ZAPPY: A TRIBUTE TO ZAPHOD BEEBLEBROX

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

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...

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 resource and food on the floor.
- resources should be evenly spread across the map.
- the resource 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.

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.

Individuals

The inhabitants are bodiless, blurry and take 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.

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 stone
- 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 attains 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 in 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.

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.

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

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 - + x
~/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
```

You must respect this command line. The order of options is not necessarily the one above. Be careful, the options and their arguments have to be separated by space characters.

When launching the server, a map will be randomly generated (you can add an option for a random seed for ease of debugging). Then, the server waits for client connections.

Clients

```
 Terminal - + x
~/B-YEP-400> ./zappy_ai -p port -n name -h machine
```

You must respect this command line. The order of options is not necessarily the one above. Be careful, the options and their arguments have to be separated by space characters.

In the example above, the client connects to the machine named "machine" on port "port" and as team "name."

To facilitate testing, a graphical client should be available. Here is a sample command line:

```
 Terminal - + x
~/B-YEP-400> ./zappy_gui -p port -h machine
```

You must respect this command line. The order of options is not necessarily the one above. Be careful, the options and their arguments have to be separated by space characters. The server and the clients communicate through TCP sockets.

Command Line Protocol

The server communicates with the clients through an ASCII exchange by using the TCP sockets. The server will create and use one socket per client.

To simplify the parsing, all the commands must end with a "\n" character. The server may send the following messages:

- msz width height – sent upon connection.
- `bct x y q1 q2 q3 q4 q5 q6 q7` – quantity per resource unit on tile (x,y)
- `mct` – sent upon client request for the whole map
- `tna name` – sent upon connection and to request the team name
- `pnw n x y o L N` – new player connection, sent to all clients
- `ppo n x y o` – player position, sent upon connection
- `plv n L` – player level, sent upon request
- `pin n x y q1 q2 q3 q4 q5 q6 q7` – player inventory, sent upon request
- `pex n` – player expulsion, sent to all clients
- `pbc n text` – broadcast text
- `pic x y L n1 n2 ...` – elevation, sent to all clients
- `pie x y result` – end of elevation, sent to all clients

- `pfk n` – player lays an egg, sent to all clients
- `pdr n i` – player drops an object, sent to all clients
- `pgt n i` – player gathers an object, sent to all clients
- `pdi n` – player death, sent to all clients
- `enw e n x y` – player's egg laid, sent to all clients
- `eht e` – egg hatches, sent to all clients
- `ebo e` – player on egg, sent to all clients
- `edi e` – player's egg death, sent to all clients
- `sgt T` – time request
- `seg N` – game end, sent to all clients
- `smg text` – server message
- `suc` – unknown command
- `sbp` – bad parameters

When the server receives commands from a client, it sends back the proper response from the list above.