UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
INFO0601/INFO0604

Projet commun INFO0601/INFO0604

Simulateur de pêche à la ligne

Le but de ce projet est de réaliser un jeu à deux joueurs en réseau et multi-threadé.

1 But du jeu

Nous souhaitons réaliser une simulation de pêche à la ligne à deux joueurs. À l'écran est représenté un étang (un grand carré tout bleu) de dimensions variables dans lesquels des poissons (de jolis carrés jaunes) se déplacent de case en case. Les joueurs ont la possibilité de déposer la ligne de leur canne à pêche (clic gauche de la souris dans l'étang) à un endroit donné dans le but d'attraper un poisson. Si un poisson arrive à proximité (case du haut, du bas, de droite ou de gauche), il est alors attrapé et attend que le pêcheur relève sa canne à pêche (si le joueur ne la relève pas assez vite, le poisson s'échappe). À noter qu'un joueur ne peut poser sa ligne que toutes les 3 secondes. Il peut la poser au même endroit que celle de l'autre joueur (de toute façon, il ne voit pas celle de son adversaire). Si un poisson arrive à proximité de deux lignes, il en choisit une aléatoirement.

Chaque fois que l'utilisateur pêche un poisson, il rapporte un point et de l'argent en fonction du poisson. L'argent peut débloquer des coups spéciaux qui sont les suivants :

- Le pneu (150 poireaus [1]) : le joueur pose un pneu dans l'étang (un carré noir invisible pour son adversaire). Si un joueur pose sa canne à pêche à côté d'un pneu, au moment où il tente de la relever, il est bloqué pendant 3 secondes (le poisson éventuel s'échappe, le pneu disparaît et le joueur qui a posé le pneu gagne 1 point sauf si c'est son propre pneu).
- La dynamite (200 poireaus) : le joueur pose une dynamite dans l'étang (un carré rouge invisible pour son adversaire). En explosant, tous les poissons situés dans un carré de 5 par 5 (centré sur la dynamite) meurent et le joueur gagne les points et l'argent associés. Si la ligne d'un joueur est posée dans le rayon d'action de la dynamite, il est bloqué pendant 3 secondes après l'explosion.
- Le requin (300 poireaus) : le joueur lâche un requin dans l'étang (un carré vert mais jaune pour l'adversaire). Si un joueur pêche ce faux poisson, il perd un point et 100 poireaus. Le requin se comporte comme un poisson.
- Le mode furtif (500 poireaus) : le joueur peut poser ou relever sa ligne à côté des poissons sans que ça les fasse fuir. Par contre, s'il remonte un pneu, s'il est pris par une dynamite ou s'il pêche un requin, cela annule le mode furtif.

Dès qu'un évènement se passe sur une case (pose de la ligne ou relevé de la canne à pêche, pose d'un pneu, d'une dynamite ou d'un requin), les poissons [2] fuient la zone (carré de 3 cases par 3 cases). Ils passent dans un état 'fuite' pendant 5 tours et ne peuvent pas être attrapés par une canne à pêche (mais peuvent être dynamités normalement).

Pour simplifier, il ne peut y avoir qu'un objet par case : le dernier objet posé remplace celui qui était déjà présent. Il ne peut y avoir qu'un seul poisson (requin compris) par case. La ligne peut être

1. Le nom de cette monnaie chère aux pêcheurs est formée de la contraction de poisson et euro. Le 's' n'est donc pas une erreur...par contre, le 'eau' oui.
2. Si l'évènement est un relevé de canne à pêche, le poisson attrapé ne s'échappe pas.

posée sur ce celle de l'adversaire, sur un objet ou sur un poisson (mais qui prendra la fuite, bien sûr, sauf en mode furtif).

Régulièrement, des poissons sont générés. Ils sont de trois types différents, représentés avec un carré jaune contenant la valeur 1, 2 ou 3. Ils valent respectivement 100, 200 et 500 poireaus. La rareté des poissons générés dépend de leur valeur. Le requin est affiché comme un type aléatoire pour tromper l'ennemi.

Dès qu'un joueur atteint le score de 15 points, il gagne.

2 Description des applications

Vous devez développer deux applications : un client qui permet à un joueur d'attraper les innocents poissons et un serveur qui permet de mettre en relation deux clients.

Le serveur prend en arguments l'adresse IP, le numéro de port UDP sur lequel il va attendre les clients et les dimensions de l'étang (largeur + hauteur).

Le client prend en arguments l'adresse IP et le numéro de port UDP du serveur.

Le serveur se met simplement en attente de requêtes qu'il place dans une file d'attente. Dès qu'il y a au moins deux clients, il crée une partie sur un numéro de port TCP donné qu'il envoie aux clients.

3 Gestion des synchronisations et du parallélisme (INFO0604)

Chaque poisson doit être exécuté dans un thread spécifique. Vous pouvez utiliser autant de *threads* que nécessaire. De manière générale, votre projet doit proposer des solutions aux principaux problèmes de gestion de ressources, de synchronisation et de parallélisme induits par les *threads*. Plus spécifiquement, il doit répondre aux questions suivantes :

- Les *threads* s'exécutent-ils seulement lorsqu'ils ont une raison de le faire ? Comment suspendre leur exécution lorsque cela est pertinent ?
- Est-ce que les accès concurrents sont correctement et efficacement gérés ? Comment assurer la protection des données tout en évitant le gaspillage des ressources et les verrous mortels (*deadlocks*) ?
- Comment rendre l'affichage de la simulation pertinent tout en permettant aux *threads* de s'exécuter de façon minimalement parallèle ?
- Lorsqu'un *thread* est détruit, est-ce que la libération des ressources qu'il utilise est gérée correctement, efficacement et le plus rapidement possible ?

Dans votre rapport, vous devez présenter les principaux problèmes auxquels vous avez été confrontés sur ces aspects et justifier les solutions éventuellement proposées et implémentées.

Pour les étudiants ne suivant pas INFO0604, il n'est pas nécessaire que les poissons soient gérés par des fils différents.

4 Programmation système (INFO0601)

La partie programmation système sera évaluée en fonction de différents critères :

- La gestion des connexions réseau (respect des protocoles demandés, messages échangés) ;
- Les appels systèmes vérifiés et la gestion des erreurs.

Le code n'étant pas suffisant en lui-même, les parties non expliquées dans le rapport ne seront pas prises en compte dans la notation. De même, la maîtrise du langage C et de la compilation séparée seront pris en compte. Pour les étudiants ne suivant pas INFO0601, le deuxième joueur correspond à l'ordinateur qui jouera de manière aléatoire : il n'est pas nécessaire d'implémenter de serveur ni de connexions réseau.

5 Notations

Les points de la note finale sont répartis sur les trois parties suivantes : le code, le rapport et la soutenance sur machine. Le code, écrit en C, doit être structuré correctement (fichiers sources et en-têtes) et un makefile doit être fourni. Un effort doit être fait sur les commentaires, sur la présentation des sources (indentation), ainsi que sur le nom des structures, des variables et des fonctions. De plus, un fichier README doit être fourni afin de spécifier les paramètres de compilation et d'exécution de votre programme. La note du rapport tiendra compte aussi bien du contenu que de la forme (la table des matières, l'ortaugraffe, le plan). Le rapport doit présenter l'ensemble du travail réalisé pour le projet sans contenir de code C (ou très peu). Il est nécessaire de présenter les structures utilisées lors de l'analyse ou pendant l'exécution, en expliquant vos choix. Des schémas seront appréciés pour étayer vos explications. Les algorithmes principaux du projet pourront être présentés sous forme algorithmique ou sous la forme de diagrammes, mais sans pour autant négliger une description claire, en dehors de l'algorithme. Vous devrez mettre en œuvre au minimum l'ensemble des recommandations citées précédemment. Vous pourrez ajouter d'autres fonctionnalités, utiliser d'autres éléments (*threads*, notions vues en Info0601, *etc.*). Dans ce cas, vous devrez le préciser dans le rapport. De même, si des éléments ne sont pas réalisés, ou que des simplifications ont été choisies, il faudra aussi le préciser dans votre rapport. La soutenance sera constituée uniquement d'une présentation sur machine de 20 minutes, pendant laquelle les enseignants vous poseront des questions. Vous devez prévoir une répartition équitable du temps de présentation des membres du binôme. Chacun doit montrer sa maîtrise du sujet et sa propre contribution. Des lacunes à ce niveau seront pénalisées non seulement au niveau individuel, mais aussi de façon solidaire. Le code envoyé sera compilé par nos soins, avec le `makefile` fourni, sur une ou plusieurs machines d'une salle de TP sous *Ubuntu* : les options de compilation doivent être impérativement celles vues en cours d'Info0601 (-Werror –ansi –pedantic –Wall). Il est conseillé de préparer votre démonstration ainsi que différents scénarii, afin d'éviter l'improvisation. Les soutenances sont prévues dans la semaine du 6 avril 2020. La date et l'horaire définitifs, ainsi que la planification des soutenances, vous seront confirmés par mail. Au terme de votre projet, vous devrez remettre votre code et votre rapport. Le tout doit être inclus dans un répertoire dont la structure arborescente est la suivante :

- Projet_Nom1_Nom2
  - Rapport_Nom1_Nom2.pdf
  - Code
    - ▷ Fichiers sources
    - ▷ `makefile`

"Nom1" et "Nom2" sont les noms de famille des deux étudiants du binôme. Pensez à retirer les fichiers non nécessaires (.o, exécutable(s), fichiers temporaires, etc.) avant la remise. L'ensemble doit être compressé dans une archive (.zip ou .tar.gz) identifiée avec le nom des auteurs et déposé sur *Moodle* au plus tard le dimanche 5 avril 2020 à 23h00. À cet effet, il est fortement recommandé de ne pas attendre la dernière minute pour envoyer votre projet !