{EPITECH}

ZAPPY

A TRIBUTE TO ZAPHOD BEEBLEBROX

ZAPPY

Preliminaries

binary name: zappy_server, zappy_gui, zappy_ai
language: C (server), C++ (gui), free (ai)

Your Makefile should contain a **zappy_server**, a **zappy_gui** and a **zappy_ai** rules to compile the eponymous binaries.

You can use the whole C standard library.

The goal of this project is to create a network game where several teams confront each other on a tile map containing resources.
The winning team is the first one where at least 6 players reach the maximum elevation.
The following pages describe all the details and constraints.

Environment

1. Geography

The game is about managing a world and its inhabitants.
This world, **Trantor** is geographically made of zero-relief planes: no craters, no valleys, no mountains.
The game board represents the entirety of this world's surface, like a world map.

If a player exits by the right of the board, they will come back through the left. Up->down, and so on…

2. Resources

The environment is rather rich in resources (mineral as well as dietary).
Therefore, by walking around this planet, the players can find succulent food and a variety of natural stones.
These stones have six distinct categories, as follows:

- ✓ linemate
- ✓ deraumere
- ✓ sibur
- ✓ mendiane
- ✓ phiras
- ✓ thystame

The server spawns resources upon starting and every 20 time units.
It follows this set of rules:

- ✓ on Trantor you must find at least one of each ressource and food on the floor.
- ✓ resources should be evenly spread across the map.
- ✓ the ressource quantity can be found with the following formula: `map_width * map_height * density`

resource	density
food	0.5
linemate	0.3
deraumere	0.15
sibur	0.1
mendiane	0.1
phiras	0.08
thystame	0.05

for instance on a 10 by 10 world there is 50 food and 5 thystame.

3. Activities

Trantor's inhabitants take care of two things:

- ✓ feeding themselves
- ✓ looking for, and collecting, stones

These objectives determine elevation, which is an important activity for the Trantorians.

4. Individuals

The inhabitants are bodiless, blurry and takes up the entire tile they are in.
They are pacifists. They are neither violent nor aggressive.
They eat and meander happily in search of stones.
They easily run into their fellow creatures on the same tile.

Trantorians can see as far as their visual capacities allow.
It is impossible to distinguish their direction when we run into them.
The food the Trantorians collect is the only resource they need to survive.
One unit of food allows them to live for 126 time units.

5. The Elevation Ritual

Everyone's goal is to rise up in the Trantorian hierarchy.
This ritual, which enhances physical and mental capacities must be done according to a particular rite: they must gather the following on the same unit of terrain:

- ✓ At least a certain number of each stones
- ✓ At least a certain number of players with the same level

The elevation begins as soon as a player initiates the incantation.
It is not necessary for the players to be on the same team; they only need to be of the same level.
Every player in a group doing an incantation attain the higher level.
Passed down from generation to generation, the elevation secret comes down to this:

elevation	nb 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

The verification of the prerequisites for the incantation is done at the beginning and at the end of the action.

If the conditions are not met at one of the verifications, the elevation fails.

Each player participating is an elevation is frozen during the ritual. They can't do any other action in the meantime.

Once the ritual succeeds, the stones are removed from the terrain.

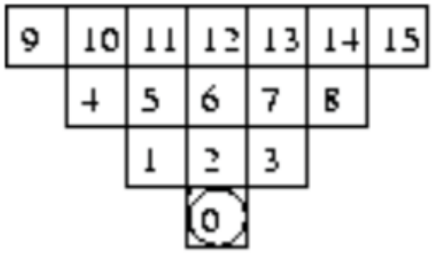
6. Vision

For various reasons, the players' field of vision is limited.
With each elevation, the vision increases by one unit in front, and one on each side of the new line.
At the first level, the unit is defined as 1.

In order for a player to recognize their team, the client sends the **look** command. The server will respond with the character string, as follows.

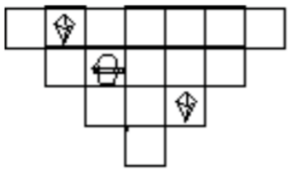
```
look
[player, object-on-tile1, ..., object-on-tileP,...]
```

The following diagram explains the numbering concept:



For example, in the next instance, the following is obtained:

```
[player,,,thystame,,food,,,,,thystame,,,,,,]
```



If more than one object is located on a tile, they will all be indicated and separated by a space.
Here's an example for a level-1 player having two objects in tile 1:

```
look
[player, player deraumere,,]
```

Beware, the tile separator is a comma followed or not by a space.

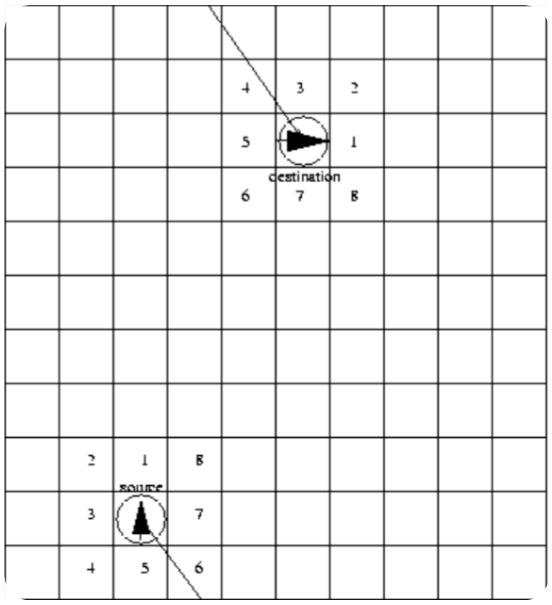
7. Sound transmission

Sound is a wave that spreads out linearly (at least on Trantor) by broadcasting.
All the players can hear the broadcasts without knowing who is playing them.
They can only perceive the direction the sound is coming from and its subsequent message.

The direction is indicated by the number of the tile affected by the sound, before arriving in the player's tile.
This numbering is done through attributing 1 to the tile that is located in front of the player, then through deducting the tiles that trigonomically encircle the player.
In the event that the broadcast is emitted from the same player receptor tile, they will receive the message coming from the 0 tile.

As the world is spherical, several trajectories are possible for the sound between the emitter and the player.
The shortest path will always be chosen.

Here is an example showing the shortest sound trajectory and the tile numbering around the player.



Programs

1. Binaries

You must create three binaries.
A server, written in C, that generates the inhabitants' world.
A graphical client, written in C++, that can be used to watch what happens in the world.
A client, with no language constraint, that drives an inhabitant through orders sent to the server.

```
Terminal
~/B-YEP-400> ./zappy_server -help
USAGE: ./zappy_server -p port -x width -y height -n name1 name2 ... -c clientsNb -f freq
```

option	description
-p port	port number
-x width	width of the world
-y height	height of the world
-n name1 name2 ...	name of the team
-c clientsNb	number of authorized clients per team
-f freq	reciprocal of time unit for execution of actions

```
Terminal
~/B-YEP-400> ./zappy_gui -help
USAGE: ./zappy_gui -p port -h machine
```

option	description
-p port	port number
-h machine	hostname of the server

```
Terminal
~/B-YEP-400> ./zappy_ai -help
USAGE: ./zappy_ai -p port -n name -h machine
```

option	description
-p port	port number
-n name	name of the team
-h machine	name of the machine; localhost by default

The server is executed as one single process and one single thread.
It must use `select` to handle socket multiplexing; the select must unlock only if something hap-

pens on a socket or if an event is ready to be executed. (See strace)

The team name `GRAPHIC` is reserved for the GUI to authenticate itself as such to the server.

The AI client is autonomous.
After it's launched the user has no further influence on how it functions.
It drives a drone (the player) (kind of similar to what a champion does in the Corewar project, remember?).

The graphical client is autonomous.

Consider an optimized approach where the server pushes changes on tiles to the GUI. If there is a change on a tile that is not caused by a player (e.g., resource respawn), only that tile is pushed to the GUI. For player actions like picking up or dropping an item, separate events handle the updates, avoiding redundant notifications to the GUI.

The **mandatory** protocol for the graphical client is given alongside this subject.
It is used by the reference server (so you can test your GUI with it aswell).

2. Teams

In the beginning of the game, a team has *n* slot(s) available (represented on the board by an egg waiting for a client to connect).
Each player is driven by a client.
The clients cannot communicate or exchange data outside of the game (no matter what it is, it should be passed via the server).
In the beginning, the player has 10 life units, which means they can survive for 1260 time units, or 1260 / f seconds.

3. Commands

Each player responds to the following actions and **only** to these ones, with the following syntax:

action	command	time limit	response
move up one tile	Forward	7/f	ok
turn 90° right	Right	7/f	ok
turn 90° left	Left	7/f	ok
look around	Look	7/f	[tile1, tile2,…]
inventory	Inventory	1/f	[linemate *n*, sibur *n*,…]
broadcast text	Broadcast text	7/f	ok
number of team unused slots	Connect_nbr	-	value
fork a player	Fork	42/f	ok
eject players from this tile	Eject	7/f	ok/ko
death of a player	-	-	dead
take object	Take object	7/f	ok/ko
set object down	Set object	7/f	ok/ko
start incantation	Incantation	300/f	Elevation underway Current level: k /ko

All commands are transmitted through a character string that ends with a new line.

In case of a bad/unknown command, the server must answer “ko”.

The protocol defined at the end of this document must be respected!

4. client ai/server communication

The communication between the AI client and the server is carried out via **tcp** sockets.
The port that is used must be indicated in parameters.

The client sends its requests, without waiting for them to be done.
The server sends back a message confirming the correct execution of the requests.

The client's connection to the server happens as follows:

1. the client opens a socket on the server's port,
2. the server and the client communicate the following way:

```
<--WELCOME\n
-->TEAM-NAME\n
<--CLIENT-NUM\n
<-- X Y\n
```

X and **Y** indicate the world's dimensions.
CLIENT-NUM indicates the number of slots available on the server for the **TEAM-NAME** team.
If this number is greater than or equal to 1, a new client can connect.

The client can send up to 10 requests in a row without any response from the server.
Over 10, the server will no longer take them into account.

The server executes the client's requests in the order they were received.
The requests are buffered and a command's execution time only blocks the player in question.

5. Time

Active waiting is not tolerated: there should not be any blocking when the clients are stopped, nor in any phase of the game.

Trantorians have adopted an international time unit.
The time unit is seconds.
An action's execution time is calculated with the following formula: action / f,
where **f** is an integer representing the reciprocal (multiplicative inverse) of time unit.

For instance, if f=1, "forward" takes 7 / 1 = 7 seconds.
By default **f=100**.

6. Object Management

Only the object class is identifiable.
Therefore, it is impossible to differentiate between two objects of the same class. For example: two **siburs** always have the same denomination because they belong to the same class.

7. Player Reproduction

A player can reproduce thanks to the **fork** command.
The execution of this command leads to the production of an egg.
Once it's layed, the player who has layed it can take care of their business.
Once the egg is layed, a new slot is added to the team.

This operation authorizes a new client to be connected.

The **connect_nbr** command sends back the number of connections that are underway and authorized for this family.

8. Egg hatching

When a client connects to a free slot of their team, an available egg from the team is selected randomly.
The selected egg then hatches and the newly spawned player starts with a random direction.

9. Inventory

This command allows you to see what object the drone has and how much time it has left to live.
The server will send a line similar to the following one:

```
[food 345, sibur 3, phiras 5, ...,deraumere 0]\n
```

10. Broadcast

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

```
Broadcast text\n
```

The server will then send the following line to all of its clients:

```
message K, text\n
```

where **K** is the tile indicating the direction the sound is coming from.

11. Ejection

A player can eject all other players from a shared unit of terrain by pushing them in the direction it is looking.
When a client send the **eject** command to the server, all of the clients that are sharing the tile will receive the following line:

```
eject: K\n
```

where **K** is the direction of the tile where the pushed player is coming from.

When a player ejects all other beings from the unit of terrain, it also destroy any eggs layed on that unit.

12. Graphical User Interface

The project must include a graphic visualization.
It should be a representation of the world.

The interface must integrate at least a 2D visualization via intermediary icons, which must also allow a representation of the world to be seen.
A 3D interface, or any other type of representation, will be a greatly appreciated asset. Still, remember that the graphical interface must be functional before being visually appealing.

In order to develop this interface, you must use C++ and remember that you'll need to handle incoming and outgoing data the right way with buffering as in the server.

The GUI authenticates itself as such to the server by sending `GRAPHIC` when the server waits for the team's name.

For 2D renders, we greatly encourage you to use the SFML because you should already know how it works.

v 3.2.2