Irrlicht Non Realtime Networking

Documentation, Team 3

Networking

The connection between two PCs is established with a TCP-Socket on each PC. One of these sockets will serve as the host (server socket) and the other will serve as the client (client socket). The host opens a socket (on a certain port) and waits for connection(s), the client socket will connect to that socket on that port.
Over that connections data in form of simple byte streams can be exchanged. Objects that are sent over that connection need to be serialized on one PC and deserialized on the other PC.
We allow two methods of establishing a connection: the two PCs can directly connect to each other, in that case the client socket needs to know the server socket's IP address, or the PCs can connect to a WebService that will pair them together and establish a server/client socket for them depending on generated session ID (odd ID - server socket, even ID - client socket) in the list of registered PCs.

Classes

NonRealTimeNetworkingUtilities

Offers general functionality related to non realtime networking, like establishing connections etc.

Constructors

NonRealtimeNetworkingUtilities();
Default constructor, the portnumber on which the connection is established is set to a default value (portnumber 6).

NonRealtimeNetworkingUtilities(int portNumber) { this->portNumber = portNumber; };
Portnumber on which the connection is going to be established is given as a parameter.

NonRealtimeNetworkingUtilities(char* masterServerHostAddress);
IP Address of WebService is set to given parameter.

Public Methods

The following methods are publicly accessible for everyone using the engine.

Getters/Setters:

void setBuffer(char* buffer);
Sets the buffer of this instance of NonRealtimeNetworkingUtilities. Values that are going to be sent over the network will be buffered there, as well as data retrieved over the established connection are stored in this buffer.

char* getBuffer() { return buffer; };
Retrieves the buffer of this instance of NonRealtimeNetworkingUtilities. Values that are going to be sent over the network will be buffered there, as well as data retrieved over the established connection are stored in this buffer.

void setPortNumber(int portNumber);
Set portnumber to establish connection.

int getPortNumber() { return portNumber; };
Get portnumber that is used to establish connection.

Server side:

void openServerSocket();
A TCP Server socket will be opened on the previously set portnumber.

void acceptClient();
The opened TCP Server Socket waits for connections and will accept a client.

void hostGame(int portNo);
Summarizes both openServerSocket() and acceptClient().

Client side:

void openClientSocket(char* ipAddress);
Opens a client socket, connects to a server socket on given IP address with previously set portnumber.

void joinGame(char* ipAddress, int portNo);
Synonym for openClientSocket(char* ipAddress);

Sending/receiving data:

void sendData();
Data that is stored in buffer (has to be set before!) will be sent to the other TCP socket.

void receiveData();
Socket waits to receive data, received data will be stored in the buffer.

Terminate:
void closeConnection();
Connection will be closed (goes for both, server and client socket).

WinSock Errors
int getWSALastError();
Returns the (int) error code of the last thrown winsock error.

Web Service
int establishConnection(char* gameName, int portNo);
Registers on the web service, informing it about the desired game and portnumber. The web service decides then whether to open a server or a client socket depending on generated session ID. Summary of different web service methods.

void registerOnTheServer();
Registers on the webservice to play a game whose name is given as a parameter.

char* getOpponentsIpAddress();
Retrieves the IP Address of the opponent that was assigned by the WebService.

char** getGamesList();
Retrieves a list of games registered on the server.

void setGameName(char* gameName);
Sets gameName with which to establish a connection on the webservice.

int getSessionId();
Retrieves ID of current session between client and WebService.

void initializeWS(char* masterServerHostAddress);
Initializes WebService that will be accessed/used to assign PCs to one another.

NonRealtimeNetwokingException
Is thrown when errors related to this engine extension occur.

Constructor
NonRealtimeNetworkingException(const char *message) : errorMessage(message) {};
Creates the Exception with a given error message.

Public Methods
const char *what() const throw() { return this->errorMessage; };
Throws the exception with the attribute of the error message.