Programmation Système UNIX

Zappy Bibicy

Contact b-psu-330@epitech.eu

Table des matières

Consignes

- Le projet est à faire par groupe de 4 à 6 personnes.
- Votre projet doit être à la norme.
- Vous pouvez utiliser l'intégralité de la librairie C standard.
- Le rendu doit avoir un fichier auteur dans lequel vous devez mettre les logins des différents membres du groupe comme suit :

```
1 (user@host 42) cat auteur
2 login1
3 login2
4 login3
5 login4
6 login5
7 login6
8 (user@host 43)
```

- Les sources doivent être rendues sur le dépôt `SVN`
- Rendu :
  `svn+ssh ://kscm@koala-rendus.epitech.net/zappy-année_promo-login_x`

Environnement

.1 Géographie

Le jeu consiste à gérer un monde et ses habitants. Ce monde, nommé `Trantor` est géographiquement constitué de plaines ne comprenant aucun relief : ni cratère, ni vallée, ni montagne. Le plateau de jeu représente la totalité de la surface de ce monde comme un planisphère. Si un joueur sort par la droite du plateau, il rentrera par la gauche.

Le jeu se joue par équipe. L'équipe gagnante est celle dont six joueurs auront atteint l'élévation maximale.

.2 Ressources

Le milieu dans lequel on évolue est plutôt riche en ressources, tant minières qu'alimentaires. Ainsi, il suffit de déambuler sur cette planète pour découvrir cette succulente nourriture ou une multitude de pierres de diverses natures.

Ces pierres sont de six genres distincts. Il existe :

1. la linemate,
2. la deraumêre,
3. le sibur,
4. la mendiane,
5. le phiras,
6. la thystame

La génération des ressources est réalisé par le serveur. Cette génération doit être aléatoire.

.3 Les Activités

Les habitants de Trantor vaquent à deux types d'occupation :

- se nourrir
- rechercher et amasser des pierres

L'élévation représente une activité importante du Trantorien.

.4 Individus

L'habitant est pacifique. Il n'est ni violent ni agressif. Il déambule gaiement en quête de pierres et s'alimente au passage. Il croise sans difficulté ses congénaires sur la même unité de terrain et voit aussi loin que ses capacités visuelles le lui permettent. Le Trantorien est immatériel, il est flou et occupe toute la case dans laquelle il se trouve. Il est impossible de distinguer son orientation lorsqu'on le croise. La nourriture que le trantorien ramasse est la seule ressource dont il a besoin pour vivre. Une unité de nourriture permet de survivre $126 \times t$ unités de temps.

.5 Le rituel d'élévation

L'objectif de tous est de s'élever dans la hiérarchie Trantorienne. Ce rituel qui permet d'accroître ses capacités physiques et mentales doit être effectué selon un rite particulier. Il faut en effet réunir sur la même unité de terrain :

- une combinaison de pierres
- un certain nombre de joueurs de même niveau

Un joueur débute l'incantation et l'élévation est alors en cours. Il n'est pas nécessaire que les joueurs soient de la même équipe. Seul leur niveau importe. Tous les joueurs du groupe réalisant l'incantation atteignent le niveau supérieur.

Transmis de génération en génération, le secret de l'élévation se résume ainsi :

élévation	joueurs requis	linemate	deraumère	sibur	mendiane	phiras	thystame
1-2	1	1	0	0	0	0	0
2-3	2	1	1	1	0	0	0
3-4	2	2	0	1	0	2	0
4-5	4	1	1	2	0	1	0
5-6	4	1	2	1	3	0	0
6-7	6	1	2	3	0	1	0
7-8	6	2	2	2	2	2	1

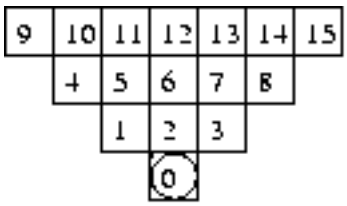
.6 Vision

Pour diverses raisons, le champ de vision des joueurs est limité. À chaque élévation, la vision augmente d'une unité de mesure en avant et d'une de chaque côté de la nouvelle rangée. Au premier niveau, l'unité de mesure est définie à 1.

Pour que le joueur ait connaissance de son entourage, le client envoie la commande voir, à cette commande le serveur répondra la chaîne de caractères comme ce qui suit.

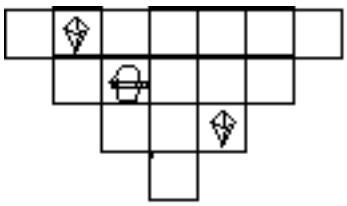
```
voir
{objet-present-case0,objet-present-case1, ..., objet-present-caseP, ... }
```

Le schéma suivant indique le principe de numérotation.



Par exemple dans le cas suivant, on obtient :

```
{,,, thystame,, nourriture,,,,, thystame,,,,,}
```



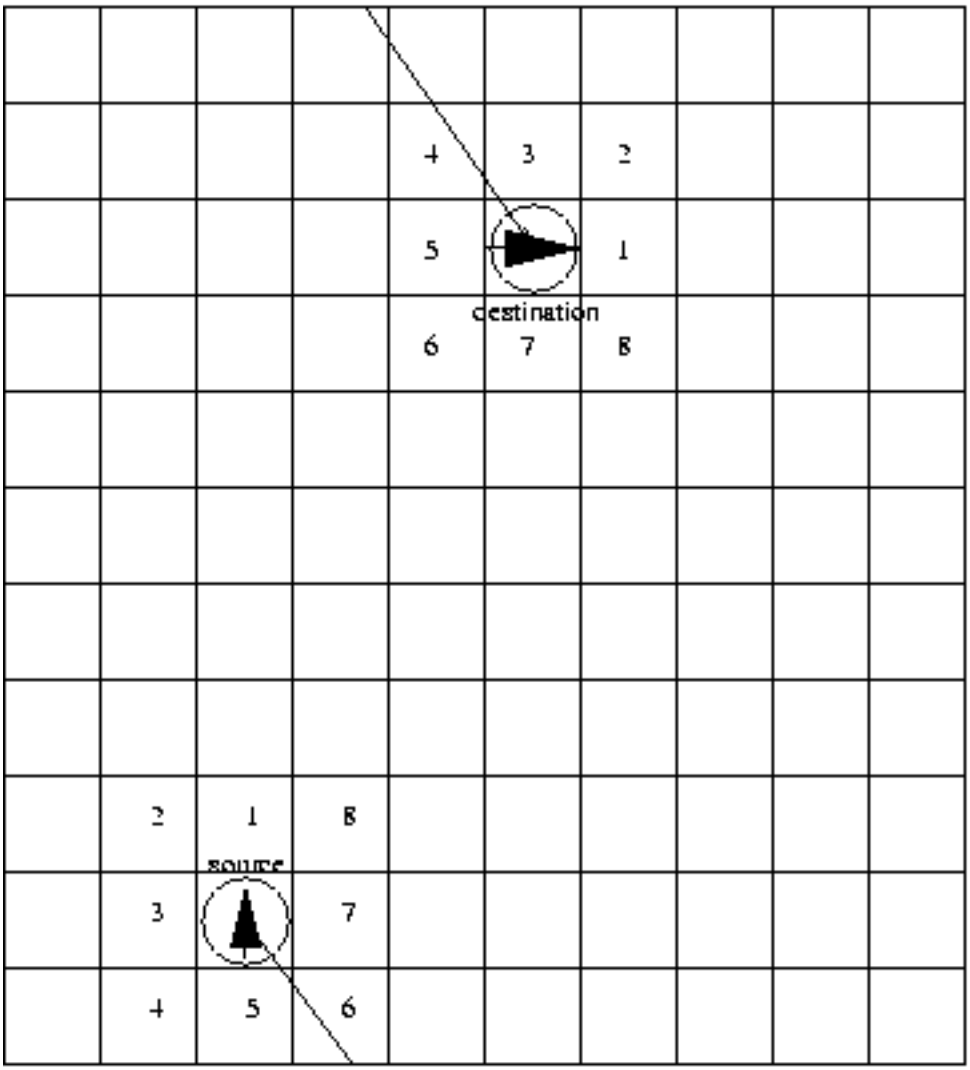
Lorsque sur une case se trouve plus d'un objet, ils sont tous indiqués et séparés par un espace. Voici un exemple pour un joueur de niveau 1 ayant deux objets sur la case 1 :

```
voir
{, joueur deraumere,,}
```

.7 Transmission du son

Le son est une onde qui se propage de manière linéaire. Tout les joueurs entendent les broadcasts sans savoir qui les émet. Ils perçoivent uniquement la direction d'où provient le son et le message émis. La direction est indiquée par le numéro de la case franchie par le son avant d'arriver dans la case du joueur. Cette numérotation est effectuée par l'attribution du "1" à la case située devant le joueur, puis à un décompte des cases entourant le joueur dans le sens trigonométrique. Le monde étant sphérique le trajet que l'on choisira pour le son sera le plus court chemin reliant l'émetteur au joueur pour lequel on le calcule.

L'exemple suivant indique le trajet du son que l'on doit choisir, ainsi que la numérotation des cases autour du joueur. Dans le cas où le broadcast est émis à partir de la même case que le joueur récepteur, il recevra le message provenant de la case 0.


4
3
2
5
1
destination
6
7
8
2
1
8
source
3
7
4
5
6

Le Travail

.1 Les Programmes

Il s'agit donc de créer deux programmes. Un serveur réalisé en C aura comme tâche de gérer le monde et ses habitants. Un client réalisé en langage libre pilotera un habitant par des ordres qu'il enverra au serveur. Le client pourra aussi être developpé sous d'autres plateformes qu'Unix.

Voici la syntaxe d'utilisation du serveur :

```
-p numero de port
-x largeur du Monde
-y hauteur du Monde
-n nom_de_eequipe_1 nom_de_equipe_2 ...
-c nombre de clients par équipe autorisés au commencement du jeu
-t delai temporel d'execution des actions.
```

Quant à la syntaxe d'utilisation du client :

```
-n nom d'equipe
-p port
-h nom de la machine, par defaut localhost
```

Le serveur s'exécute sous la forme d'un seul processus et un seul thread. Le client est autonome, après son lancement l'utilisateur n'influe plus sur son fonctionnement. Il pilote un drône (joueur) comme dans le projet coreware.

.2 Les équipes

Au commencement une équipe est composée de n joueurs. Chaque joueur est piloté par un client. Les clients ne peuvent pas communiquer ou échanger entre eux des données en dehors du jeu, de quelque façon que ce soit.

Au début le client possède 10 unité de vie, il peut donc survivre 1260 unités de temps, soit $1260 \times t$ secondes.

.3 Les commandes

Chaque joueur répond aux actions suivantes et uniquement avec la sytaxe suivante :

Action	Commande	Delai	Réponse
avance d'une case	avance	$\frac{7}{t}$	ok
tourne à droite de 90 degrés	droite	$\frac{7}{t}$	ok
tourne à gauche de 90 degrés	gauche	$\frac{7}{t}$	ok
voir	voir	$\frac{7}{t}$	case1, case2, case3 ...
inventaire	inventaire	$\frac{1}{t}$	linemate n, sibur n, ...
prend objet	prend objet	$\frac{7}{t}$	ok/ko
pose objet	pose objet	$\frac{7}{t}$	ok/ko
expulse les joueurs de cette case	expulse	$\frac{7}{t}$	ok/ko
broadcast texte	broadcast texte	$\frac{7}{t}$	ok
débute l'incantation	incantation	$\frac{300}{t}$	elevation en cours niveau actuel : K
fork un joueur	fork	$\frac{42}{t}$	ok
Nombre de slots non utilisés par l'équipe	connect_nbr	0	valeur
mort d'un joueur	-	-	mort

Toutes les commandes sont transmises via une chaîne de caractères terminée par un retour chariot.

.4 Dialogue client/serveur

Le dialogue entre le client et le serveur s'effectue via des sockets tcp. Le port utilisé sera indiqué en paramètre. Le client envoie ses requêtes sans attendre leurs réalisations, le serveur renvoie un message confirmant le bon déroulement de l'exécution des requêtes. La connection du client au serveur se déroule comme suit : le client ouvre une socket sur le port du serveur, puis le serveur et le client dialoguent comme suit :

```
<-- BIENVENUE\n
--> NOM-EQUIPE\n
<-- NUM-CLIENT\n
<-- X Y\n
```

Le NUM-CLIENT indique le nombre de clients pouvant encore être acceptés par le serveur pour l'équipe NOM-EQUIPE. Si ce nombre est supérieur ou égal à 1 un nouveau client se connecte.

Le client peut envoyer jusqu'à dix requêtes successives sans avoir de réponse du serveur. Au-delà de dix le serveur ne les prendra plus en compte. Le serveur exécute les requêtes des clients dans l'ordre de réception. Les requêtes sont bufferisées et le délai d'exécution d'une commande ne bloque que le joueur concerné. X et Y indiquent les dimensions du monde.

.5 Le Temps

Aucune attente active n'est toléré. Il ne doit pas y avoir de bloquage lorsque les clients sont stoppés, ou dans n'importe quelle phase du jeu.

Les Trantoriens ont adoptés une unité de temps internationale. L'unité de temps est la seconde. Le temps d'exécution d'une action est calculé avec la formule suivante : $action/f$ où f est une fréquence en Hertzs.

Si f = 1, 'avance' prend 7/1 secondes. On choisira par défaut, $f = 100$. f sera toujours un entier. Le référentiel de temps est le temps absolu.

.6 Gestion des objets

Seul la classe des objets est identifiable. Il est ainsi impossible de distinguer deux objets de la même classe. Par exemple, deux siburs auront la même dénomination puisqu'ils appartiennent à la même classe.

.7 Reproduction des joueurs

Un joueur peut se reproduire grâce à la commande fork. L'exécution de cette commande entraîne la production d'un oeuf. Dès la ponte le joueur l'ayant pondu peut vaquer à ses occupations en attendant qu'il éclose. Lors de l'éclosion de l'oeuf, un nouveau joueur en sort, il est orienté au hasard. Cette opération autorise la connexion d'un nouveau client. La commande connect_nbr renvoie le nombre de connexions en cours et autorisées pour cette famille.

Délai de ponte : $\frac{42}{t}$. Délai entre la ponte et l'éclosion : $\frac{600}{t}$.

.8 Inventaire

Cette commande permet de voir ce que le drône a comme objet et combien de temps il lui reste à vivre. Le seveur enverra par exemple la ligne suivante :

```
1 {nourriture 345, sibur 3, phiras 5, ..., deraumere 0}
```

.9 Le Broadcast

Pour émettre un message, le client doit envoyer au serveur la commande suivante :

```
broadcast texte
```

Le serveur, lui, enverra à tous les clients la ligne suivante :

```
message K,texte
```

avec K indiquant la case d'où provient le son.

.10 Expulsion

Le drône à la faculté d'expulser tous les drônes partageant la même unité de terrain. Il les pousse dans la direction vers laquelle il regarde. Lorsqu'un client envoie au serveur la commande expulse, tous les clients partageant cette case reçoivent la ligne suivante :

```
deplacement: K\n
```

avec K indiquant la direction de la case d'où le drône provient.

.11 Interface graphique

Le projet devra être muni d'une visualisation graphique. Celle-ci devra proposer une représentation du monde. L'interface pourra être intégrée soit au serveur, soit à l'un des clients. Ce choix est laissé à l'appréciation des développeurs, sachant qu'il est à priori plus aisé de l'inclure dans un client particulier.

L'interface doit intégrer au minimum une visualisation 2D par l'intermédiaire d'icônes, permettant ainsi une représentation du monde. Une interface 3D ou tout autre type de représentation sera un atout appréciable au projet mais gardez à l'esprit que cette interface graphique doit être fonctionnelle avant d'être agréable visuellement.

Pour développer cette interface graphique vous êtes libre d'utiliser les langages et librairies suivants :

- En C, en utilisant la librairie `SDL`.
- En C++, en utilisant la partie graphique de la librairie du `GDL`, SFML, ou Qt.

Si vous désirez utiliser un autre langage, vous devrez obtenir l'accord du professeur responsable de votre module de système Unix. Sans cet accord, obtenu par mail uniquement, toute utilisation d'une bibliothèque non autorisée pourra être considérée comme de la triche.