ESIAL - C

Mega Guitar Blindtest

Julien VAUBOURG && Guillaume GÉRARD
<julien@vaubourg.com>
<guillaume.gerard@esial.net>

3 juin 2011

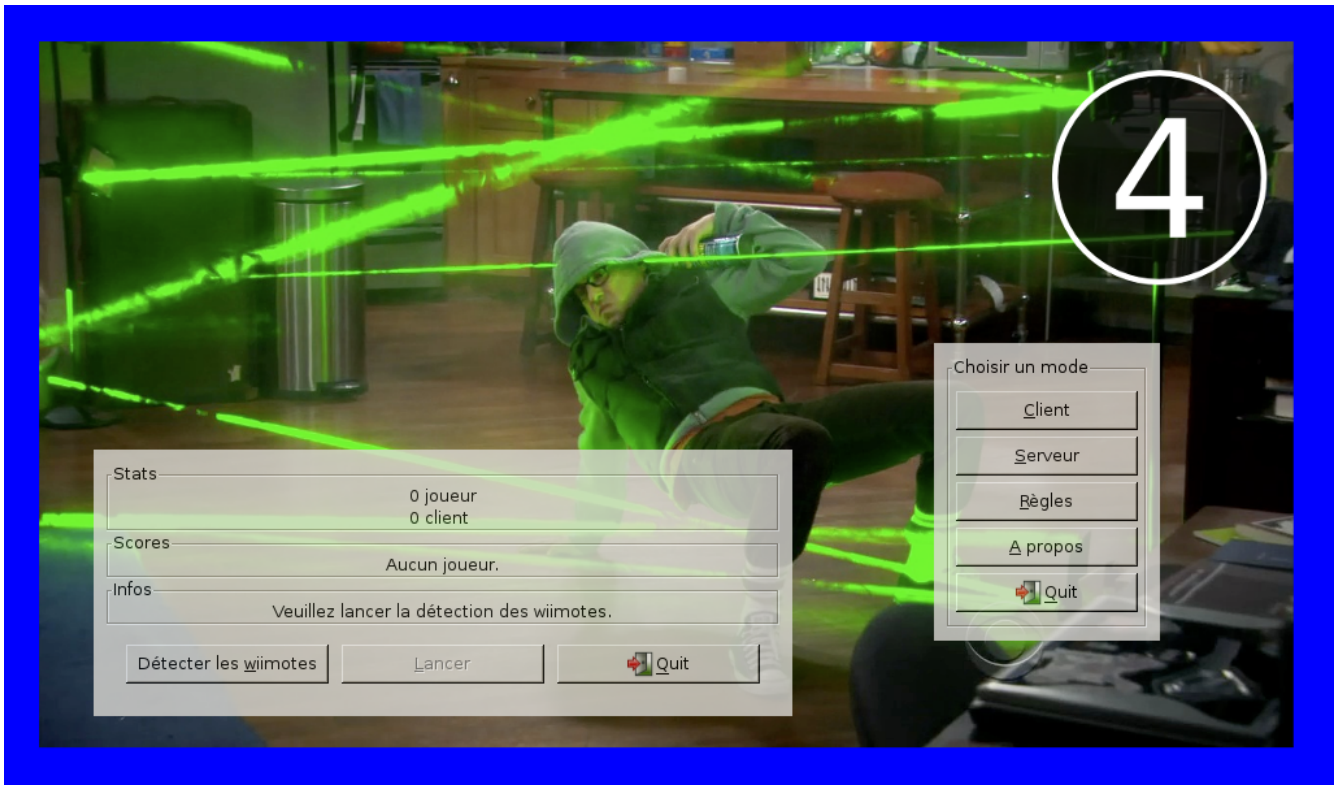
Table des matières

1 Jeu

1.1 Description

Venez vous affronter dans un *blindtest* cinématographique géant. Immergé dans une salle informatique, vous devrez reconnaitre et valider, à l'aide d'une guitare de Wii, un maximum de copies d'écran qui correspondent au son que vous entendrez. Serez-vous capable d'allier votre culture avec votre guitare pour imposer votre numéro sur un maximum d'écran ? Quatre joueurs maximum, une infinité d'ordinateurs.

1.2 Illustration



1.3 Licence

GNU GPL 3 et supérieur.

Peut apparaitre sur le site de l'ESIAL.

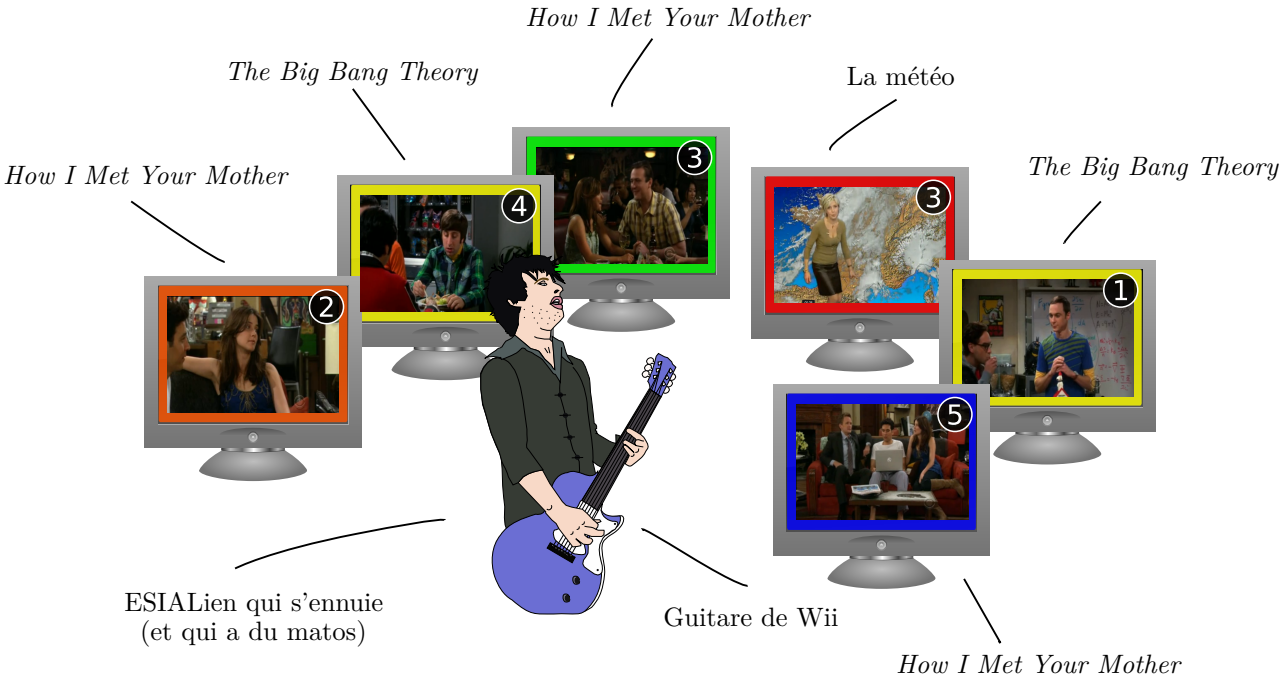
2 Détails

2.1 Matériel

Pour jouer, il faut :

- De 1 à 4 wiimotes équipées de guitares *Guitar Hero 3*
- De préférence une dizaine de machines GNU/Linux, reliées en réseau et équipées d'un moniteur
- Une des machines doit être équipée d'un récepteur bluetooth et de haut-parleurs

2.2 Mise en scène du jeu



2.3 Fonctionnement

Le jeu est lancé en tant que serveur sur le poste qui est équipé de la clé bluetooth. Sur tous les autres, il est lancé en tant que client, qui connaitra l'adresse du serveur.

La détection des wiimotes est lancée sur le serveur : chaque wiimote s'identifie, un numéro de joueur est attribué (la led correspondante est allumée sur les wiimotes), une vibration sur chacune confirme la détection.

Quelques clients sont lancés et affichent le logo du jeu : puisqu'il y a au moins une wiimote et un client de reliés au serveur, celui-ci autorise le début de partie.

Quatre joueurs (au maximum) équipés des guitares de Wii vont s'affronter sur un *bindtest* de génériques de séries. Selon la base de données, cela peut aussi être des musiques ou des dialogues de films. Ils se tiennent debout parmi les moniteurs des postes clients.

Un compte à rebours retentit, puis un générique de série s'enclenche. Tous les postes clients se mettent alors à afficher une copie d'écran de série, ainsi qu'un liseré de couleur, et un nombre.

Le but est de valider un maximum de moniteurs qui correspondent à la série du générique en cours, en fonction de la couleur (bouton de la guitare à enclencher) et du nombre affichés (nombre de fois qu'il faut gratter la guitare pour valider).

Exemple avec le schéma ci-dessus : Si la musique de *The Big Bang Theory* retentit, le joueur doit laisser appuyé le bouton jaune, puis gratter quatre fois et relâcher. Le second moniteur se valide. S'il avait gratté une seule fois avant de relâcher le bouton, c'est l'avant dernier qui se serait éteint.

Remarque : En admettant qu'il souhaite valider le second et le dernier, il aurait pu gagner du temps ! Presser les boutons jaune et bleu, gratter quatre fois, relâcher le jaune (et donc valider le second), gratter encore une fois, et relâcher le bleu (et donc valider le dernier).

Si le moniteur ainsi validé correspond bien à la série du générique en cours, un son de victoire retentit, la guitare vibre, et le moniteur passe en vert en affichant le numéro du joueur responsable. Celui-ci gagne des points.

Sinon, c'est un son de défaite qui retentit, la guitare vibre plus méchamment, et le moniteur passe en rouge en indiquant le numéro du responsable. Il perd des points.

Lorsqu'un joueur pense qu'il n'y a plus de moniteurs corrects à valider, il doit s'empresser de donner un coup de

vibrato à sa guitare. S'il avait juste, un son de victoire retentit, sa guitare vibre, et son numéro de joueur est affiché sur tous les moniteurs. Il gagne un bonus pour avoir validé le changement de générique. Sinon, il prend un malus.

Si tous les moniteurs corrects n'ont pas été validés ou qu'aucun joueur n'a demandé le changement, une limite de temps indique que c'est trop tard.

La musique s'arrête, le compte à rebours réapparait alors sur tous les moniteurs et la boucle du jeu recommence, en laissant les joueurs accumuler leurs points (ou partir dans le négatif).

2.4 En plus

Sur le serveur, durant toute la partie, les scores sont affichés en temps réel, avec le nombre de clients et de joueurs.

Durant la partie, les joueurs peuvent mettre le jeu en pause en appuyant sur le bouton moins ou plus de leur guitare. Le bouton A quitte le jeu (serveur et clients).

Pour compléter la base de données, il suffit d'ajouter un répertoire, rempli de copies d'écrans nommées *screen_i.png* (i allant de 0 à l'infini) et disposant d'un fichier *sound.ogg* correspondant au générique, au dossier *data/series/*.

Un client peut rejoindre ou quitter le serveur à n'importe quel instant de la partie, sans prévenir.

3 Développement

3.1 Notions exploitées

Le jeu recours à un certain nombre de notions majeures :

- Gestion des wiimotes bluetooth (*wiiuse*)
- Connexions réseau multiples (*socket*)
- Multi-processus (*pthread*)
- Lecture en flux de fichiers musicaux vorbis (*openal*)
- Interfaces graphiques (*gtk*)

3.2 Difficultés rencontrées

Chacune des notions évoquées ci-dessus nous a contraint à nous former à de nouveaux domaines.

La difficulté majeure de ce jeu est de gérer tous les événements en même temps. À un même instant, le serveur doit gérer n clients connectés (ainsi que leur arrivée ou leur départ), m joueurs qui envoient des événements wiimotes, son propre affichage, et la boucle de jeu.

À noter que nous avons souhaité que les clients soient totalement libres. Ainsi, à n'importe quel instant du jeu, un nouveau client peut se connecter au serveur et participer. Il affichera alors le logo du jeu, puis sera entrainé dans la boucle de mise à jour des clients qui suivra, affichant sa propre copie d'écran au prochain tour.

La lecture des flux audio nous a posé, et nous pose toujours problème. Des erreurs OpenAL persistent de manière aléatoire, sans pour autant déranger le jeu. Une solution serait de commencer par ne pas lire le sons d'événements en flux, puisqu'il s'agit de sons brefs qui peuvent être mis en mémoire.

Le projet doit être compilé en statique pour pouvoir fonctionner sans encombre sur des machines neutres : nous avons découvert que la compilation statique pouvait être très fastidieuse, dans la mesure où toutes les dépendances ne proposent pas de fichier adapté.

La conception sans objets a été perturbante, nous avons eu du mal à trouver l'équilibre entre la philosophie procédurale qu'impose le langage, et la possibilité de tenter de se rapprocher le plus possible du modèle objet.

3.3 Travail en équipe

Nous avons utilisé le subversion mis à disposition par l'École pour travailler.

Nous estimons le travail de développement à une centaine d'heures, en incluant la conception et les recherches. Nous avons particulièrement fait attention à la propreté de la conception (il n'y a, par exemple, aucune variable globale).

4 Sources

Nous avons principalement abusé de :

- http://www.wiiuse.net/docs/
- http://franckh.developpez.com/tutoriels/posix/pthreads/
- http://broux.developpez.com/articles/c/sockets/
- http://www.gtk-fr.org/wakka.php?wiki=letutorial
- http://developer.gnome.org/gtk/2.24/
- http://loulou.developpez.com/tutoriels/openal/premiers-pas
- http://www.devmaster.net/articles/openal-tutorials/lesson8.php
- /usr/bin/man