Network Working Group
Internet-Draft
Expires: November 2, 2014

J. Cranston
PSU

May 2014

Joel's Gaming protocol

Abstract

JGP (Joel's Gaming Protocol) provides a framework for a number of games utilizing a turn based seek and destroy style game-play.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 2, 2014.

Table of Contents

1. Introduction

1.1. Purpose

The protocol provide a simple interface for a multi-player game in a turn based free-for-all seek and destroy style game-play implemented via TCP. The actual rules may vary by implementation beyond what is required. The number of clients allowed in a single game instance is also variable, and depends on the rules set by the server.

1.2. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.3. Terminology

This specification uses a number of terms to refer to the roles played by participants

connection
: A transport layer virtual circuit established between two programs for the purpose of communication.

message
: The basic unit of communication matching the syntax defined in section 4

client
: A program that establishes connections for the purpose of sending requests.

server
: An application program that accepts connections in order to service requests by sending back responses.

  Any given program may be capable of being both a client and a server; the use of these terms refers only to the role being performed by the program for a particular connection.

Scan Hit
: A scan hit results from successfully locating an enemy, it contains the x,y coordinate of the the enemy at their last known position.

Player
: A player may be defined as the user interacting with a client, or the member of a game instance.

1.4. Overall Operation

The protocol operates in a request/response fashion. The process starts with the server sending a welcome message containing details about the server. The client then sends a login message to the server containing the clients username, the server will reply with a list of currently available games. The client can then connect to an existing game or start a new game using a connect message. The server will reply indicating a successful connection. Once the minimum number of clients have connected to an instance the server will send a StartOfGame message with the name of the opposing players. The clients then need to send a Start message with their starting position, once the players have sent their start positions the server will send a StartOfTurn message. Each client must then

send a turn action message, which can be; Move, Fire, or Scan. Once all clients have made their moves the server will send a EndOfTurn message. If both of the players are still alive it is followed by another StartOfTurn message and the process repeats. If a clients was destroyed then the server will send a EndOfGame message to the client indicating that they lost, the last remaining client will recieve a EndOfGame message indicating they have won. The game may also by terminated at any time by a client sending Quit message.

2. Client

1. The client is responsible for communicating the server messages to the player, and for querying the user for information needed by the server.

2. It SHOULD provide the user with the servers welcome message, and a list of available games to join.

3. It MUST query the user for their username, game-name, and starting location. Usernames and game-names MUST be limited to 25 chars [0-9|a-z|A-Z], starting locations MUST be between [0-9] for each the x and y axes.

4. It MUST query the user for a single turn action at the beginning of every turn.

5. Move actions MUST be a number between 1 and 9 excluding 5.

6. It SHOULD display the players remaining hit points.

7. It SHOULD inform the player when they have been hit.

8. It MUST provide the users with the results of a scan move in the form of the x,y coordinates of enemy players as provided by the server.

9. It MUST provide the user with the results of a fire action in the form of (hit or miss).

10. It SHOULD tell the player if they won or lost the game upon receiving a EndOfGame message.

11. Clients must respond immediately to heartbeat messages by echoing the message back to the server.

12. Clients MAY send heartbeat messages to insure an active connection with the server.

3. Server

1. The server is responsible for tracking the state of each game. This include the usernames of the players, their location on the game board, and current hit points.

2. The server SHOULD maintain timers to catch non responsive clients. A heartbeat message should be sent to the non-responding client, if the message is not returned the client should be removed from the game and disconnected.

3. The server MUST maintain a list of all currently active games. This includes games being played and those awaiting additional players. There is a limit of ten games awaiting players. The number of games actively being played is only limited by the server design.

4. The server MUST only allow a single action to be performed by each client on a specific turn. Such a message will be referred to as a valid message.

5. The server MUST respond to valid Move messages by updating the clients position.

6. The server MUST respond to valid Fire messages by checking the location specified against the locations of the other clients and if a client occupies the location then calculate if a hit has occurred, the formula for this may be server specific. If a hit has occurred it must update the hit points of the client occupying the location. The value may depend on the rules of the specific server but MUST be a integer value between one and ten.

7. The server MUST respond to valid Scan messages by checking the location specified and the eight squares surrounding it against the positions of other clients. If a player is within this nine square area the server MUST report its position, in the EndOfTurn message, to the client performing the Scan.

8. The server SHOULD send an available games message when changes to the available games list occur.

9. The server MUST immediately respond to heartbeat messages by echoing the message back to the client.

10. The Server MAY implement additional scanning rules, allowing probable detection of players outside the nine squares that are required. In such cases the position reports may not indicate

actual player locations, but possible locations. A EndOfTurn message with a single location will always be an actual player position.

4. Messages

The messages can be divided up into client-to-server and server-to-client types. All messages are in ASCII text and are terminated by a line feed, ASCII code 10.

The The following terminology is used:

[0-9] a single ascii decimal digit form zero to nine.

[0|1] a single ascii decimal digit of zero or one.

[00-99] two ascii decimal digits, 00 to 99.

{[0-9][0-9]}* zero or more two digit pairs

<string> is a string of chars. Unless otherwise specified it may contain any printable character.

4.1. Client to server messages

L<string>	login to server with <username>.
C<string>	Connect to or create <game-name>.
S[0-9]0-9]	Start at position [x][y].
F[0-9]0-9]	Fire at position [x][y].
P[0-9]0-9]	Scan position [x][y].
M[0-9]	Move in direction 1-9.
H<string>	Heartbeat; string is time stamp of the message>.
Q	Terminate the game and connection.

4.2. Server to client messages

W<string>	Welcome; <welcome message, with colon as endline char>.
N[0-9]	NewGame; Reply to request to start or join a game. [0]:=Successful join, [1]:= successfully created a

new game, [2]:= failed to join or create. [3-9] are unused.

X[0-9] Error; X1:=login error, username already in use. X0 and X[2-9]x are currently undefined.

A[0-9]<string> Available games; [number of games] in <colon delimited list>.

H<string> Heartbeat; string is time stamp of message.

B<string> StartOfGame; Game starting with <Opponent player name>.

T[0-9] StartOfTurn; Turn is starting [players hit points].

O[0|1] EndofGame; [1]:= client won, [0]:= client lost.

E[0|1][00-99]{[0-9][0-9]}* EndofTurn; [0|1] field indicates hit(1) or miss(0), [00-99] indicates the number of {[x][y]} pairs that will follow, each of which is a scan-hit coordinate.

5. Description of Message Flow

5.1. Connection

Once the TCP session is established The server will send a Welcome message, W<welcome string> containing a string providing information about the server.

5.2. Login

The client will then need to send a login message with a the clients username, If this name is unavailable the server will reply with the error message X1.

5.3. Game Setup

Upon successful login the server will send a message containing a list of available games, A[number games]<colon delimited list>. The client can then send a Connect message, C<"name of game instance"> containing a game instance string that the client wishes to connect to. The server will replay with a NewGame message N[0|1|2]. A [0] indicates the client successfully joined a game, a [1] indicates the client successfully created a game, and a [2] indicates a failed attempt to join or create a game.

5.4. Starting the Game

Once a game instance has connected to the minimum number of players as determined by the server configuration it will send all of the clients a StartofGame message, B<players>, containing the name of the opposing player(s). At this point the clients must send a Start message, S[x][y], containing their starting position as a single ascii digit from 0 to 9 for the each of the x, and y coordinates.

5.5. Starting a Turn

Once the server recieves a starting location for each of the players, it will send out a startOfTurn message T[0-9], containing the players current/initial hit points.

5.6. Turn Actions

Each client will after receiving the StartOfTurn message send one of the three turn action messages.

Move
: The move message, M[1-9] request that the sever move the player in one of the eight possible direction. Figure 1 illustrates this with the player centered at [X].

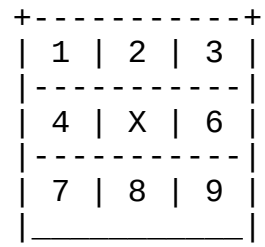

Figure 1

Note that 0 and 5 are not valid directions and the server will quietly ignore such messages.

Fire
: The fire message, F[x][y] indicates the player would like to initiate an attack on the coordinate provided, where [x] and [y] are a single ascii decimal digit.

Scan
: The scan message, S[x][y] indicates the player would like to perform a scan at the coordinates provided, where [x] and [y] are a single ascii decimal digit.

5.7. Ending a Turn

Once all the players have sent valid turn action messages, the server will send all of the clients a endOfTurn message. The end of turn message contains a flag indicating if the player scored a hit, a '1' indicating a hit and '0' indicating no hit. The message also contains the number of scan hits and there locations.

5.8. Ending a Game

If the server detects that a player has been eliminated it will send the client a EndofGame, O[win|lose], message after the end of the turn and the client will be disconnected. This message may also be sent if all of the other clients have been eliminated or disconnected. [win|lose] is a single ascii char of '0' or '1', with '1' indicating a win.

5.9. Heartbeat Messages

Both the client and server should respond to heartbeat messages by retransmitting the received message back to the sender. Both the client and server may send a heartbeat message after a undefined period of inactivity in order to insure the connection is still intact. Failure to respond to a heartbeat message may result in the connection being terminated.

6. Security Considerations

This memo raises no security issues;

7. IANA Considerations

There are no IANA considerations

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Author's Address

Joel K. Cranston
Portland State University

Email: jcrans2@pdx.edu