The Labyrinth of Dooom Protocol Specification

March 11, 2010

This document describes the protocol used by the The Labyrinth of Dooom game for communication between the client and the server.

1 Notation

Within this document, the phrases MUST, MUST NOT, SHOULD, SHOULD NOT and MAY have a specific meaning. The convention used is the one described in RFC 2119. See:

`http://www.ietf.org/rfc/rfc2119.txt`

2 Commands

The protocol is made up of a series of commands. These comprise the name of the command (a human readable string) and then optionally a space and any arguements to the command, then a new line character. There are two types of command, client commands and server commands. A client command MUST only be sent by a client and a server command MUST only be sent by a server. Most client commands have a response. If a command has a response, it MUST recieve exactly one of these possible messages as the next message sent from the server. Server commands are used in two different ways, they are either responses or are informational (and thus do not require a response from the client). Responses MUST only be sent after the server processes a client command that requires such a response. Informational message MAY be sent at any time, regardless of which player's turn it is.

2.1 Client Messages

2.1.1 HELLO

Command: `HELLO <string:username>`
Type: Client
Responses: `HELLO <string:username>`

This command sets the name of the player. The client MAY send this command immediately after connecting to the server. If this message is not recieved the server SHOULD assign a unique default name to the client. The client MUST NOT send this more than once or at any other time.

2.1.2 LOOK

Command: `LOOK`
Type: Client
Responses: LOOKREPLY
The client sends this command to request a look return message from the server, indicating his immediate surroundings. The client MAY send these at any time after it has recieved it's first STARTTURN or CHANGE, even if it is not currently the player's turn. The server MUST reply with a look reply.

2.1.3 MOVE

Command: `MOVE <character:direction>`
Type: Client
Responses: SUCCEED, FAIL
The client sends this command to move one square in the indicated direction. The direction MUST be either N, S, E or W. The client MUST only send this message during the player's turn.

2.1.4 ATTACK

Command: `ATTACK <character:direction>`
Type: Client
Responses: SUCCEED, FAIL
The client sends this command to attack another player in an adjacent square in the indicated direction. The direction MUST be either N, S, E or W. The client MUST only send this message during the player's turn. The server will return FAIL if the attack fails to cause any damage or if the attack is directed against something other than a player.

2.1.5 PICKUP

Command: `PICKUP`
Type: Client
Responses: SUCCEED, FAIL
The client sends this command to pick up the item in it's current location. The client MUST only send this message during the player's turn.

2.1.6 SHOUT

Command: `SHOUT`
Type: Client
Responses: None
Sends a message to be passed to other clients. The server MAY choose a subset of clients (for example, only those who are nearby) to recieve the message. The server message type MESSAGE will be used to send the message to other clients. The client MAY send these at any time after it hasrecieved it's first STARTTURN or CHANGE, even if it is not currently the player's turn.

2.1.7 ENDTURN

Command: `ENDTURN`
Type: Client
Responses: `ENDTURN`
The client can send this command to end their turn.

2.2 Server Messages

2.2.1 HELLO

Command: `HELLO <string:username>`
Type: Server
Usage: Response
Returned as a response to a client HELLO.

2.2.2 GOAL

Command: `GOAL <number:n>`
Type: Server
Usage: Informational
The server SHOULD define the amount of gold required to win the game. If this message is sent it MUST only be sent once and MUST be the first message to be sent to the client. The argument MUST NOT be negative.

2.2.3 WIN

Command: `WIN`
Type: Server
Usage: Informational
The server MUST send this to the winning client.

2.2.4 LOSE

Command: `LOSE`
Type: Server
Usage: Informational
The server MUST send this to all losing clients (either at the end of the game or if they are killed in combat). After recieving a LOSE message, a client MAY disconnect, alternatively they may remain in the game but they MUST only send SHOUT commands and MUST only recieve MESSAGE and LOSE messages.

2.2.5 CHANGE

Command: `CHANGE`
Type: Server
Usage: Informational
When a player changes the state of the game (moves, picks up an object, attacks, etc.) the server SHOULD inform all other players who may be aware (via LOOK) of this change. The server MAY choose not to send a CHANGE message to any player who (via LOOK) can not see the change.

2.2.6 STARTTURN

Command: `STARTTURN`
Type: Server
Usage: Informational
The server MUST send this message to a client at the start of the player's turn.

2.2.7 ENDTURN

Command: `ENDTURN`
Type: Server
Usage: Informational
The server MUST send this message to a client at the end of the player's turn.

2.2.8 HITMOD

Command: `HITMOD <number:n>`
Type: Server
Usage: Informational
The server MUST send this message to the client if his hit points change. The argument indicates the number to add to the current hitpoints. (This can be negative.) Changes to hit points are normally caused by picking up health potions or being attacked by another player.

2.2.9 TREASUREMOD

Command: `TREASUREMOD <number:n>`
Type: Server
Usage: Informational
The server MUST send this message to the client if the amount of treasure they have changes. The argument indicates the number to add to the treasure score. (This can be negative.)

2.2.10 MESSAGE

Command: `MESSAGE <string:content>`
Type: Server
Usage: Informational
The arguement given for this message is a text string that a client SHOULD pass to the user. The server MAY use these messages to pass human readable, game related informations, for example announcing that as user has fulfilled the conditions required to win the game.

2.2.11 SUCCEED

Command: `SUCCEED`
Type: Server
Usage: Response
Returned as a response if a client action suceeds.

2.2.12 FAIL

Command: `FAIL <string:comment>`
Type: Server
Usage: Response
Returned as a response if a client action fails. The arguement MAY be used to provide a comment or additional information.

2.2.13 LOOKREPLY

Command: `LOOKREPLY`
Type: Server
Usage: Response
After a LOOKREPLY, a number of lines, indicating the layout of the nearby labyrinth, are transmitted. The symbols used are defined as following:

- `X` Indicates a grid location cannot be seen
- `#` Indicates a wall
- `.` Indicates a space

- `E` Indicates an exit tile
- `G` Indicates a space with one or more bits of treasure
- `H` Indicates a space with a health pickup
- `S` Indicates a space with a sword pickup
- `A` Indicates a space with an armour pickup
- `L` Indicates a space with a lantern pickup
- `P` Indicates a space with a player

The current player is assumed to be at the centre of the reply but is not included in the look reply. The look reply MUST be as small as possible (i.e. only include four 'X' characters if the player doesn't have a latern and twelve 'X' characters if the player does). This is an example of a look reply:

```
LOOKREPLY
X###X
#.###
.GG..
###.#
X##.X
```

In this example, the player is located at the same position as the second 'G'. This look reply consists out of five lines, each terminated by a new line character.

The server MAY provide additional detail about items and other players using RENDERHINT messages. This MUST be sent immediately after the LOOKREPLY it refers to.

2.2.14 RENDERHINT

Command: `RENDERHINT <number:hints>`
Type: Server
Usage: Informational
This message MUST only be sent after a LOOKREPLY and MUST relate to the LOOKREPLY it follows. Hints are intended to allow better graphical clients to be developed. Thus client software MAY choose to disregard them. The argument to the message gives the number of hints to be transmitted. This MUST be a positive number greater than 1. Each hint in then sent on a separate line, using the following format:

```
<number:relative-x> <number:relative-y> <string:hint>
```

Where relative-x and relative-y give the location of the square in the LOOKREPLY to which the hint refers (the centre square is taken to be location (0,0)). The string describes the square or the object in the square. It's format is implementation specific.

3 Example

This gives an example of communication between a client and a server using the protocol described above.

Client	Server	Comment
HELLO Dave	HELLO Dave	
	GOAL 5	The goal of the game is to collect 5 lots of treasure.
	STARTTURN	
LOOK	LOOKREPLY X##GX G##.# .GG.. ###G# X..AX	The square with the player on also contains treasure.
MOVE N	FAIL Wall	The player can't move because there is a wall in the way.
PICKUP	SUCCEED	
	TREASUREMOD 1	The result of picking up the treasure.
LOOK	LOOKREPLY X##GX G##.# .G... ###G# X..AX	
ENDTURN		The player chooses not to do anything else.
	CHANGE	Another player's actions have changed something visible.
LOOK	LOOKREPLY X##GX G##.# .G..P ###G# X..AX	Note that another player is visible.
	MESSAGE Oi!	Message from another player.

Client	Server	Comment
	CHANGE	
LOOK	LOOKREPLY X##GX G##.# .G.P. ###G# X..AX	The other player moves closer.
	HITMOD -1	The other player attacks.
	STARTTURN	The start of the player's next turn ...