

Étude de Cas 2016-2017

Christophe Saint-Jean
Matthieu Robert
Laurent Mascarilla
Joffrey Leblay

Licence 3

13 octobre 2016

Objectifs de l'étude

Sous la forme d'un projet long,

- Pratiquer la programmation *C* (1/2 index TIOBE)
- Appréhender le travail en équipe (trinômes)
- Gérer un projet informatique sur deux mois
- Travail collaboratif sur git.univ-lr.fr
- Manipuler réellement les outils de débogage, de compilation, de version, etc ...
- Environnement : *nux

Domaine de l'étude : Mots mélangés

Source : Site 20 minutes

Principe du jeu I

Les ingrédients :

- 1 Une grille de lettres
- 2 Un ensemble de mots

Objectif

Trouver tous les mots de la grille en un minimum de temps.

Exemple de grille (grid56)

5 6
MAISON
AERIEN
BARQUE
SERTIE
JOUETS
TRES
AERIEN
MER
RAS
NEES
RE
SI
JOUE

Principe du jeu II

Calcul du score en mode solo

- Tous les mots doivent être trouvés
- et rapidement...

$Score = \text{temps mis pour les trouver}$

Calcul du score en mode compétition

- Si un des joueurs est le premier à trouver un mot, il gagne autant de points que le mot est long.
- Si un des joueurs n'est pas le premier à trouver un mot, il gagne 0 point.

$Score = \text{nombre de points}$

Vous êtes libre d'inventer d'autres modes de calcul...

Architecture/Spécification I

On a choisi une architecture client serveur

- Un serveur "arbitre++"
- Des clients joueurs

On testera dans un premier temps sur une machine locale.

```
#define PORT1 8000  
#define PORT2 8001  
const char * ADDR_SERVER = "localhost";
```

Architecture/Spécification II

Déroulement du jeu du point de vue du client :

- Le client se connecte au serveur via le port PORT1
- S'authentifie via le message (une chaîne de caractères) :
JOIN <identifiant>
- Il reçoit la grille via le port PORT2
GRID <lig> <col> <grille>
- Il reçoit les mots via le port PORT2
WORDS <mot1> <mot2> ... <motn>
- Pour chaque mot trouvé, il envoie un message via le port PORT1
FOUND <identifiant> <mot> <i1> <j1> <i2> <j2>
- Suivant qu'il veule participer ou non au prochain jeu, il envoie un message via le port PORT1
JOIN <identifiant> ou LEAVE <identifiant>

Architecture/Spécification III

Déroulement du jeu du point de vue du serveur :

- La grille et les mots sont sélectionnés coté serveur (fichier / générateur)
- Le serveur recueille les joueurs intéressés via le port PORT1
- Démarre le jeu en envoyant la grille et les mots via le port PORT2

```
GRID <lig> <col> <grille>
WORDS <mot1> <mot2> ... <motn>
```

- Récolte les mots trouvés par les joueurs sur le port PORT1
- Valide, calcule et affiche les scores de chaque joueur.

Exemples de client à coder

TEXTE 1 Le joueur connaît la grille et les mots avant, il saisit manuellement les messages à envoyer sur le ports PORT1 et PORT2.

TEXTE 2 TEXTE1 + affichage de la grille dans la console

ALGO1 Un algorithme naïf.

GUI Un client graphique pour le joueur.

ALGOOn Des algorithmes moins naïfs.

Remarque : Le client connaît la grille seulement via le serveur.

Fonctionnalités du serveur

- Gestion des grilles (fichiers / générateurs + sauvegarde)
- Gestions des joueurs et du déroulement d'une partie.
- Validation des mots trouvés et calcul des scores.
- Affichage des scores / historique des scores.
- Un retour graphique.

Si tout est bien fait...

Vos clients et vos serveurs sont interchangeables entre projets...

Bibliothèques à utiliser

Nous avons regardé et choisi pour vous :

- La bibliothèque **0-MQ** pour la gestion des communications ([ici](#))

- Connect your code in any language, on any platform.
- Carries messages across inproc, IPC, TCP, TIPC, multicast.
- Smart patterns like pub-sub, push-pull, and router-dealer.
- High-speed asynchronous I/O engines, in a tiny library.

- La bibliothèque **SDL 1.2** et ses extensions (en particulier **SDL_image**) peuvent gérer pour vous :

- Le chargement et l'affichage d'image standard,
- L'affichage du texte (score, diverses choses),
- L'interaction avec l'utilisateur (clavier ou souris).

La version 1.2 bien qu'ancienne est déjà installée et surtout mieux référencée (est traitée par plus de tutoriels)...

- L'environnement de développement n'est pas imposé mais Eclipse est fortement recommandé...

Éléments C/ Outils de développement

- Configuration de votre environnement Git.
- Installation des bibliothèques requises...
- Configuration de votre projet pour votre IDE :
Bibliothèques pthread, dl, math, freetype, zmq et celles de la SDL...
- Rappel de C : typedef, struct, enum, macros
- Tableaux bi-dimensionnels, pointeurs : allocation, libération de la mémoire.
→ annexe “Mémoire” du rapport (ex. : memcheck/Valgrind)
- Débogueur : (éval. en séance)
- Documentation des fonctions principales : Doxygen (→ rapport)
- Annexe “Profilage” - ex. : callGrind/Valgrind (→ rapport)

Quelques éléments sur les mots/les grilles

- Un mot peut être présent plusieurs fois sur une grille...
 ⇒ Le serveur doit réellement valider les mots.
- Il sera très facile de générer des très grandes grilles pour mettre en compétition les clients "algorithmes".
- Une grille exemple est donnée sur moodle pour voir le format et s'entraîner.

Organisation au sein d'un trinôme

- Répartissez les tâches, discutez de l'environnement lors de la première séance.
- Séparation claire entre clients et serveur.
- La mise en place de l'environnement de travail peut être longue (≥ 1 séance).¹
- Pas mal de choses à comprendre dès le départ : git, 0-MQ, SDL...
- Bref, c'est un travail collectif!!!

1. Attention les vacances ..

Les éléments de l'évaluation

- Contrôle continu intégral → Présence ...
- La note ne sera pas forcément identique à l'intérieur d'un trinôme.
- Certaines compétences seront évaluées à la volée en TP (technique et avancement).
- Vote sur les fonctionnalités ?
- Dernière séance (le 09/12) :
 - Démonstration de l'application.
 - Soutenance d'une dizaine de minutes.
- Un petit rapport.

Rapport final

Le rapport demandé est un document propre non formel répondant aux questions suivantes.

- Quelles structures de données avez vous utilisé pour représenter en mémoire la grille, les mots, les joueurs, l'historique du score, etc ... ?
- Quel(s) algorithme(s) proposez vous pour explorer la grille ?²
- Analyser la complexité de l'algorithme proposé en fonction de la taille de la grille, etc .. ? son coût en mémoire.

Remise du rapport

Le rapport final (.pdf) sera rendu après la séance de démonstration (TP5).

Celui-ci comportera entre 5 et 10 pages³ hors titre, sommaire et annexes (documentation, rapports “Mémoire” et “Profilage”).

Tout retard dans la remise de ce document sera pénalisé.

2. Cette partie peut-être rédigée indépendamment de tout code.

3. Police Times New Roman 11pt, simple interligne