

Projet de Programmation réseau

Pong évolué

L'objectif du projet est de réaliser en binômes un jeu de Pong en réseau.

On rappelle qu'il s'agit d'un projet de programmation *réseau*, c'est donc cet aspect qui sera évalué avant tout.

1 Sujet

1.1 Quelques liens utiles :

- Sockets : <https://docs.python.org/library/socket.html>
- Select : <https://docs.python.org/library/select.html>
- Pygame : <http://www.pygame.org/docs/>

1.2 Description générale

L'objectif de ce projet est de réaliser un jeu de pong en réseau. Pong, créé en 1967, est un des premiers jeux vidéos. Une petite balle se déplace à travers l'écran, rebondissant sur les rebords du haut et du bas, et les deux joueurs commandent chacun une raquette, la faisant glisser verticalement entre les extrémités de l'écran à l'aide des contrôles. Si la balle frappe la raquette, elle rebondit vers l'autre joueur. Si elle manque la raquette, l'autre joueur marque un point. La balle rebondit de différentes manières selon la façon dont elle touche la raquette.

Là où l'original se jouait sur une seule machine, vous devrez implémenter le jeu Pong afin que deux joueurs puissent jouer depuis des machines différentes.

1.3 Information relative au code source

Le projet est disponible à <http://dept-info.labri.fr/~thibault/Reseau/pong.tgz> , il contient simplement les images pour le jeu et un bout de code python utilisant pygame pour implémenter une version basique du jeu.

La structure du code est très simple : on a deux objects *ball* et *racket* qui se baladent, il n'y a pas de notion de réseau.

La boucle principale teste d'abord la pression des touches, puis calcule le déplacement des objets, détecte si le joueur a perdu, affiche le tout, et attend 10ms avant de recommencer.

2 Passage en réseau

Il s'agit maintenant de jouer avec une autre machine.

Dans un premier temps, faites une version simple avec un support pour seulement 2 machines à la fois. Sur une machine, on lance le programme sans argument, et le programme doit alors ouvrir une socket TCP en écoute. Sur l'autre machine, on lance le programme avec le nom de la première machine en paramètre, le programme doit, dans ce cas, se connecter en TCP à la machine passée en paramètre.

Maintenant que l'on a une socket établie entre les deux programmes, il s'agit de transmettre les informations sur les objets. Il faut donc établir un protocole pour les exprimer. Il est recommandé d'établir un protocole textuel ascii, qu'il sera ainsi facile de déboguer à la main. Réfléchissez et implémentez dans un premier temps votre propre protocole.

Le principe du protocole est que chaque machine reste maître du mouvement des objets qu'elle gère. Ces objets sont la raquette du joueur et un ensemble d'objets mobiles (dont la balle), qui se déplacent dans une portion d'écran de jeu. Ainsi, on peut découper l'espace en deux et quand la balle passe d'un côté à l'autre, la machine responsable de la balle change. Les autres machines se contentent d'afficher ces objets aux nouvelles positions, lorsqu'elles les reçoivent via la socket. Il sera facile de vérifier que les affichages sont bien exactement les mêmes (au délai de latence près).

3 Extensions

Une fois cette version de base développée, de nombreuses extensions sont possibles et intéressantes à développer. Il n'est pas obligatoire de toutes les développer pour avoir une bonne note, mais cela y contribue bien sûr fortement, il faut en développer au minimum quelques-unes. La liste n'est également pas exhaustive. Si vous avez des idées, discutez-en avec votre chargé de TD.

3.1 Éviter la triche

Pour être sûr qu'il n'y a pas de triche, un moyen simple est de synchroniser les machines : pour chaque pas de temps (10ms), chaque machine annonce à l'autre quels mouvements a lieu. Chacune peut donc vérifier que les mouvements de raquette de l'autre sont bien limités à la vitesse maximale. La boucle principale devient donc :

- tester les touches pour changer les vitesses
- calculer les nouvelles positions des objets pour lesquels on est responsable,
- envoyer ces nouvelles positions à l'autre machine,
- recevoir les nouvelles positions des objets de l'autre machine,
- vérifier qu'elles sont correctes, i.e. pas trop loin des positions précédentes,
- afficher le résultat,
- attendre 10ms.

3.2 Les balles spéciales et bonus

Pour que le jeu devienne plus intéressant, on peut ajouter des bonus. Ces bonus sont des objets supplémentaires qui "tombent" lentement vers les raquettes. Si la raquette parvient à toucher un de ces objets (ce qui n'est pas évident si la balle est en train de retomber de l'autre côté de l'écran!), le bonus est gagné. De très divers bonus sont possibles. On peut éventuellement avoir des bonus "surprise" dont on ne sait pas à l'avance ce qu'ils sont.

- ralentisseur et accélérateur
- raquette plus large
- raquette aimantée
- plusieurs balles
- père Noël
- ...

La probabilité d'apparition des bonus pourrait être lié au nombre de points gagnés par exemple.

3.3 Pas de triche avec les bonus ?

Le problème des bonus, c'est qu'une machine pourrait très bien décider de positionner des tas de bonus juste au-dessus de sa raquette, pour elle tout seule...

Il vaut donc mieux que les bonus soient gérés par une machine tierce, qui sert d'arbitre en quelque sorte.

3.4 Plus de 2 joueurs

L'écran a 4 côtés, on peut donc jouer jusqu'à 4 joueurs !

L'idée est que chaque machine a des connexions vers tous les autres joueurs, donc toutes les machines font à la fois client et serveur en fait, c'est juste l'ordre de lancement qui change dans quel sens les connexions sont établies, mais ensuite elles sont symétriques.

Lorsqu'un client se connecte à un autre client, il en récupère une liste de l'ensemble des machines actuellement inter-connectées, et donc peut se connecter à elles.

Si un client quitte, il disparaît simplement du jeu.

4 Organisation

Le projet est travaillé et étudié en binôme mais la notation est individuelle. Le projet *doit* être réalisé avec un outil de gestion de révision (svn, git, ...) : directement dans votre *home* ou sur la savanne du CREMI (<https://services.emi.u-bordeaux1.fr/projet/savane/>). Le chargé de TD évaluera la contribution de chacun des éléments du binôme au travail commun. Le chargé de TD se réservera la possibilité de modifier la composition de chacun des binômes. Afin de permettre un travail profitable, il est conseillé de ne pas créer de groupes avec des niveaux trop différents.

5 Rapport

Il s'agit de mettre en valeur la qualité de votre travail à l'aide d'un rapport. Pour cela le rapport doit explicitement faire le point sur les fonctionnalités du logiciel (lister les objectifs atteints, lister ce qui ne fonctionne pas et expliquer - autant que possible - pourquoi).

Ensuite le rapport doit mettre en valeur le travail réalisé sans paraphraser le code, bien au contraire : il s'agit de rendre explicite ce que ne montre pas le code, de démontrer que le code produit a fait l'objet d'un travail réfléchi et même parfois minutieux. Par exemple, on pourra évoquer comment vous avez su résoudre un bug, comment vous avez su éviter/éliminer des redondances dans votre code, comment vous avez su contourner une difficulté technique ou encore expliquer pourquoi vous avez choisi un algorithme plutôt qu'un autre, pourquoi certaines pistes examinées voire réalisées ont été abandonnées.

Il s'agira aussi de bien préciser l'origine de tout texte¹ ou toute portion de code empruntée (sur internet, par exemple) ou réalisée en collaboration avec tout autre binôme. Il est évident que tout manque de sincérité sera lourdement sanctionné.

1. Directement dans le texte, par une note en bas de page comme celle-ci ou par une référence bibliographique entre crochet [1].