifié par un pseudo et un identifiant, il sera donc capable de jouer un coup sur le plateau.
 2. Board.hpp/cpp : le fichier résumant de manière général tout ce qui se rapporte au plateau du jeu. Le plateau est un tableau à deux dimensions de six lignes et sept colonnes. C'est ici que sont rassemblées les méthodes permettant de définir ce qu'est un coup, de jouer un coup, mais aussi de vérifier si une partie est gagnée, ou si il y a match nul.
 3. Game.hpp/cpp : c'est ici que se fait la gestion d'une partie, un jeu (Game) fait le lien entre le plateau (Board) et les joueurs (Player).
- **L'interface graphique** : ici deux fichiers résume le code de l'interface complète
 1. UIGame.hpp/cpp : l'implémentation de UIGame regroupe la gestion de l'interface graphique associé aux mécanismes du jeu comme l'affichage du plateau et des jetons, le placement des jetons selon le coup joué, mais aussi l'affichage des fenêtres victoire, défaite, et égalité selon la condition du joueur.
 2. UIInterface.hpp/cpp : Ici nous sommes plus dans l'interface dite de présentation. Le menu général, les différentes fenêtres de jeu, la fenêtre crédits sont, par exemple, implémentées ici.
- **La partie réseau** :
 - La partie serveur (Serveur.hpp/cpp MainServer.cpp) : Le serveur se compose, dans les grandes lignes, d'une adresse IP, d'un port, et d'une liaison par joueur. Dès que deux joueurs se lient au serveur, ce dernier lance la partie. Le Main rend le serveur autonome et s'exécute en amont par rapport au jeu.
 - La partie client (Client.hpp/cpp) : Le client quand à lui, reprend les mêmes éléments, à savoir l'adresse IP et le port, la liaison avec le serveur. Il se charge de demander une connexion au serveur, et lors d'une partie réseau, il communique aux moyens de messages avec le serveur.
 - **Remarque** : le réseau sera détaillé dans la partie suivante, rubrique "Gestion du réseau"

Enfin, comme convenu dans la demande du client et le cahier des charges, le logiciel fonctionne sur l'environnement Linux (testé et approuvé sur les distributions Ubuntu/Lubuntu/Xubuntu) en langage C++ avec les bibliothèques SFML 2.3. **Attention** : une utilisation du jeu sur une version antérieure de la SFML, la v1.6 par exemple, provoquera le non fonctionnement du jeu tout simplement, tout le projet a été pensé autour de la dernière version en date, à savoir la 2.3.

2.2 Architecture générale

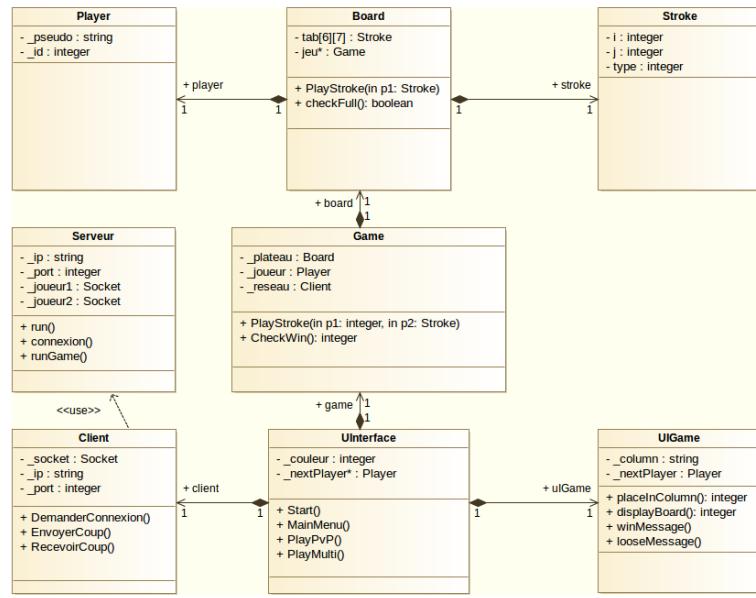


FIGURE 5 – Diagramme de classe du projet

2.3 Gestion du réseau

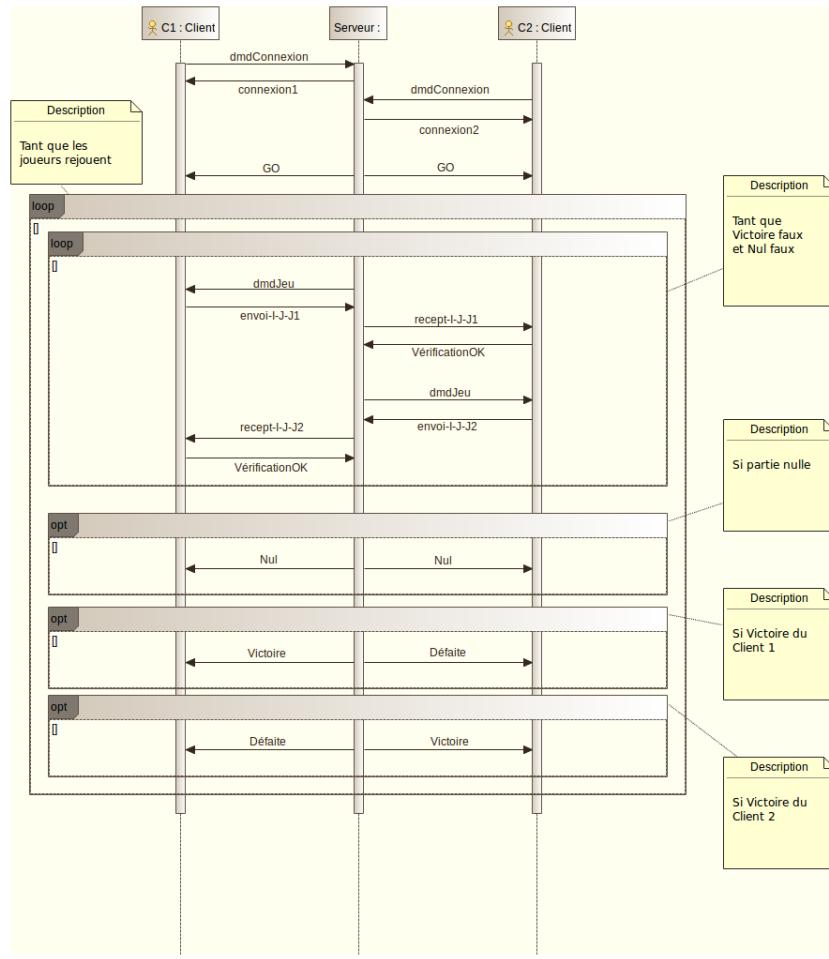


FIGURE 6 – Diagramme de séquence du réseau

2.3.1 Quelques explications ...

dmdConnexion : Message Client vers Serveur, le client fait une demande de connexion au serveur

connexion1/2 : Message Serveur vers Client, le serveur valide la connexion en lui envoyant ce message

GO : Message Serveur vers Client, une fois que deux clients sont connectés, le serveur lance une partie.

dmdJeu : Message Serveur vers Client, le serveur demande au client de jouer

envoi-I-J-J1/J2 : Message Client vers Serveur, le client joue et envoie les données du coup au serveur.

- I est le numéro de ligne
- J est le numéro de colonne
- J1 ou J2 indique au serveur si le client est le joueur 1 ou le joueur 2

recept-I-J-J1/J2 : Message Serveur vers Client, le client réceptionne le coup que le joueur adverse a joué et envoyé au serveur, ce dernier transmet donc le coup.

VérificationOK : Message Client Serveur, une fois que le client a réceptionné le coup de l'adversaire, il tâche de vérifier si le coup est correct, ainsi que de vérifier si la partie est éventuellement gagné, perdue, ou nulle après le placement du coup de l'adversaire. Si l'état de la partie reste inchangé,

il envoie OK au serveur qui demandera au joueur adverse de jouer. Sinon si la vérification décèle un changement d'état, on sort de la boucle de jeu, et on entre dans une des conditions adéquat adaptée à la situation, à savoir Victoire, Défaite, ou math Nul.

Nul : Message Serveur vers Client, dans le cas dans le match nul, le serveur envoie le message nul aux deux clients en même temps.

Victoire / Défaite : Message Serveur vers Client, le serveur envoie le message "Victoire" au client vainqueur, et "Défaite" au client battu.

3 Bilan

3.1 Déroulement du projet

L'équipe est composée de quatre personnes et nous avons la chance d'avoir deux personnes relativement à l'aise concernant la partie graphique, et les deux plus accès sur la conception du code source. Nous avons donc pu partager facilement les tâches en deux binômes distincts pour plus d'efficacité. Un troisième binôme, composé d'un développeur de chaque équipe, sert de pont entre les deux équipes afin de mettre en accord la conception du jeu et l'interface graphique afin que cette dernière soit intégrer par l'équipe graphique.

Skøll&Yard est donc divisé de la façon suivante :

- l'équipe Skøll (interface et intégration graphique) : Maes Benjamin - Prévot Clément
- l'équipe Yard (conception du moteur du jeu et du réseau) : Trabelsi Nadir - Brimeux Benoit
- binôme de transition (mise en relation entre conception du jeu et interface graphique) : Maes Benjamin - Trabelsi Nadir

Concernant l'environnement de développement, nous utilisons deux serveurs Subversion / Redmine (un proposé par la Fac disponible à l'adresse , un créé pour l'occasion par Trabelsi Nadir), ils sont disponibles aux adresses suivantes :

- Subversion de l'université : https://193.49.201.37/svn/projet_skollandyard
- Redmine de l'université : https://193.49.201.37/redmine/projects/projet_skollandyard
- Subversion de l'équipe : <http://draczakkserver.no-ip.biz svn> - nécessite des logins pour y accéder, demande de logins possible à Trabelsi Nadir
- Redmine de l'équipe : <http://draczakkserver.no-ip.biz:3000> - ne fonctionne pas sur eduspot, nécessite de passer par un proxy, sinon fonctionne parfaitement sur un réseau "domicile".

Une documentation complète générée par Doxygen est également disponible dans le dossier doc : il suffit de double-cliquer sur "index.html" afin que la doc s'ouvre dans votre navigateur favori.

3.1.1 Plannification

Selon les prévisions détaillées dans le cahier des charges, la conception du jeu, le choix de la thématique(images et musiques du jeu), ainsi que l'interface graphique (fenêtre, jetons, plateau) ont été implémenté beaucoup plus vite que prévu. Le jeu été jouable en console (avec toutes les vérifications) en trois jours, à la place d'une semaine de prévue, et l'intégration de l'interface par l'équipe graphique était totale au bout d'une semaine à peine.

C'est la partie réseau qui nous a compliqué la tâche, nous faisant perdre l'avance que nous avons accumulé, si bien que pour le prototype, nous n'avons pas eu le temps de lier le réseau, à l'interface graphique. La solution était donc de continuer le travail à la maison, permettant ainsi de valider tout les objectifs demandés en temps et en heure.

3.2 Réalisation des objectifs

fonctionnalités	réalisation
fonctionnalité 1 : jeu de base	complète
fonctionnalité 2 : interface graphique	complète
fonctionnalité 3 : jeu en réseau	complète

Pour la fonctionnalité 2 : Interface Graphique, il y a un souci si on redimensionne les fenêtres de jeu, les coups joués après redimensionnement sont décalés. Pour la fonctionnalité 3 : Réseau, si le joueur clic plein de fois à la suite, les clics sont enregistrés et sont joués automatiquement lorsque le joueur à de nouveau la main. Pour jouer proprement il faut donc tâcher de cliquer qu'une seule fois.

3.3 Conclusion pour les projets futurs

Notre avancée concluante sur le développement de ce projet confirme notre choix de la thématique du sujet, à savoir développer un jeu en réseau. Nous avons pris plaisir à chercher une thématique originale, changeant de l'ordinaire. Cela nous a permis d'apprendre de nouvelles technologies, tel que la librairie SFML. Nous avons donc pris conscience de la capacité à développer un projet en équipe destiné à un client, nous permettant d'avoir un bon aperçu de la vie active d'une équipe de développeurs.