
# Rapport de projet

Fabien SAVY Jérémie SOETENS Killian TROLÈS

4 mai 2019

**Résumé**

*Le Puissance 4 est un jeu dont les règles sont connues de tous. Le principe est simple, mais stratégie et réflexion sont nécessaires pour vaincre l'adversaire. Pour coder ce jeu, nous avons utilisé PodSixNet, une bibliothèque réseau qui permet de créer des jeux multijoueurs en Python. Celle-ci gère les interactions entre un serveur et des clients et permet d'envoyer et recevoir toutes les données utiles au bon fonctionnement du jeu.*

## Le serveur, maître du tournoi

Nous avons choisi de donner la plupart des responsabilités au serveur. C'est lui qui reçoit l'inscription de l'ensemble des joueurs et fait le lien entre les différents clients. Il s'assure de la connexion des différents participants au tournoi et affecte un statut à chacun : CONNECTED (connecté, en attente d'une partie), PLAYING (en partie) ou DISCONNECTED (déconnecté). Il stocke les pseudonymes des participants, leur score et leur état. Un bouton permet de lancer le tournoi : seuls le joueurs qui étaient connectés avant le début du tournoi peuvent alors se connecter au serveur en cas de besoin. Ensuite, nous avons choisi d'intégrer le moteur de jeu dans le serveur et non dans chaque client. Cela permet de centraliser les informations sur les parties en cours et d'éviter aux clients de prendre des décisions contradictoires et d'occasionner des conflits. Lorsqu'une partie est lancée, une instance dans la classe GameEngine est créée : elle modélise un plateau de jeu.

## Le moteur de jeu GameEngine, abstraction du jeu

Nous avons modélisé la grille du Puissance 4 par un array numpy (un tableau à 2 dimensions) contenant des chaînes de caractères : une chaîne vide s'il n'y a pas de jeton ou le pseudo du joueur s'il y en a un. Pour vérifier si un des joueurs a gagné, on représente

la situation en une grille de “0” pour les cases vides ou contenant les pions du joueur adverse, et de “1” pour les jetons du joueur actuel. Ensuite on représente la grille comme une liste de nombres binaires (dans le sens horizontal et vertical) puis par une comparaison bit à bit avec l'opérateur “&”, on détecte si quatre “1” sont présents à la suite horizontalement ou verticalement (et diagonalement de la même manière mais en “shiftant” les bits avec l'opération “<<” de la liste au préalable). Si c'est le cas, cela signifie que 4 jetons sont à la suite et que le joueur a gagné. Quand c'est à son tour, le client envoie la colonne dans laquelle il veut placer son pion. Le serveur fait le traitement (si c'est possible, si il a gagné etc...), puis envoie l'information de la position du jeton aux deux clients qui l'affichent à l'écran.

## L'affichage graphique Gui d'une partie en cours

Le rôle de Gui est de gérer l'interface graphique d'une partie du point de vue d'un client. Sur un canevas bleu sont affichés 42 emplacements (7 colonnes par 6 lignes), blancs par défaut. Lorsqu'un jeton est joué, l'emplacement correspondant devient jaune ou rouge, en fonction de la couleur du joueur. Le nom des deux joueurs est affiché en haut de la fenêtre, celui dont c'est le tour est sur un fond de sa couleur, l'autre sur un fond blanc. Pour jouer un pion, il suffit de cliquer sur l'un des emplacements de la colonne correspondante. Si le coup est possible, on voit le pion tomber le plus bas possible, avec une animation de chute. Si la colonne est pleine, ou que ce n'est pas à notre tour de jouer, l'emplacement cliqué affiche une animation de grossissement pour faire comprendre que le coup n'est pas possible.

# Schémas des interactions entre le serveur et les clients