INSA Lyon

Programmation Reseau

4BIM

Targui

Auteurs :
Emma Prudent
Maxime Sainlot
Johan Chan

Enseignant :
Christophe Pera

12 février 2014

1 Introduction

Nous avons choisi pour notre projet, de développer Targui : un jeu de société pour deux joueurs, en y intégrant une composante réseau.

Dans le jeu de société Targui, vous incarnez le chef d'une des tribus de ces "hommes en bleu" que sont les Targuis appelés également Touareg.
L'objectif est simple : sillonner le désert, les oasis et les camps afin de mettre la main sur différentes marchandises qui permettront à votre tribu de prospérer et à vos richesses de s'accroître. Votre rôle est donc d'avoir la plus grande tribu.

2 Les règles du jeu : en bref

Déroulement d'une manche

Phase 1 Chaque joueur place successivement ses 3 "targuis" (pions en forme de personne) sur les cartes "contours" (les cartes fixes exterieures sur le plateau central) à l'exception des coins, selon des règles de placement spécifiques.

Phase 2 Les joueurs placent leurs "marqueurs" à l'intersection des lignes et des colonnes où ils ont leurs targuis

Phase 3 Les joueurs retirent tous leurs pions (targui+marqueurs) en appliquant les effets des cartes présentes sous leurs pions.

Fin de manche Le voleur est un pion qui tout autour du jeu progresse sur les cartes contour. A chaque fois que celui-ci atteint un coin du plateau (4 fois par partie) il dérobe aux joueurs des ressources ou des pv. En fin de manche, le voleur avance d'une case dans le sens des aiguilles d'une montre.

Les différentes cartes et leurs fonctions

Les cartes marchandises rapportent des ressources (poivre, sel, dattes, or, pv)

Les cartes tribus nécessitent des ressources pour être achetées et sont la source principale de points de victoire. Ces cartes appartiennent chacune à une catégorie (puit, camp, oasis, targuia, chameau) et sont placées dans le "tableau de bord" 3x4 lignes. La façon dont ces cartes sont agencées en fin de partie (suivant les catégories) est une ultime source de pv.

Les cartes spéciales , présentent sur les contours du plateau uniquement, sont au nombre de 6 : Noble, Marchand, Fata Morgana, Orfèvre, Caravane et Expansion Tribale. Elle ont chacunes une fonction spécifique permettant entre autre de piocher des cartes (marchandise ou tribu), de modifier l'emplacement d'un de ces marqueurs ou encore de faire des échanges entre ressources, or et pv. Pour plus de détails voir les règles officiels en pièce-jointe.

Fin de Partie

La partie prend fin lorsque le voleur a atteint la dernière case contour ou lorsqu'un des deux joueurs a placé 12 cartes tribus dans sont tableau de bord.

3 Structure du code

Le code est découpé en 4 parties afin de le rendre plus clair et fonctionnel :

Le jeu

Le fichier `targui_jeu.py` est le code qui gère le fonctionnement du jeu. On y retrouve les règles du jeu implémentées en langage informatique ainsi que les attributs et fonctions nécessaires au déroulement d'une partie.

L'interface graphique

Le fichier `targui_gui.py` est résponsable de tout l'aspect graphique du jeu. On y retrouve plusieurs classes et fonctions permettant la création de la fenètre principale, l'agencencement des widgets, les gestions d'interactions avec l'utilisateur (survol et clic souris) ainsi que les fonctions de mise à jour de l'affichage.
Un second fichier `tab_connexion.py` est importé par celui-ci et permet l'affichage d'une première fenètre de connexion permettant à l'utilisateur de choisir entre créer ou joindre une table avant de réelement commencer se lancer dans le jeu. Cette interface est le support de la possibilité pour plusieurs tables de 2 joueurs d'être créées en simultanées. On pourra également y suivre la progression des parties déjà lancées.

Le réseau

Le client `client_targui.py` est en étroite relation avec l'interface graphique. Il instancit les fenêtres et les met systématiquement à jour à l'aide des informations qu'il reçoit du serveur. A chaque utilisateur (joueur) correspond un client.

Le serveur `serveur_targui.py` est en étroite relation avec le code `targui_jeu.py`. C'est en effet dans celui-ci que l'on retrouvera les instances de jeux ainsi que toutes les informations nécessaires au bon déroulement d'une partie qu'il communiquera avec les clients. Son rôle est de centraliser les connexions de tous les clients. On a donc un gros serveur auquel se connecte plusieurs clients.

4 Les choix technologiques

Le langage informatique utilisé est le Python, d'abord parce que c'est celui que nous maitrisons le mieux et ensuite parce qu'il permet de réaliser des architectures client-serveur relativement simplement et de manière efficace.

Le projet s'intéressant à un jeu de société, l'utilisation d'une interface graphique s'est révelée judicieuse. Ces interfaces ont été réalisées en PyQt pour plusieurs raisons : la puissance, la documentation, et le système de signaux et slots spécifique à Qt.

La nature du jeu demandait à ce que les échanges soient fiables et que tous les paquets arrivent entiers, dans l'ordre où ils avaient été envoyés. L'utilisation du protocole TCP, dans l'ensemble du projet, se justifie ainsi.

Les Threads

L'utilisation de Threads s'est avérée indispensable pour réaliser ce projet. Ces threads, présentent dans le client et dans le serveur, permettent que des processus aient lieux en parallèle. Typiquement :

Coté serveur Des threads, héritées du module threading de Python, sont utilisées pour séparer les connexions de nouveaux clients, les parties en cours et le tchat. Nous avons fait le choix d'ouvrir 2 sockets différentes entre chaque client et le serveurs afin de rendre indépendant les échanges en rapport avec le jeu et les échanges de messages (tchat) entre les deux joueurs.

Coté client L'utilisation de QThread issus du module de PyQt permettent de séparer l'affichage graphique principal, les attentes d'actions du client (clic) durant la partie, l'émission de message par le tchat et la réception des messages des autres joueurs. L'utilisation de QThread (au lieu de Thread) dans la partie client s'est justifiée par le fait que la *mainloop* de l'affichage graphique devait se trouver dans la thread principale alors que les threads secondaires (déroulement de la partie et réception des messages du tchat) devaient avoir la possibilité de communiquer avec celle-ci pour actualiser l'affichage. L'utilisation de signaux et slots, seulement disponibles dans des environnements basés sur PyQt, nous a permis d'établir cette connexion entre threads tout en conservant le parallélisme d'éxecution.

5 Les intéractions réseaux

Le fonctionnement global du programme

On commence par lancer le serveur. Idéalement, ce serveur tourne jour et nuit sur une machine distante en attente de nouveaux clients. Ce n'est pas notre cas dans ce projet. Pour jouer à Targui, il faudra donc commencer par lancer le serveur. On entend alors par serveur, le programme global qui centralisera toutes les parties en cours et connexions entrantes.

Lancement du serveur Le serveur commence par ouvrir 2 sockets, sur deux ports différents, comme on l'a vu précédemment. L'une va servir à échanger les informations relatives au jeu, aux parties et à la création de nouvelles tables, l'autre sera utilisée pour le tchat. Le serveur lance ensuite l'exécution d'une thread `ThreadServeur`, en écoute perprétuelle de nouvelle connexion clients.

Lancement d'un client Lorsque l'utilisateur souhaite jouer à Targui, il lance le client avec le pseudo de son choix. Immédiatement, celui-ci va ouvrir 2 sockets (en miroir du serveur avec les mêmes numéros de ports) ayant les mêmes vocations que celles du serveur. Puis il s'occupera de lancer l'éxecution de la thread `ThreadJeu` qui va gérer la connexion (c'est-à-dire l'affiliation du client à une table) puis le déroulement de la partie une fois celle-ci commencée.

Connexion de nouveaux clients Lorsque ceci se produit, la `ThreadServeur` lance à son tour l'éxecution d'une nouvelle thread `ThreadClient` qui aiguillera le client vers le début d'une partie. Du point de vu du client, celui-ci a le choix entre créer une nouvelle table et attendre l'arrivée d'un second joueur ou rejoindre une table créée par un autre joueur et commencer une partie. Ces choix se font à l'aide d'une fenètre de connexion graphique où le joueur peut consulter les tables en attente de joueurs et les parties en cours avec leurs caractéristiques.

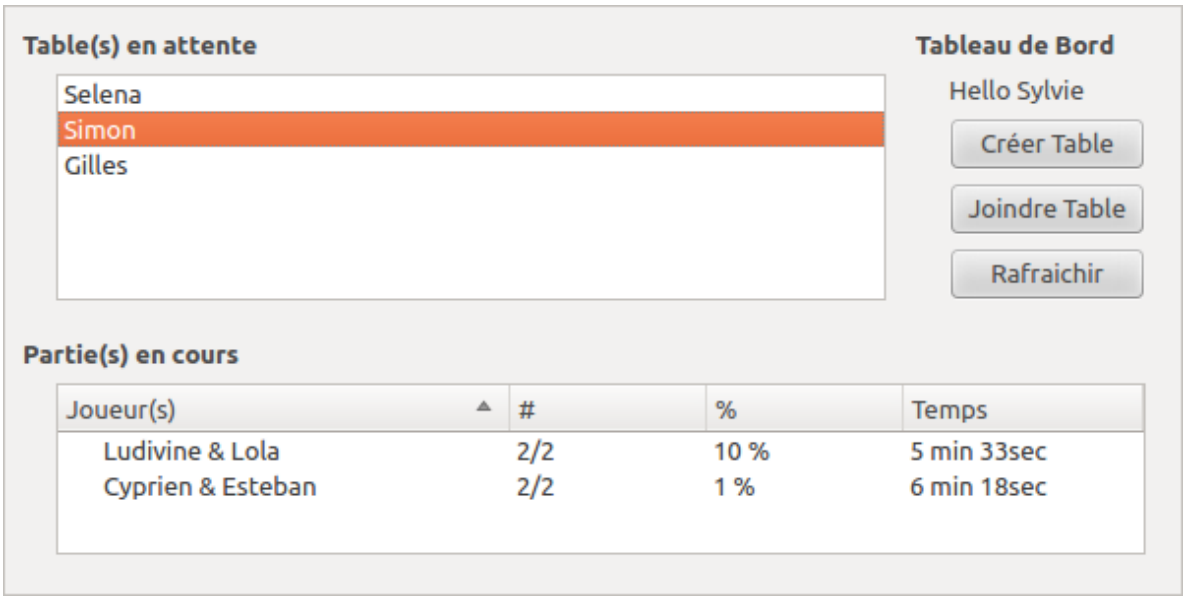

Figure 1 – Interface de connexion à Targui

Lancement du jeu De retour du côté serveur, une fois le client connecté à une table avec un autre joueur, la thread `ThreadClient` en cours va lancer l'éxecution d'une ultime thread `ThreadPartie` qui va commencer le jeu à proprement parlé. Cette découpe apporte la possibilité pour le serveur de supporter plusieurs tables en même temps. Côté client, la thread continue son exécution vers le début de la partie en ouvrant une seconde fenêtre (et en fermant la première) dévoilant ainsi le plateau de jeu.

La partie se déroule en plusieurs manches de 3 phases (détaillées précédemment dans les règles) jusqu'à ce qu'une des conditions de fin de partie soit remplie. La communication client serveur prend alors la forme d'un dialogue suivant plusieurs conditions. En fin de partie, il a été important de bien stopper toutes les threads et de fermer toutes les sockets impliquées dans une partie. Seul le serveur restera en écoute.

6 Les difficultés rencontrées

Nous avons rencontrer beaucoup de difficultés tout au long de ce projet. On a commencé par se questionner sur la meilleure façon d'organiser les choses, quels outils, quels langages, comment est-ce que que les clients échangeront avec le serveur sachant qu'ils sont limités à des chaines de caractères (blocage pour transmettre des instances de classes)...Nous avons ainsi pu opter pour la répartition présentée sur le schéma suivant :

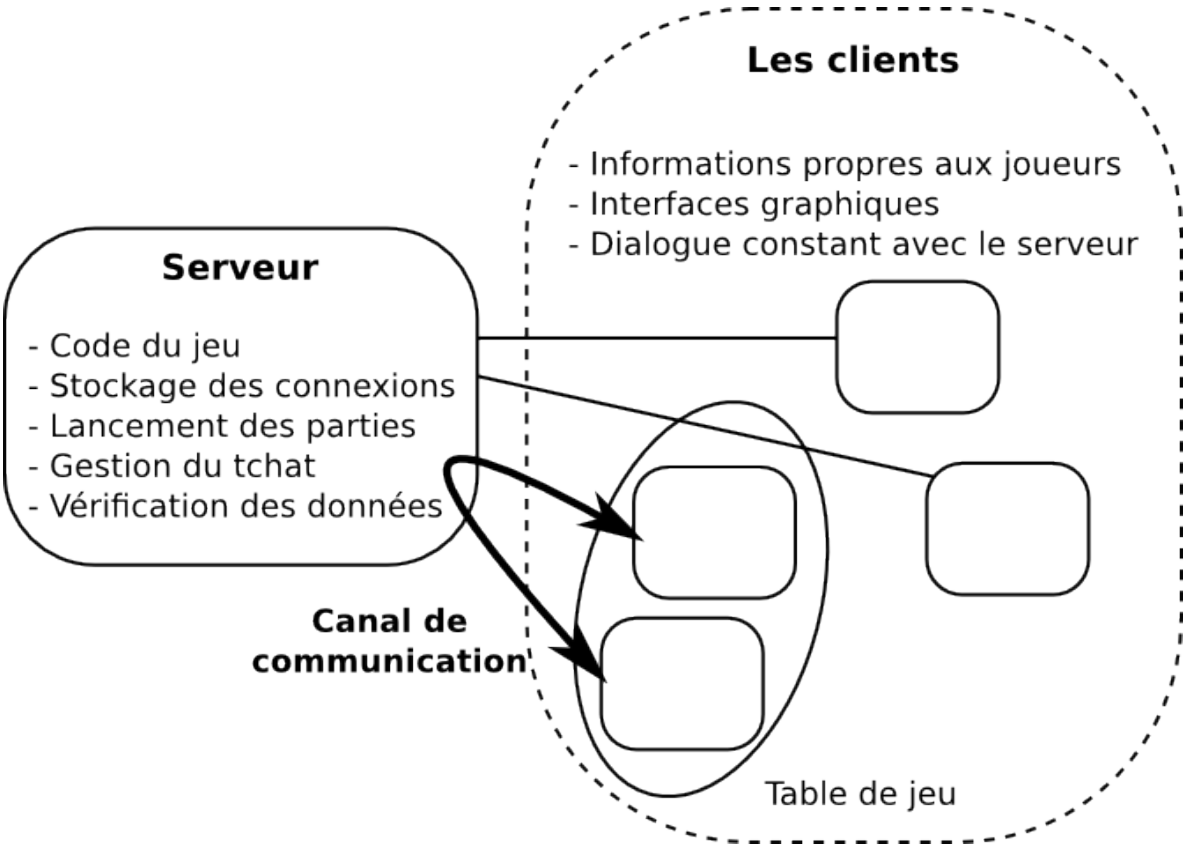

Figure 2 – Organisation réseau

7 Les améliorations futures

Ils nous reste encore de nombreuses améliorations possibles. Au niveau du jeu nous pourrions pousser l'implémentation en prenant en compte les pouvoirs spéciaux des cartes, les cartes multi-ressources...Nous pourrions également travailler sur l'optimisation de notre code afin de rendre son exécution plus fluide. Nous nous sommes en effet rendu compte qu'il monopolisait une grande partie, pour ne pas dire la totalité, du processeur lors de son exécution (de nombreuse boucles infinie en attente d'évènements). A côté de ça nous pourrions encore améliorer le fonctionnement avec une meilleure gestion des pseudo pour éviter les répétitions, la possibilité d'annuler un de ces choix ou encore la possibilité de retrouver la table que l'on a quitté suite à des problèmes techniques. Enfin nous avons rencontrés des difficultés pour appréhender PyQt, nous avons notamment un warning qui persiste lorsque les widget de l'écran sont modifiés alors qu'ils sont en train d'être survolés (visualisation zoom), que nous aurions aimé corriger.

Nous nous sommes néanmoins battus jusqu'au bout, malgré l'importance du travail à accomplir et c'est avec plaisir que nous vous rendons cette version, que nous avons pu aboutir un peu plus grâce au délais accordé.

Il ne nous reste plus qu'à vous souhaiter bon jeu !

Annexe

Légende de l'interface du jeu

❶ Barre de progression de la partie

❷ Panneau d'instructions et d'informations s'adressant au joueur

❸ Visualisation instantanée par survol souris. Notamment utile pour voir la carte complète des tribus déjà achetées et placées dans son tableau de bord

❹ Tableau de bord du joueur en cours, on y retrouve les tribus déjà achetées. Il est possible de les consulté par survol de la souris.

❺ État des ressources actuelles dont dispose le joueur en cours

❻ Permet l'affichage d'une éventuelle carte en main (non-géré pour le moment)

❼ Tchat permettant de dialoguer avec son adversaire

❽ Plateau central de jeu

47%

1

Retirez vos pions.

2

3

4

5

6

Tribalcards

Joseph

4 3 7

0 9

Ninon

4 2 5

0 10

Tribalcards

Joseph : Bonjour Ninon !
Ninon : Salut Jo'

7

8

ROBBERY

NOBLE

ROBBERY

MERCHANT
(several times)

PUT 1 GOLD FROM THE SUPPLY ON THIS CARD. IF YOU LATER PLAY A CARD THAT COSTS GOLD, TAKE THIS GOLD.

TRIBAL EXPANSION

AT THE END OF THE GAME YOU SCORE 1 ADDITIONAL VICTORY POINT FOR EVERY 2 CARDS „OASIS" IN YOUR DISPLAY.

CARAVAN

WHEN YOU USE THE ACTION 'MERCHANT', YOU ARE ALLOWED TO EXCHANGE GOODS IN RATIO OF 1 TO 1.

FOR EACH 2 FREE PLACES IN YOUR DISPLAY YOU GET 1 GOOD OF YOUR CHOICE ONCE.

ROBBERY

SILVERSMITH

FATA MORGANA

ROBBERY