LUBE

by Bartbes

Improvements are welcome, please send via LÖVE Club (the forums)

Client

Easy part:
The client is located in “Client.lua”.
The table that contains every client function and variables is called “client”.
For most (if not all) actions there is a function, with an easy to remember name.

Functions:
(all in client table/namespace)

Init
Returns: –
Arguments: -socket type (string, optional, default=udp)
Does: the initializing

setCallback
Returns: –
Arguments: -callback (function): receives all incoming messages
Does: set the callback used with *update*

setHandshake
Returns: –
Arguments: -handshake (string): string containing handshake with server
Does: sets an optional handshake

connect
Returns: –
Arguments: -host (string): ip/dns name
-port (number)
-dns (boolean): if name should be DNS-translated
Does: “connects” socket
Remarks: Implemented in socket type

disconnect
Returns: –
Arguments: –
Does: “disconnects” socket
Remarks: Implemented in socket type

send
Returns: udp: -results of sendto call
else: unknown, see documentation of socket type
Arguments: -data (string): anything you want to send
Does: sends data to server
Remarks: Implemented in socket type

receive
Returns: -data (string): anything the server sent to you
Arguments: –
Does: receives data from server
Remarks: Implemented in socket type

update
Returns: –
Arguments: –
Does: checks if there is incoming data and calls the callback set by setCallback if necessary
Remarks: recommeded to use in update() callback of Löve

Variables:
(all in client table/namespace)
host (string): as set in Init, the host IP
port (number): as set in Init, the host port
connected (boolean): if socket is “connected”
socktype (string): type of socket (default=udp)
protocol (string): protocol of socket type (tcp/udp)
socket (socket): as returned by LuaSocket
callback (function): to be called whenever update gets data
handshake (string): string to be sent on connect and disconnect

Server

Easy part:
The server is located in “Server.lua”.
The table that contains every client function and variables is called “server”.
For most (if not all) actions there is a function, with an easy to remember name.
Is a lot like the client.

Functions:
(all in server table/namespace)

<u>Init</u>
Returns: –
Arguments: -socket type (string, optional, default=udp)
Does: the initializing

<u>setCallback</u>
Returns: –
Arguments: -callback recv (function): receives all incoming messages
-callback connect (function): receives all connect messages
-callback disconnect (function): receives all disconnect messages
Does: set the callbacks used with *update*

<u>setHandshake</u>
Returns: –
Arguments: -handshake (string): string containing handshake of client
Does: sets an optional handshake

<u>send</u>
Returns: udp: -results of sendto call
else: unknown, see documentation of socket type
Arguments: -data (string): anything you want to send
-rcpt (string/ip, optional, default=all): client to send to
Does: sends data to 1 client or all clients depending of rcpt argument
Remarks: Implemented in socket type

<u>receive</u>
Returns: udp: -results of receivefrom call
Arguments: –
Does: receives data from clients
Remarks: Implemented in socket type

<u>update</u>
Returns: –
Arguments: –
Does: checks if there is incoming data and calls the callback set by setCallback if necessary
Remarks: recommeded to use in update() callback of Löve

Variables:
Clients (table): contains a list of clients, index = ip, value = port, <u>***IT DOES NOT SUPPORT MULTIPLE CLIENTS SHARING AN IP***</u>

(following in server table/namespace)
socket (socket): as returned by LuaSocket
handshake (string): string to be sent on connect and disconnect
callback (function): to be called whenever update gets data
connectcallback (function): to be called whenever a client connects
disconnectcallback (function): to be called whenever a client disconnects
protocol (string): protocol of socket type (tcp/udp)
socktype (string): type of socket (default=udp)

Binary packing

Easy part:
The binary packing is located in “Binary.lua”.
The table that contains every client function and variables is called “bin”.
For most (if not all) actions there is a function, with an easy to remember name.

Functions:
(all in bin table/namespace)

pack
Returns: -result (string)
Arguments: -in (table)
Does: turns the table into a string

packvalue
Remarks: for internal use only

unpack
Returns: -result (table)
Arguments: -in (string)
Does: opposite of pack, turns string into a table

unpackvalue
Remarks: for internal use only