Technical guide

Connecting to the game for the first time

Basics

The interaction with the game is based on TCP/IP and a text protocol. We've configured a test game before finals. You can find your password in your profile after logging in at `https://contest.pizza`

`telnet` connection

To test basic commands you can connect to the game server using `telnet` (or `netcat`) and issue commands manually (i.e. without any software).

In annonuncements on `https://tr.contest.pizza` you can find an information about the port that the test game is listening on.

In the command line interface (on Windows: `cmd.exe`) enter
`telnet tr.contest.pizza <port number>`, for example:
`telnet tr.contest.pizza 10000`.

You should be able to see a welcome message: `LOGIN`. Respond with your login, confirm and proceed with entering your password when asked.

If you got `OK` in a response, it means that it's now possible to send commands to the game. Specific commands and their usage can be found in the task description. Here is how the communication can look like (`<` is always before commands sent to server, while `>` defines its responses):

```
> LOGIN
< test
> PASS
< pass
> OK
< TURNS_LEFT
> OK
> 12
```

Connecting using Python

Here is a code fragment that (after logging in to the server) requests the number of turns left and prints it.

```
import socket

HOST = 'test.natodia.net'
PORT = 10000
USER = 'test'
PASS = 'pass'

def command(f, cmd):
    f.write(cmd + "\n")
    f.flush()
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST,PORT))

f = s.makefile()
f.readline() # read ’LOGIN’
command(f, USER)
f.readline() # read ’PASS’
command(f, PASS)
f.readline() # read ’OK’

command(f, "TURNS_LEFT")
f.readline() # read ’OK’
print f.readline().strip()
```

The download link for the code can be found on `https://contest.pizza/technical`.

A few facts about the protocol

- The last character of each command has to be (`\n`, code 10).
- After each command, in the first line, there is `OK` or `ERR <code number> <message>`.
- In most games there is a `WAIT` command that works in a tricky way: immediately after receiving it, the server sends a response `OK` and just before the beginning of the next turn it sends *additional* message `OK`. The intention here is that this command should be used after running all commands in a given turn to make a synchronization with the game server easier.

Additional information

More information (e.g. links to codes that connect to the server over TCP/IP in C++) can be found on `https://contest.pizza/technical`. You can send your questions to `pizza@natodia.net`