Very Massive Multiplayer Online Game

Marion Dalle & Rémi Piotaix
Tuteur : Jacques Léger

POLYTECH
GRENOBLE

Présentation du projet

- Jeu de course
- Partie rapide (≈ 10 minutes)
- Joueurs > 10 000 000
- Objectifs
  - Classement approximatif à tout moment
  - Un seul vainqueur
  - Vision du voisinage

Problématique et idées de résolution

Performances

- Trop de clients pour un seul serveur
  - Nécessité de déléguer
- Connaissance locale uniquement
  - Découpage spatial de la carte

Architecture cellulaire

- Cellules
  - Gérées par un leader
  - Connaissance des joueurs de notre cellule uniquement

- Leaders
  - Centralisent les connexions client
  - Communiquent entre eux
  - Notifient les changements de position

Architecture cellulaire

Problèmes

- Système distribué
  - Pas d'état global
    - Concensus pour le classement
- Problème à l'extérieur des cellules
  - Chevauchement nécessaire

Réactivité

Contraintes de "temps réel"

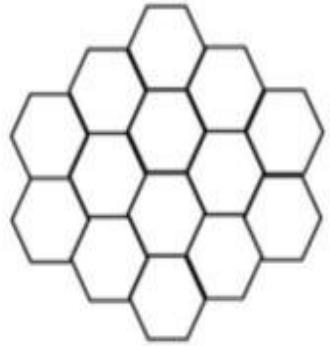
- Fin de partie
  - Gagnant
  - Temps écoulé

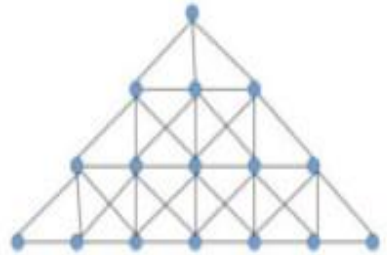
- Affichage des concurrents
  - Volume de données variable
    - Nombre de joueurs ?
    - Fréquence de rafraîchissement?

Modélisation et simulation

Modélisation

- Chaine de Markov
  - L'état suivant ne dépend que de l'état présent
- Sous forme de ruche
- Initialisation des cellules de bord "pleines"
- Pas d'arrivée
  - Politique de routage uniquement





Simulation

Modèle markovien

- Logiciel Psi

- Variables
  - Probabilité de routage
  - Nombre de clients/cellule au départ

- Résultats de la simulation
  - Routage vers le bas > 10% = surcharge des cellules de bord
  - Moins de 75% du nombre maximum de clients au départ

Simulation

Dans la pratique

- Dimensionnement
  - Cellule de taille > 2500
  - Taille de la map proportionnelle à la vitesse moyenne des joueurs, ainsi que la durée désirée

- Initialisation
  - Clients regrouppés par zone géographique

Implémentation du moteur de jeu

Couche réseau

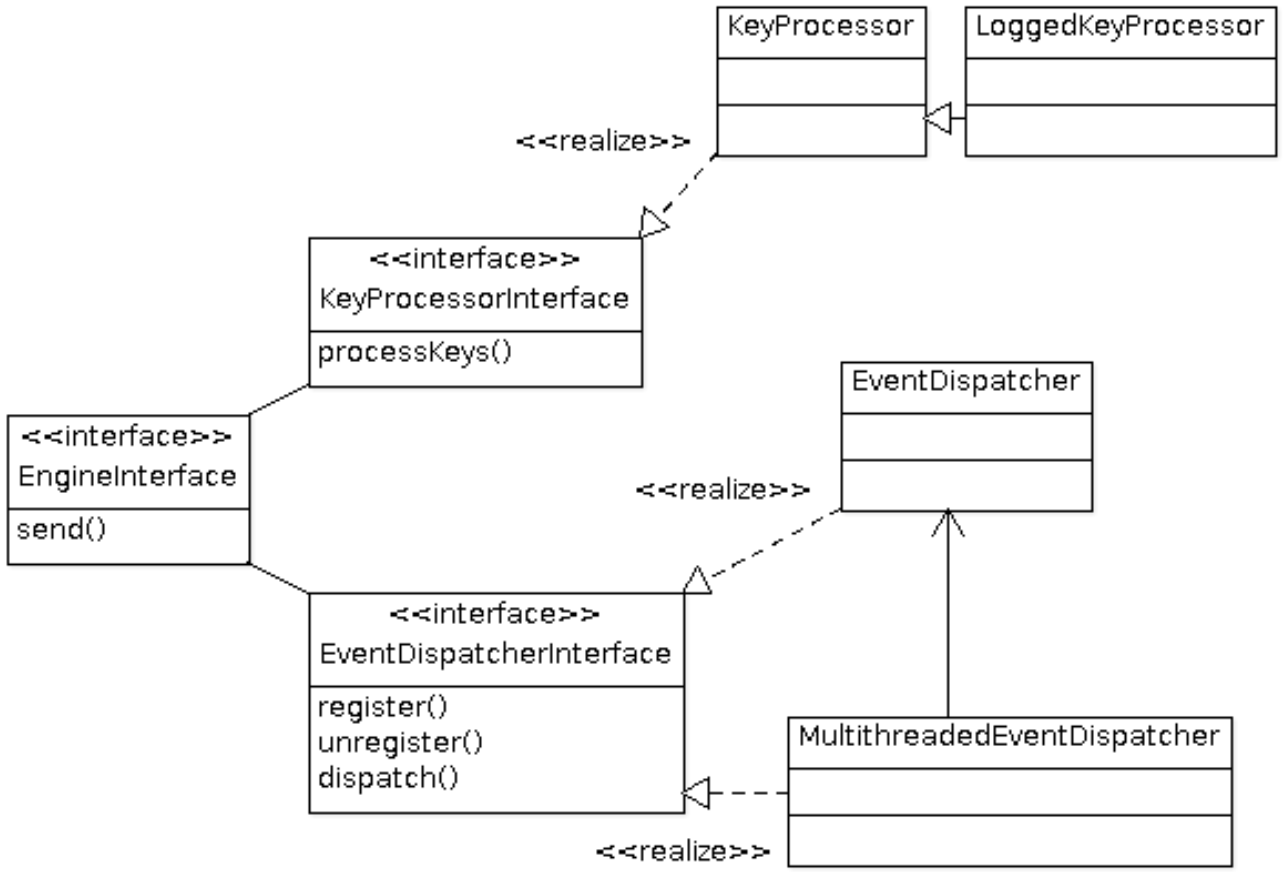
Choix de la technologie

- Java IO
  - API bloquante
  - 1 thread par client

- Java NIO
  - Programmation evenementielle
  - 1 thread pour tous les clients
  - Plus complexe

Couche réseau

Fonctionnement



Couche réseau

Utilisation

1. Connexion
    1. Creer Socket
    2. Engine: enregistrer le socket
    3. EventDispatcher: register

1. Déconnexion
    1. Fermer le socket
    2. Engine: evenement deconnexion
    3. EventDispatcher: unregister

Couche réseau

Extensions

- Engine générique
  - Ajout de comportements via des Extensions

- Extension
  - boot()/shutdown()
  - register()/unregister()
  - Conteneur
    - EventListener(s)
    - Variables d'état

Couche réseau

Multithreading

- Problematiques
  - NIO non thread-safe
  - Monothreading limite les performances

- Choix
  - 1 thread lit/ecrit
  - Pool de thread pour les evenements
    - Synchronisation

Couche applicative

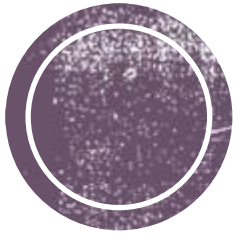
Implémentation

- Extensions indépendantes
  - Joueur
  - Leader
  - Manager

- Ajout/retrait à chaud
  - Devenir Leader en cours de partie

Problèmes rencontrés/Limites

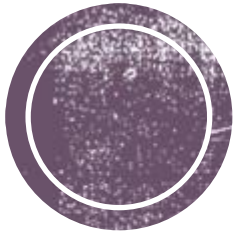
- Problèmes
  - Peu de documentation sur NIO
    - Retards dans le développement
  - Synchronisation complexe
  - Fuite(s) de mémoire
- Limites
  - Cellules non prises en charge
  - Authentification/sécurité non implémentée
  - Système peu tolérant aux pannes



Démonstration

Partie client

- La map, comprenant les obstacles
- Les clusters
- La position de ses voisins
- Le podium des 3 meilleurs joueurs
- Une estimation de son classement



Organisation

Planning

- 1ère partie :
  - Appropriation du sujet
- 2ème partie :
  - Implémentation initiale
  - Modélisation et simulation
- 3ème partie :
  - Implémentation suite
  - Elaboration d'un démonstrateur

Avez-vous des questions ?