B4 - Year-End Project

B-YEP-410

zappy

a tribute to Zaphod Beeblebrox

2.87

zappy

binary name: zappy_server, zappy_ai, zappy_gui
language: C / free
compilation: via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

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 of the standard C library.

The goal of this project is to create a network game.
Several teams confront on a tiles map containing resources.
The winning team is the one with 6 players who reached maximum elevation.
The following pages describe all the details and constraints.

Environment

1. Geography

The game consists of managing a world and its inhabitants.
This world, **Trantor** is geographically made up 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, he/she will come back through the left.

2. Resources

The environment is rather rich in resources (as many mineral 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 spawn resources at start and every for 20 units of time.
it follow this set of rules:
* on Trantor you must find at least one of every ressources and food on the floor.
* if possible the ressources can not be on the same spot.
* 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 two objectives determine elevation, which is an important activity for the Trantorians.

4. Individuals

The inhabitants are bodiless, blurry and takes up the entire tile he/she is 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 terrain.

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

5. The Elevation Ritual

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

- a combination of stones
- 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; only their level is important.
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 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

Incantation checks are done at the begening and at the end of the action.

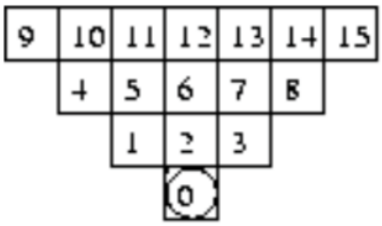
6. Vision

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 his/her team, the client sends the **look** command. The server will respond with the character string, as follows.

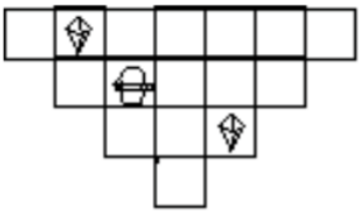
```
look
[player, object-on-tile1, ..., object-on-tileP,...]
```

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 coma followed or not by a space.

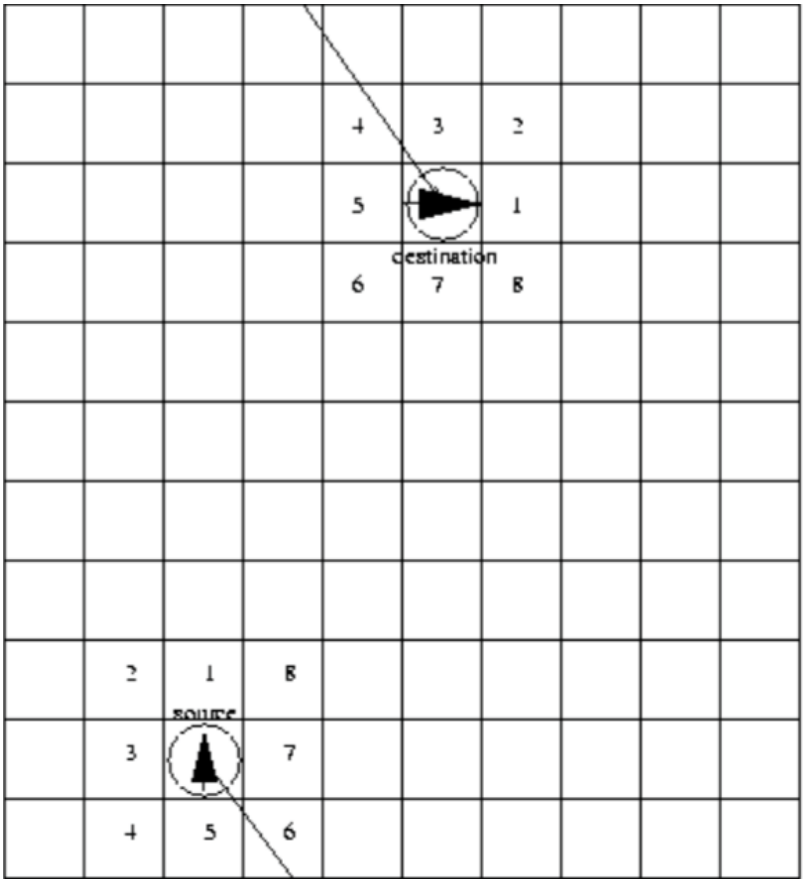
7. Sound transmission

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, he/she 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 shorteste sound trajectory and the tile numbering around the player.



PROGRAMS

1. BINARIES

You must create three binaries.
A server, created in C, that generates the inhabitants' world.
A graphical client, created in free language, that display the world.
A client, created in free language, that drives an inhabitant through orders sent to the server.

Terminal

```
~/B-YEP-410> ./zappy_server -help
USAGE: ./zappy_server -p port -x width -y height -n name1 name2 ... -c clientsNb
-f freq
      port       is the port number
      width      is the width of the world
      height     is the height of the world
      nameX      is the name of the team X
      clientsNb  is the number of authorized clients per team
      freq       is the reciprocal of time unit for execution of actions
```

Terminal

```
~/B-YEP-410> ./zappy_gui -help
USAGE: ./zappy_ai -p port -h machine
      port     is the port number
      machine  is the name of the machine; localhost by default
```

Terminal

```
~/B-YEP-410> ./zappy_ai -help
USAGE: ./zappy_ai -p port -n name -h machine
      port     is the port number
      name     is the name of the team
      machine  is the name of the machine; localhost by default
```

The server is executed in the form of one, single process and one, single thread.

It must use `select` to handle seocket multiplexing; the select must unlock only if somthing happen on a socket or if an event is ready to be executed. (See strace)
The ai client is autonomous.
After it's launched the user has no more influence on how it functions.
It drives a drone (the player), just like in the Corewar project.
The graphical client is autonomous.

There is no mandatory protocole for the graphical client!
The one used by the reference server is given with this subject, as a proposal only so (that you can test you client with the reference client).

2. Teams

In the beginning, a team is made up of *n* players.
Each player is driven by a client.
The clients cannot communicate or exchange data outside of the game (no matter what it is).
In the beginning, the client has 10 life units, which means he/she can survive for 1260 time units, or $\frac{1260}{f}$ 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 dialogue between the ai client and the server is carried out via **tcp** sockets.
The port that is used must be indicated in paramater.

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 clients that can still be accepted by 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 blockage 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: $\frac{action}{f}$,
where **f** is an integer representing the reciprocal (multiplicative inverse) of time unit.

For instance, if f=1, "forward" takes $\frac{7}{1} = 7$ 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 be reproduced 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 his/her business and wait for it to hatch.
Once the egg has hatched, a new player comes out, and is given a random direction.

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.

Egg-laying time:	42/f
Time between egg-laying and hatching:	600/f

8. 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
```

9. 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 where the sound is coming from.

10. Ejection

The drone can eject all of the drones 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 drone is coming from.

11. Graphical User Interface

The project must be equipped with 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 orer to develop this interface, you are free to use whatever language you want, but remember that you'll need to handle incoming and outgoing data the right way with buffering same as in the server.