UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
INFO0601/INFO0604

Projet commun INFO0601/INFO0604

Jeu d'aventure

Le but de ce projet est de réaliser un jeu multi-joueurs en réseau et multi-threadé. Il peut être réalisé en monôme ou en binôme.

1 But du jeu

Nous souhaitons réaliser un jeu d'aventure collaboratif. Chaque joueur contrôle un héros et le déplace dans un monde infini. Son seul but : pourfendre les ennemis et ramasser artefacts et trésors.

Un héros est caractérisé par cinq attributs : points de vie, armure, force, vitesse de déplacement et vitesse de frappe. Leurs valeurs de base sont fixées au début de la partie mais peuvent évoluer. Si le nombre de points de vie est à 0, le héros meurt, ses attributs sont ramenés à leurs valeurs de base et il est replacé à son point de départ. Tout au long de la partie, il ne peut pas regagner plus de points de vie que son maximum mais les artefacts peuvent augmenter ce maximum. Le héros possède également des points d'expérience qu'il gagne lorsqu'il tue des monstres ou des adversaires. Arrivé à un certain niveau, cela augmente les attributs du héros.

Le monde est potentiellement infini. Il est constitué d'un assemblage de cartes dont les dimensions sont de 40 (largeur) par 20 (hauteur) cases. Chaque carte est créée à l'aide d'un éditeur et est indépendante des autres. Une carte contient des cases de base : l'herbe, l'eau, le sable et les cailloux. Sur ces cases (sauf l'eau) peuvent se trouver des éléments : un trésor, un monstre, un artefact ou un obstacle. Les monstres et les artefacts sont caractérisés par des noms de longueur variable ainsi que par les cinq attributs définis précédemment pour le héros.

Lorsqu'une nouvelle partie est démarrée, un fichier unique est créé, nommé `monde.sav`. Il contient les cartes constituant le monde et est construit au fur-et-à-mesure des déplacements des héros. La première carte correspond au point de départ de tous les héros (et l'endroit où ils réapparaissent quand ils meurent). À chaque fois que le héros se déplace et quitte une carte (vers le haut, la droite, le bas ou la gauche), il passe sur une nouvelle carte. L'état de l'ancienne carte est alors sauvegardé dans le fichier `monde.sav` et les ressources associées sont libérées de la mémoire. Si le héros retourne par la suite sur une carte déjà visitée, il la retrouve dans le même état (monstres tués ou blessés, artefacts, trésors, etc.). Les cartes sont choisies aléatoirement parmi celles situées dans un répertoire spécifique. Il est possible que des endroits différents du monde correspondent à une même carte.

Le héros est déplacé par le joueur à l'aide des touches fléchées. Il ne peut pas se déplacer dans l'eau, ni sur les obstacles. S'il tente d'aller sur une case contenant un monstre ou un héros adverse, il attaque celui-ci. Si les points de vie de l'entité attaquée passent à 0, elle meurt et en fonction de ses caractéristiques, le héros gagne des points d'expérience.

Un héros peut ramasser des artefacts (pas plus de 5). Ceux-ci permettent de modifier un ou plusieurs de ses attributs (en moins comme en plus !). Il est possible de les déposer à tout moment si le joueur trouve un meilleur artefact.

Le héros peut également ramasser des trésors qui sont positionnés sur les cartes par défaut, mais qui peuvent apparaître aléatoirement toutes les 1 ou 2 minutes n'importe où sur la carte. L'effet

d'un trésor est choisi aléatoirement : cela peut être soit un gain de points de vie (sans dépasser le maximum), soit une des pièces du **grand-tout**.

Les pièces du grand-tout se cumulent et peuvent être activées à tout moment (et toutes en même temps). Une fois activées, toutes les pièces que le héros possède disparaissent. S'il n'y a qu'une pièce, cela permet de ralentir les monstres du monde entier pendant 1 minute. Deux pièces permettent de stopper tous les monstres qui ne peuvent ni se déplacer, ni attaquer pendant 1 minute. Trois pièces permettent de stopper tous les monstres pendant 1 minute et leur fait perdre la moitié de leurs points de vie. Le héros ne peut pas récupérer plus de 3 pièces simultanément (s'il en récupère une, alors elle est perdue à tout jamais). De même, il ne peut pas activer ses pièces si un autre héros a déjà activé les siennes. Il doit attendre la fin de l'effet. Si jamais un monstre est attaqué, l'effet est immédiatement interrompu.

2 Description des applications

Vous devez développer trois applications : un éditeur de carte, un client qui permet à un utilisateur de diriger son héros et un serveur qui permet de mettre en relation les clients.

L'éditeur de carte a déjà été présenté dans un sujet de TP d'INFO0601.

Le serveur prend en arguments le numéro de port TCP sur lequel il va attendre les connexions des clients et le nom du répertoire contenant les cartes. Il exécute la simulation mais ne contient pas d'interface *ncurses*.

Les clients prennent en paramètre l'adresse IP et le numéro de port du serveur. L'interface est constituée de plusieurs fenêtres. La fenêtre «Informations» affiche les messages quelconques : attaque d'un monstre (en affichant son nom et ses caractéristiques), activation d'une pièce du grand-tout par un autre héros, la mort d'un autre héros, etc. La fenêtre «Carte» affiche l'état de la carte sur laquelle se trouve le héros du joueur. La fenêtre «Attributs» affiche les attributs du joueur : points de vie et points de vie maximum, armure, force, vitesse de déplacement. Elle affiche également le nombre de pièces du grand-tout et les artefacts en sa possession.

Les ordres de déplacement sont envoyés au serveur et l'état de la carte est reçu périodiquement. Le héros se déplace grâce aux flèches du clavier. La barre espace permet d'activer les pièces du grand-tout. Les touches 1 à 5 permettent de déposer un artefact sur la case située à côté du héros.

3 Gestion des synchronisations et du parallélisme (INFO0604)

Chaque monstre et héros doit être exécuté par un *thread* spécifique. Vous pouvez utiliser autant de *threads* que nécessaire. Votre projet doit, entre autres, implémenter les éléments suivants :

- Affichage de la simulation par un *thread* dédié ;
- Gestion du blocage des monstres par variable(s) de condition ;
- Destruction des monstres ou des héros par annulation retardée (*deferred*) des *treads*.

De façon générale, votre projet doit proposer des solutions aux principaux problèmes de gestion des ressources, de synchronisation et de parallélisme induits par les *threads*, notamment la gestion des accès concurrents aux données, l'allocation/libération/utilisation des ressources et l'utilisation correcte et judicieuse des fonctionnalités de la bibliothèque `Pthread`. 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, les simplifications du projet seront à évoquer avec M. Rabat.

4 Programmation système (INFO0601)

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

- La gestion des connexions réseau ;
- La gestion des fichiers binaires ;
- La vérification du retour des appels systèmes 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 prises en compte.

5 Notations

Les points de la note finale sont répartis sur les parties suivantes : le code, le rapport et l'application (les fonctionnalités implémentées).

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 `lisezMoi.txt` doit être fourni afin de spécifier les paramètres de compilation et d'exécution de votre programme. Le code envoyé sera compilé sur la VM fournie, à l'aide du `makefile` : les options de compilation doivent être impérativement celles vues en cours d'INFO0601 (`-Werror -Wall`).

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.

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 10 avril 2022 à 23h30. À cet effet, il est fortement recommandé de ne pas attendre la dernière minute pour déposer votre projet !