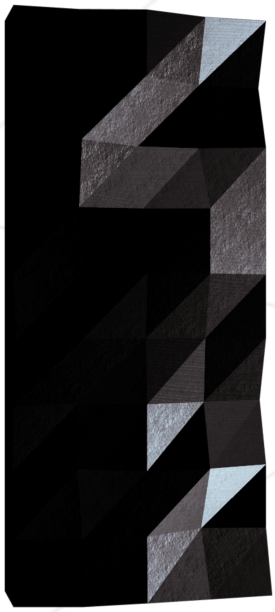


zappy

UNIX sweet project

Summary: You will implement a multiplayer game using TCP/IP

Version: 2

Contents

Chapter I

Foreword

Zappy is an automatic video game where AIs play with each others. The game speed will be setup at start and defined by a certain amount of time that define the time unit. Each action will cost some time units.

Time unit is defined by 1/t:

- `t` will be an argument of the server,
- each time unit lasts `1/t seconds`

Here are the different parts of the game:

- A server with the current map, resources, it will take care of all the timing in the game and is also the judge of the game (it will enforce the rules)
- un ou plusieurs clients qui se connectent au serveur et “pilotent” chacun un joueur. Les joueurs sont répartit en équipe.
- One or more client can connect to the server. Each client is a player. Players are part of teams.
- A graphic client that will connect to the server to show what is currently happening and let us be amazed by your work.

Chapter II

The game overview

II.1 Geography

The game is about managing an entire world and its population. That world, Trantor is geographically made of plains that have no height: no crater, no valley, no mountains. The game board represents the entire surface of that world, like a map. If a player exits on the right of the board, he will re-enter on the left. The game is played by teams. The winning team is the one that will have its 6 players reach the maximum level.

II.2 Resources

The location where we are is quite rich in resources: mining or food. Thus, all you need is to walk around to discover this fabulous food or many stones of various nature. food is called 'nourriture'

These stones have 6 distinct kinds: linemate, deraumere, sibur, mendiane, phiras and thystame. The server generates the resources. Generation must be random but must follow some rules.

II.3 Activities

The population of Trantor has two type of occupation:

- Get 'nourriture' to eat and not die of hunger.
- Look for stones, pick them up to create totems, have an elevation ritual and get to the next level.

Elevation is an important activity for Trantorians.

II.4 Individuals

The local is pacific. He is neither violent nor aggressive.

He is happy-go- lucky looking for stones and feeding on the way.

He meets without issues his people on the same location and sees as far as the eye can see.

Trantorian is immaterial, he is blurry and occupies the entire square in which hestands. It is impossible to guess his orientation when crossing him.

The 'nourriture' that the Trantorian picks up is the only resource he needs to live. One 'nourriture' unit allows him to survive 126 time units, therefore 126/t seconds.

II.5 Elevation ritual

Everyone's goal is to reach the top of the Trantorian hierarchy. The ritual that allows to increase one's physical and mental capacities must be accomplished according to a particular rite. One needs to bring together on the same field unit:

- A combination of stones
- A certain amount of players of the same level

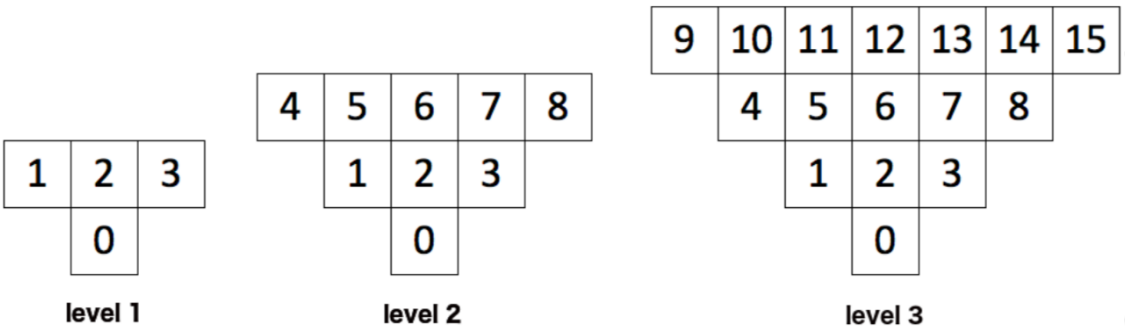
The player will start the incantation and the elevation will start. It isn't necessary that all the players be on the same team. Only their collective levels matter. All players within the incantation group will reach the higher level.

Passed on from generation to generation, the secret to the elevation goes as such:

Elevation	Prerequisite						
Level	Number of players	linemate	deraumere	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

II.6 Vision

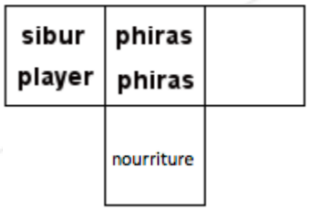
For various reasons, the vision field of the payers is limited. With each elevation their vision goes up one unit of measure ahead and one on each side of a new row. Thus we obtain, for example, for the first 3 levels and so the first two elevations the following visions (our player is on 0):



For the player to know his surroundings the client sends the command 'VOIR' and the server responds the following chain of characters (for level 1):

```
{content-square-0, content-square-1, content-square-2, content-square-3}
```

Our player doesn't even see himself, also if on a box there is more than 1 object, there are all indicated and separated by a space:



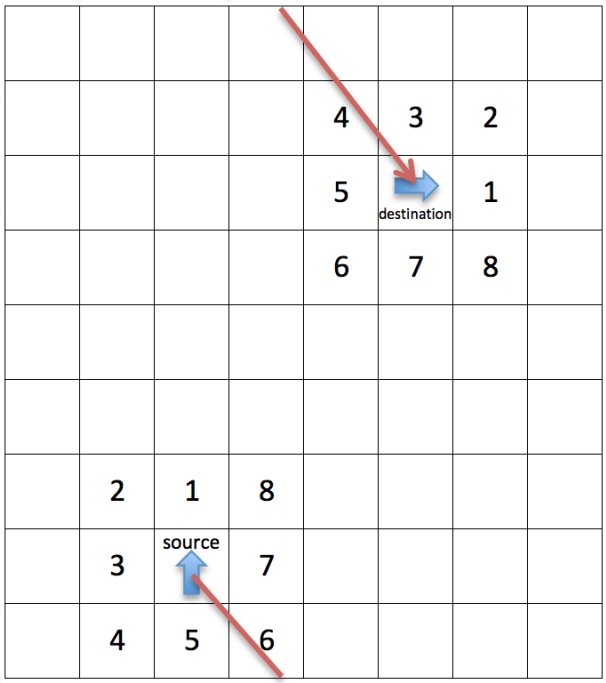

```
{nourriture, player sibur, phiras phiras, }
```

II.7 Sound Transmission

Sound is a wave that moves in a linear way.

All the players hear the broadcasts without knowing who emits them. They perceive only the direction the sound comes from and the message. The number of the square crossed by the sound before it arrives to the player indicates the direction. This numbering is done through the attribution of '1' to the square in front of the player, then a count down of the squares surrounding the player in the trigonometric direction (counter-clock wise). The world is round therefore we will choose the shortest trajectory for the sound between the transmitter to the player for which we calculate.

The following example indicates the sound trajectory that we must choose, as well as the numbers of the squares around the player. The player receives the broadcast through square 3.



In case the broadcast is emitted from the same box as the receiving player, he will get the message from square 0.

Chapter III

The project

III.1 The server

A server developped in C on the school dump will have to manager the world and its population.

```
Usage: ./server -p <port> -x <width> -y <height> -n <team> [<team>] [<team>] ... -c <nb> -t <t>
  -p port number
  -x world width
  -y world height
  -n team\_name\_1 team\_name\_2 ...
  -c number of clients authorized at the beginning of the game
  -t time unit divider (the greater t is, the faster the game will go)
```

The server runs under a single process and a single thread (which means: NO THREAD!)

III.2 The client

Client can be in any langage. It must work out of the box on the school dumps. it will control one player by giving orders to the server.

```
Usage: ./client -n <team> -p <port> [-h <hostname>]
  -n team\_name
  -p port
  -h name of the host, by default it'll be localhost
```

The client is autonomous, after its launch the user won't influence its operation. He pilot a drone (player) like in the corewar project.

III.3 Teams

At the beginning a team is made of n player and only n. Each player is controled by a client. The clients cannot communicate or exchange amongst each other data outside of the game, in any way.

At the beginning the client has 10 life units, he can therefore survive 1260 time units, ie 1260/t seconds.

III.4 The commands

Each player responds to the following actions and only with the following syntax:

Action	**Command**	**Delai**	**Response**
advance one square	`avance`	`7/t`	`ok`
turn right 90 degrees	`droite`	`7/t`	`ok`
turn left 90 degrees	`gauche`	`7/t`	`ok`
see	`voir`	`7/t`	`{case1, case2, ...}`
inventory	`inventaire`	`1/t`	`{phiras n, sibur n, ...}`
take an object	`prend <object>`	`7/t`	`ok/ko`
put down an object	`pose <object>`	`7/t`	`ok/ko`
kick the players from the square	`expulse`	`7/t`	`ok/ko`
broadcast	`broadcast <text>`	`7/t`	`ok`
begin the incantation	`incantation`	`300/t`	`elevation en cours`
			`niveau actuel : K`
fork a player	`fork`	`42/t`	`ok`
know the number of unused connections by the team	`connect_nbr`	`0/t`	value
death of a player	`-`	`-`	`mort`

All the commands are transmitted via a chain of characters that end by a newline.

III.5 Client/server communication

The communication between client and server will happen via sockets and tcp. The port used will be an argument of the programs.

The client will send its requests without waiting for their execution, the server sends back a message confirming successful execution of the requests.

The connection client to server will happen as such:

The client open a socket on the port of the server; then the server and the client communicate as such:

client's message	**server's message**
	`BIENVENUE\n`
<team-name>\n	
	<nb-client>\n
	<x> <y>\n

The `nb-client` indicates the number of clients that can still be accepted by the server for the team `team-name`.

If that number is greater than 1 a new client connects.

`The client can send successively up to 10 requests without response from the server. Beyond 10 the server will no longer take them into account.`

The server executes the requests of the client in the order they are received. The requests are buffered and the execution time of a command will only block the player concerned.

`x` and `y` indicates the dimensions of the world.

III.6 Time

No active wait time will be tolerated. There cannot be any blocking when the clients are stopped, or in any phase of the game

The Trantorians have adopted an international time unit. Time unit is a second. If t=1 “forward” takes 7 seconds. We choose by default, t=100. t is a integer.
The time reference is absolute time.

III.7 Objects’ names

Only the classification of an object can be identifiable. It is therefore impossible to distinguish two objects of the same class; ie two siburs will have the same denomination since they belong to the same class.

III.8 Reproduction des joueurs

A player can reproduce with the fork command. This execution of this command results in the production of an egg. Once the egg is laid, the player that laid the egg cango around until it hatches. When the egg hatches a new player pops. He is oriented randomly. This operation authorizes the connection of a new client. The connect_nbr command returns the number of authorized and unauthorized connections for this team.
Time to lay an egg: 42/t
Time for the egg to hatch 600/t

III.9 Inventory

This command allows to see what objects the player has and how long it has to live. The server will send back for example the following line :

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

III.10 Broadcasting

To send a message the client must send to the server the following command:

```
broadcast <texte>
```

The server will send to all its client this line:

```
message <K>,<texte>
```

With K indicating the square where the sound comes from.

III.11 Kick

A player can expulse all the players sharing the same square. It pushes them in the direction he is looking at. When a client send to the server the kick command, all the clients in this square receive the following line:

```
deplacement <K>\n
```

With K indicating the direction of the square where the player comes from

III.12 Graphic interface

The project will have to have a graphic visualization client. That client will propose a real-time representation of the world as it is on the server.

The interface will integrate at minima a 2D visualization through icons allowing a representation of the world. A 3D interface or any other type of representation will be an appreciated bonus for this project. You also need to include the visualization of the sounds.

It will be developped in C, in PHP, in Perl, in Python, etc. (as long as it works on the dumps) and will communicate within the network with the server to retrieve the content of the map, teams, inventories, etc. ie everything needed to see what's going on in the game.

LThe goal of the graphic client is to be able to follow games, so to know how each player each team progresses, who is ahead, who won etc. You must design the user interface with that in mind.

Chapter IV

turn-in

IV.1 turn-in

This project will be corrected by humans only. You're allowed to organise and name your files as you see fit, but you must follow the following rules.

The binary for the server should be `serveur`, the one for the client should be `client`, and the graphic part should be `gfx`

A Makefile must compile both binaries and must contain the following rules: client, serveur, gfx, all, clean, fclean, re. It must recompile and re-link the programs only if necessary.

You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).

Within the mandatory part, you are allowed to use every libc functions as well as every syscalls.

WARNING: You are however not allowed to use threads (for the server) and you're not allowed either to use non-blocking sockets (meaning no fcntl(s, O_NONBLOCK))

You are allowed to use other functions to complete the bonus part as long as their use is justified during your defence. Be smart! You can ask your questions on the forum, on slack...