UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
INFO0601/INFO0604

Projet commun INFO0601/INFO0604

Jeu de plateformes

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

> *29 janvier 2197. Votre vaisseau spatial a été capturé par une race extra-terrestre alors que vous étiez dans votre sommeil cryogénique, en route vers une colonie humaine lointaine. Votre seule chance de survivre et de continuer votre voyage : participer à un jeu diabolique avec plusieurs autres prisonniers et sortir en premier d'un labyrinthe rempli de pièges.*

Vous devez réaliser un jeu de plateformes multi-joueurs en réseau. Chaque joueur contrôle un personnage et le déplace dans un monde contenant plusieurs tableaux (des pièces). L'objectif est d'atteindre la sortie avant les autres, en récupérant les clés pour passer les portails, tout en évitant les ennemis (robots et sondes), les pièges et les bombes des adversaires.

Tous les joueurs commencent simultanément au niveau de la porte d'entrée (les joueurs peuvent occuper la même place). Ils ne peuvent se déplacer que vers la droite ou la gauche (jusqu'à rencontrer des obstacles) ou monter et descendre les échelles. Les robots se déplacent dans une direction jusqu'à rencontrer un obstacle puis repartent dans le sens opposé. Ils ne peuvent pas monter les échelles. Les sondes se déplacent en l'air aléatoirement dans les quatre directions.

Un joueur possède par défaut 5 vies dans sa jauge (et ne peut pas en avoir plus). Il peut en perdre s'il tombe d'une plateforme (volontairement ou à cause d'un piège), s'il entre en contact avec un ennemi (robot/sonde) ou s'il est dans le rayon d'explosion d'une bombe (nous reviendrons ci-après sur le fonctionnement des bombes). Des objets "vie" sont disséminés dans le monde : cela remet la jauge au maximum. Dès qu'un joueur passe dessus (sauf si sa jauge est pleine), l'objet disparaît et réapparait un peu plus tard. Quand un joueur perd une vie, il devient invincible pendant 3 secondes. S'il perd toutes ses vies, il a perdu. Il a la possibilité de recommencer au point de départ (ce n'est pas automatique), mais dans ce cas, il perd toute sa progression (clé, bombes).

Dès qu'un joueur passe sur un objet "bombe", il récupère au maximum 3 bombes (comme pour l'objet "vie", l'objet disparaît et réapparait au bout d'un certain temps). Un joueur peut poser une bombe (elle est statique, ne tombe pas et apparaît au niveau des pieds de son personnage). Lorsqu'elle explose, tous les personnages dans un rayon de 5 blocs sont touchés (les objets ne sont pas détruits par les bombes). Les joueurs touchés perdent une vie et restent paralysés pendant 5 secondes sauf si un ennemi les touche ou qu'une autre bombe explose (dans les deux cas, il perd une vie). Les ennemis sont simplement immobilisés pendant 5 secondes et ne peuvent plus infliger de dégâts aux joueurs (pour passer au travers un ennemi, on peut le paralyser avec une bombe).

Les niveaux comportent également les éléments suivants :

- Les portes : à part la porte d'entrée (le point de départ) et de sortie (qui permet de gagner), elles permettent de passer d'un niveau à un autre. Les joueurs et les ennemis peuvent passer au travers les portes. Seuls les joueurs peuvent y entrer.

- Les échelles : elles permettent aux joueurs de monter ou descendre. Pour descendre, il doit y avoir une échelle sous les deux pieds du joueur. Pour monter, il faut qu'il y ait une échelle au niveau des pieds du joueur. Les joueurs et les ennemis peuvent passer à travers les échelles.
- Les blocs (murs et sols) : les joueurs et les ennemis ne peuvent pas les traverser.
- Les pièges : ce sont des blocs qui apparaissent et disparaissent régulièrement. Si un joueur se trouve sur un piège au moment de sa réapparition, alors il meurt directement. Les ennemis voient les pièges comme des murs (ils ne peuvent donc pas être sur un piège au moment de sa réapparition).
- Les portails : il y en a quatre, de couleurs différentes ; pour traverser un portail, il faut que le joueur possède la clé correspondante. Les ennemis peuvent passer au travers les portails.
- Les clés de portail : le joueur récupère la clé dès qu'il passe dessus et l'objet ne disparaît pas.

2 Description des applications

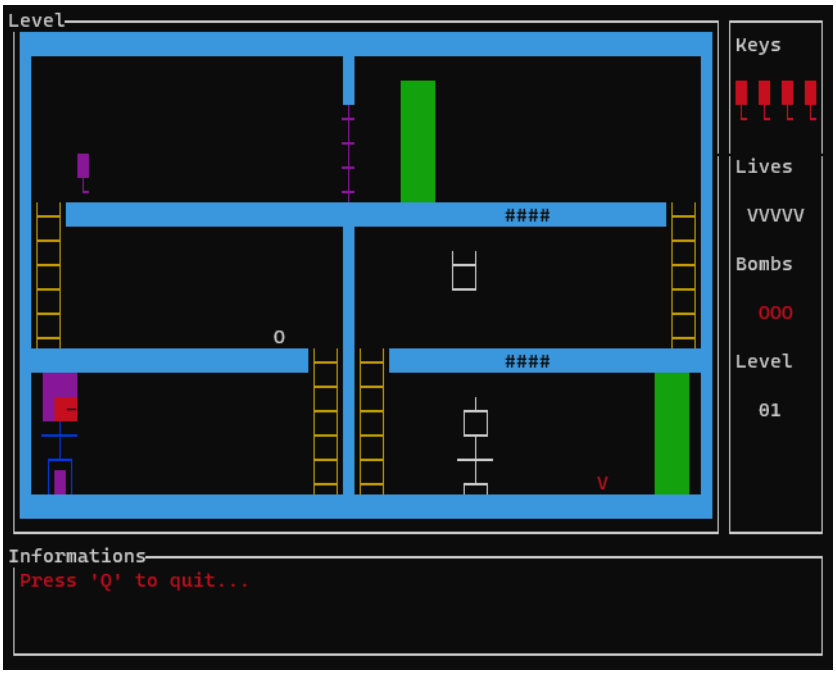
Vous devez développer trois applications : un éditeur de mondes, un client qui permet aux joueurs de se connecter au monde et un serveur qui permet de mettre en relation les clients.

L'éditeur de mondes a déjà été présenté dans un sujet de TP d'INFO0601. Il permet de créer des mondes et de les sauvegarder dans un fichier binaire via une structure spécifique.

Le serveur met en relation les joueurs. Il indique quelles sont les parties qui sont en attente de démarrage (il précise le nombre de joueurs actuels et attendus, ainsi que le nom du monde utilisé). Un joueur peut s'inscrire à une partie non démarrée ou bien en créer une nouvelle. Dans ce cas, il choisit parmi les mondes proposés (le serveur lui en fournit la liste) et un nombre de joueurs maximum.

Lorsqu'un joueur démarre une partie ou qu'il en sélectionne une, il est en attente d'une réponse du serveur (un message UDP). Lorsque le nombre de joueurs nécessaire est atteint pour démarrer une partie, le serveur crée une socket TCP avec un numéro de port aléatoire (nous laissons le système d'exploitation choisir) et envoie ce numéro de port aux joueurs. Ceux-ci peuvent alors s'y connecter. La partie démarre une fois que tous les joueurs sont connectés. Pour gérer la socket, le serveur crée un fils (et pas de *thread*). Ce dernier gère la partie et les communications entre les joueurs.

Les clients prennent en paramètre l'adresse IP et le numéro de port UDP du serveur. Pour simplifier, le choix de créer une partie ou d'en rejoindre une peut être réalisé à l'aide d'un menu dans le terminal. Seul le jeu nécessite la bibliothèque *ncurses*. L'interface ressemble à la capture suivante :



3 Gestion des synchronisations et du parallélisme (INFO0604)

Chaque joueur et ennemi (robot/sonde) 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 (suite aux bombes) par variable(s) de condition ;
- Gestion des pièges par des *threads* dédiés ;
- Destruction des ennemis ou des joueurs 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 INFO0601 ou INFO0604, les simplifications du projet seront à évoquer avec le responsable de la matière.

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 ;
- La gestion éventuelle via des signaux pour l'arrêt des applications.

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, la vidéo de présentation 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.

Vous devez également créer une vidéo de 9 à 11 minutes permettant de montrer l'exécution de votre application et de présenter les éléments importants de votre projet. Vous pouvez également vous reposer sur un diaporama. Les deux étudiants du binôme doivent présenter. La vidéo est inutile si le projet ne s'exécute pas !

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 le dépôt. L'ensemble doit être compressé dans une archive (.zip ou .tar.gz) identifiée avec le nom des auteurs. Cette archive et la vidéo doivent être déposés sur *Moodle* au plus tard le dimanche 9 avril 2023 à 23h30. À cet effet, il est fortement recommandé de ne pas attendre la dernière minute pour déposer votre projet !