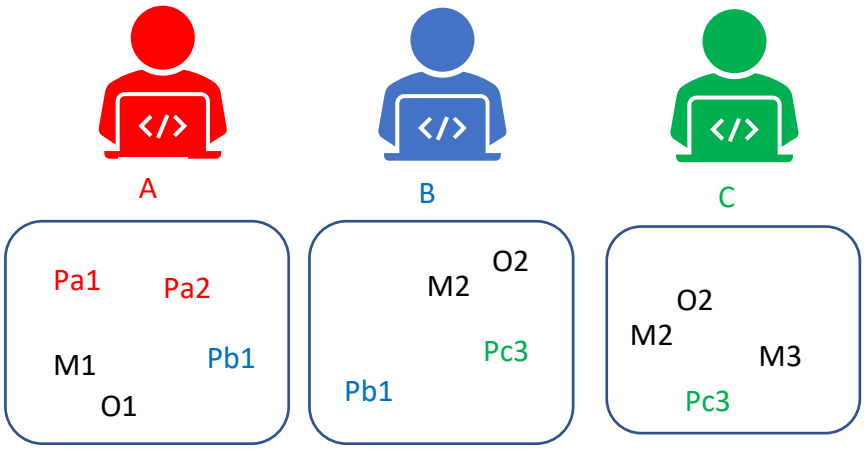
Jeu réparti cohérent Age of Cheap Empires

Christian Toinard

1. Introduction

L'objectif est d'apporter un aspect multi-joueurs au projet de programmation précédant en permettant à différents joueurs d'entrer en compétition sur la base d'un système réparti à large échelle.
Pour cela vous allez travailler en groupe et choisissez d'étendre l'un des projets Python réalisés précédemment.
Il s'agit de répartir entièrement une partie de jeu entre les différents participants en **garantissant une cohérence** sans introduire le moindre serveur ni de façon permanente ni de façon transitoire. Chaque joueur a donc une copie locale du jeu qui lui permet de visualiser la scène de jeu, d'interagir avec des objets, de combattre avec les autres joueurs.
Un joueur interagit localement avec son logiciel de jeu via son clavier et sa souris. Les actions d'un joueur produisent des changements d'état de la partie qui sont transmis aux copies distantes sans passer par un serveur. Afin de garantir une jouabilité, il faut que les changements d'état soient observés de façon cohérente en transmettant aux copies distantes les changements effectués localement.



2. Principe de répartition

L'objectif est de réaliser un vrai jeu en réseau sur une base entièrement répartie. La solution doit offrir un niveau de jouabilité importante en garantissant que chaque participant respecte les règles du jeu communes tout en disposant d'une jouabilité et d'une fluidité satisfaisantes.

Pour garantir la cohérence, chaque **élément** (personnage, monstre, objet, case du jeu) ou partie de cet élément (exemple : épée contenue dans un coffre) dispose d'un **attribut de propriété** qui est cessible. Lorsqu'un joueur n'en dispose pas, il doit **demander la propriété**. La propriété est transmise par son propriétaire en même temps que l'état de l'élément ou partie d'élément. Ce mécanisme de **transmission de la propriété et de l'état** doit permettre à la fois une **concurrence** et une **cohérence** pour l'élément ou une partie de celui-ci. Pour cela, le propriétaire doit être unique à un instant donné. Le propriétaire dispose donc de l'état cohérent de cet élément ou de cette partie. Il peut donc modifier l'élément ou en céder la propriété avec

un état cohérent à un joueur distant. Cependant, vous allez devoir réfléchir aux contraintes en termes d'interface utilisateur. En effet si un joueur interagit avec un élément dont il n'est pas propriétaire, il devra attendre l'état de l'élément avant de décider ou non de poursuivre son interaction. Ainsi, un personnage qui veut se déplacer sur une case occupée verra l'occupation au moment de la réception de l'état et ne devra pas pouvoir effectuer ce déplacement (c'est l'entité de jeu locale qui refuse l'opération et pas le joueur).

3. Travail du point de vue réseau

Vous faites un travail de définition de **protocole**, notamment **pour la cohérence**. L'ensemble de ces protocoles supportent vos fonctionnalités. Vous les concevez et justifiez et en donnant les limites.
Pour chaque protocole, vous expliquez la mise en œuvre. Donc vous devez justifier que votre code réalise de façon satisfaisante ce protocole. En effet, on peut tout à fait proposer un bon protocole mais dont la réalisation est défaillante. Par exemple, vous ne calculez pas les limites des messages lorsque vous transmettez ces messages via TCP donc votre réalisation ne fonctionne pas c'est-à-dire marche de façon tout à fait erratique.

4. Travail du point de vue système

Vous allez devoir définir l'architecture système, c'est-à-dire les processus et activités ainsi que les moyens de communication et de synchronisation.
Vous devez justifier votre architecture système vis-à-vis du traitement de la concurrence. En effet, votre architecture système peut être incapable de garantir la cohérence des données et des opérations sur ces données.
Il vous est aussi demandé de justifier l'efficacité de votre architecture. Ainsi, une architecture utilisant des activités peut être rendue inefficace en raison des mécanismes de synchronisation choisis (par exemple, chaque activité s'exécute en exclusion mutuelle ce qui interdit tout parallélisme ou gain de performance).
Vous justifiez que la mise en œuvre est correcte. Vous montrez notamment que vous traitez bien les cas d'erreurs et les exceptions système. En autre terme, vous devez justifier le comportement de votre code et ses limites.

5. Organisation de l'équipe

Un étudiant sera choisi comme coordinateur pour l'équipe afin de rendre compte du travail au sein de l'équipe, mettre en place la coopération entre les membres de l'équipe. Le coordinateur sera chargé des échanges avec l'enseignant qui encadre le projet.

Vous êtes entièrement libre de vous organiser au sein de l'équipe. La présence aux séances de suivi est obligatoire pour toute l'équipe et doit contribuer à la coordination et aux échanges.

Une auto-évaluation vous est demandée. Vous devez le plus possible prendre en charge un problème ou une fonctionnalité du début à la fin (c'est-à-dire de sa définition, conception de la solution, mise en œuvre, test et intégration, justification). Si le travail à plusieurs, sur un même problème ou fonctionnalité, est toléré vous devez par ailleurs avoir proposé une contribution individuelle. Vous devez décrire et justifier vos contributions. Vous vous donnez une note ou une position dans le groupe et vous les justifiez. La présence sur le GIT de vos contributions fait foi.

Une évaluation par les pairs est demandée. Pour cela les différents membres devront donner un avis sur le coordinateur. Celui-ci attribuera une note ou une position à chacun des membres. Il justifiera les notes données au regard des contributions de chacun des membres.
Une évaluation par les pairs est demandée. Pour cela les différents membres devront donner un avis sur le coordinateur. Celui-ci attribuera une note à chacun des membres. Il justifiera les notes données au regard des contributions de chacun des membres.

6. Contraintes de développement

Vous devez développer l'approche de répartition et la mise en réseau en C avec les mécanismes système adaptés (API socket, processus, thread, IPC, mécanismes de synchronisation, …).

De préférence, nous vous conseillons de produire la partie réseau dans un ou plusieurs processus indépendants de la partie Python. Il vous faudra donc utiliser des mécanismes de communication et de synchronisation entre les processus Python et C. Cela vous demande donc de faire de la programmation système à la fois en Python et en C. Par contre, la partie réseau doit être développée entièrement en C au moyen de l'interface socket.

Nous vous laissons libre de développer sous Unix ou sous Windows. Le fait de fournir une solution sous Windows vous donnera une plus-value dans la mesure où c'est un OS sur lequel vous n'avez pas de cours de programmation système ni réseau. Un OS Windows actuel est très proche d'un Unix et la programmation en C sur ces deux plateformes présente de nombreuses similarités. Cependant, il existe des spécificités à Windows que vous devrez étudier afin de maîtriser la programmation système sur cet OS. C'est pourquoi nous évaluerons positivement votre capacité à proposer une solution sous Windows.
Bien entendu vous pouvez proposer une solution multi-plateformes supportant à la fois Unix/Linux et Windows.

7. Travail attendu

Vous devez proposer au plus vite une première version du jeu que vous présentez durant les séances de suivi. Ainsi, plus vous fournissez rapidement une première version qui permet à deux joueurs d'interagir concurremment sur un objet partagé en garantissant sa cohérence.

La première version doit supporter une partie avec deux joueurs. Sur la base de cette première version vous proposerez différentes améliorations pour les fonctionnalités, l'interface, le réseau, l'architecture système voire le passage à l'échelle et la sécurité. Pour chacune de ces améliorations vous présenterez les limites.

La livraison finale consiste en 4 diapositives qui résument la solution, les avantages, les inconvénients ainsi que les évaluations et mesures faites. La soutenance repose essentiellement en une démonstration qui illustre le fonctionnement, les avantages et inconvénients. Votre travail sera évalué sur la base 1) de votre capacité à présenter et justifier votre solution durant les séances, 2) du rapport d'auto-évaluation/évaluation des pairs de vos dépôts sur le GIT et 3) de la soutenance finale.

Une attention particulière sera portée à votre autonomie, votre capacité à communiquer et justifier vos contributions.