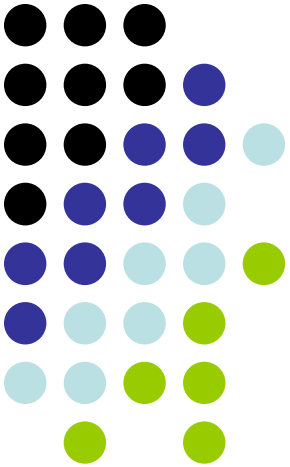

Videogame protocol

DRAFT 1.3

Giorgio Metta, Jan 2009

Compiling the game_server

- Available from
  $YARP_ROOT/example/game/game_server
- *CMakeList.txt* available, just run *CMake* on that directory, generate the project files and compile it
- You should obtain an executable called *game_server*
- *Game_server* registers a /game port on TCP port 8080 (for historical reasons)

Connection

- The server program (*game_server*) writes a welcome message upon startup
- Responses of the server to client messages are undefined up to this point -- in other words, the client should not send messages before the server is up and running
- After the above message from the server, the Login phase can begin

Login

- Client just connects to the /game port:
  - e.g. *yarp connect /myport /game*
- The next command is taken by the server as an indication to stay connected, thus it stores a user with "*/myport*" username
- The server says:
  - *Completing login…* and prints details of the connection
- The response of the server to any other message in this phase is undefined.
- After the welcome message, the Gameplay phase begins.
- CURRENT server implementation ignores password -- any username will do.

Gameplay

- Client must send Bottle messages made of strings with commands like:
  - "go" instruction,
  - "fire" instruction,
  - "look" instruction
  - "say" instruction

Go

- GO instructions - can send any one of:
  - go left
  - go right
  - go up
  - go down

- server responds with:
  - ack [move] “move requested”
- and will try to implement the move on the next game simulation step

Fire (will not really be used)

- FIRE instructions - can send any one of:
  - fire left
  - fire right
  - fire up
  - fire down
- server responds with:
  - @fire requested

Look

- LOOK instruction:
  - look

```
(look
  (map
    ":::::::::::::::::::::::::"
    ":                       :"
    ":                       :"
    ": #   #####       ####:"
    ": #       #            :"
    ":###  #   ####  ######:"
    ":         #Q      #   :"
    ":###  #####       #   :"
    ": #                #   :"
    ": #   ########    ####:"
    ":     #             #   :"
    ":                       :"
    ":::::::::::::::::::::::::")
  (players
    (carlos (location 17 15) (life 6))
))
```

Meaning of the map

- The map lines give a summary of what is around the player, using a set of characters with the following meanings:

  : -- edge of map
  # -- immovable obstacle
  Q -- the player, when alive
  0 -- other players when alive
  % -- the player, when dead
  * -- other players, when dead
  = -- traces the path where someone fired; visible for a short time

- The see lines give the name of the players visible, and how many lives they have left. The first player listed is always the player who issued the look command.

Say

- SAY instruction:
- say <text>
  - The <text> is broadcast to all players connected to the game.
  - The message they receive from the server will be of the form "broadcast <username> <text>". Your client must be prepared to receive this message at any time.

Command not understood

- Port command not understood

Disconnection

- Just disconnect the port
  - e.g. *yarp disconnect /myport /game*

Steps to run the server under Ubuntu

- Compile the server
  $YARP_ROOT/example/game/game-server/
- sudo cmake .
- sudo make
- Run the yarp name server
- yarp server
- Run the game server
- ./game_server
- Connect to the game server and have fun!

Important

- Presentazione esercitazione gioco: 31 marzo 2009 (se non la presentano scade l'esame orale).
- L'esame orale sara' a gennaio/febbraio (data da concordare).