LibNRG – A Networked Multi-player Game Framework

Specification

Problem

More and more video games in recent times are focusing on providing a multi-player experience to enhance their appeal and longevity, however implementing networked multi-player functionality into a game from scratch can be difficult, often meaning that a third party library is used.

Currently most networked multi-player libraries provide only a low level abstraction enabling the exchange of packets, and it is up to the programmer to decide what data they contain and how and when they are sent between players. This can lead to implementation problems where the protocol created sends more data than is necessary or gives too much trust to players, allowing them to cheat by altering the data that they send to others.

Other libraries that are commonly used have proprietary licenses or are tied to a particular service, such as Microsoft's Games for Windows Live [1], which limits their flexibility.

Objectives

LibNRG (Networking for Real-time Games) aims to solve the above problems by providing an open source, permissively licensed networking library with a high level abstraction that provides programmers with a relatively bandwidth-efficient and cheat-resistant protocol.

When completed, the library should give access to a Client / Server networking model including functionality such as maintaining the state of players on the server, determining the changes in game state that players need to know about, serialising and de-serialising data, and sending this data to players at regular intervals.

It should be possible to create small-scale games using the library where one game instance corresponds to one server, and up to around 32 clients depending on the complexity of the game. Suitable genres of game that should be able to be created include First-person shooters (FPS), platformers, or any game with a small number of players and a focus on real-time, low-latency game play. The library will not be suitable for Massively-multilayer online games (MMOs) as these usually require a highly scalable back-end split across multiple servers.

Since the library focuses on real-time games, it should be based on the User Datagram Protocol (UDP) so as to avoid delays that may arise from retransmission if the Transmission Control Protocol (TCP) were used. Because of this, the library must provide its own abstraction of a persistent connection for programmers to use, since UDP is connectionless.

Some form of interpolation and/or lag compensation should also be offered by the library to ensure that it provides an enjoyable game experience. It should also be able to cope with small amounts of packet loss, though it need not be engineered to work well on very high latency, unstable or slow connections.

In an effort to reduce players cheating, LibNRG should be designed to place a small amount of trust with player clients, and have them send only necessary information such as mouse and keyboard state, from which changes in the server's game state can be determined. It should also avoid sending unnecessary data from server to client by making sure only the changes in game state that clients don't already know about are sent.

The server component of the library should take care to ensure that it is resilient, and cannot be crashed by rogue clients or malformed inputs. In addition, the library could provide some way of encrypting player connections, granting them confidentiality and integrity, however the library itself need not provide any notion of accounts, client authentication or actually implement its own form of encryption.

The library should be written in C++, and compile on the GNU/Linux operating system as a shared library (.so file). It could also be extended to additionally compile on Microsoft Windows if there is time remaining after the other objectives have been completed.

The source code to the library should be released under a “copyleft” license which permits modification and redistribution of modifications, so that anyone is free to make changes or improvements as they see fit. The library should also be free for commercial and closed-source games to use, and provide a good amount of documentation using a system such as Doxygen.

In addition to the library itself, an example game should be produced which makes use of the library and demonstrates its use. The full source code to the game must be provided so that it can satisfy this objective.

The library could also allow for replay files to be saved by clients that contain all the packets it received. This would make it possible for games to be watched back at a later date by having a local server replay the packets saved in the file.

Methods

The first part of the project that will need to be developed is the basic communication layer over UDP. This will comprise of classes to represent sockets, packets, and an abstraction of player connections.

Once data can be sent and received, the client and server classes can be created by making use of the previously created socket and packet classes. Following this, the high-level abstraction can be built by adding classes for game entities (e.g. players, projectiles), and associated serialisation methods to encode their data and send it over the network.

The client and server will then need to be expanded upon so that player input received by clients can be sent to the server, and the server can invoke methods to determine what changes in game state the player input will cause. At this point the server's functionality should be augmented to track a history of player's state, and the changes in game state that they need to be sent based on the player's last acknowledged state. The client will need to be able to receive and apply updates in game-state and manage the creation and deletion of any entities involved automatically.

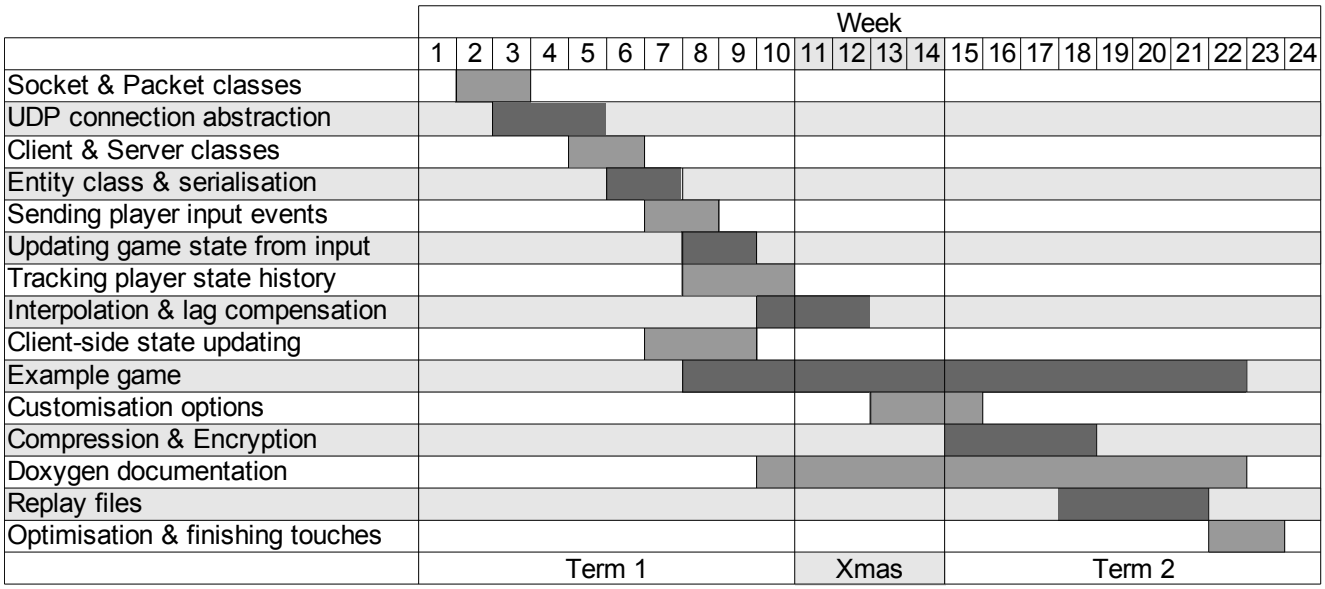
Once the basis of the library is complete and functional, extra features such as interpolation, lag-compensation, compression, replay files, and methods to customise how the library operates – such as how many state updates are sent per second – can be added, as well as ways to obtain useful information about the connection between server and client e.g. packet round-trip-time and jitter.

Aspects of the example game that are unrelated to networking, such as displaying graphics, can be worked on in parallel to the development of the library, however the networking side of things will depend on the library being in a usable state.

Each component of the library should be tested as it is created to make sure it behaves as expected;

the example game should also be created to make use of all aspects of the library, so that it effectively doubles as a test-suite.

Timetable



Resources

The project will be written on my personal computer running GNU/Linux on which I have root access, and will be compiled with gcc. The source code will regularly be committed to a git repository on Github.com, which doubles as an online backup copy. If a Windows version of the library needs to be compiled, then this can be done via the Mingw cross-compiler from within GNU/Linux.

In the event that my personal computer becomes unusable due to malfunction, theft, or otherwise then the computers in the terminal room of DCS will serve as an adequate backup system on which the project can continue to be developed, using my Warwick unix account with standard access privileges.

It will be possible to test the project solely on a single computer using its loop-back network interface. If the library needs to be tested in environments with higher latency, then then 'tc' command can be used to add latency to the loopback device, or alternatively since I have access to another computer in a different geographical location, the server side can be run there.

Legal, social, ethical and professional issues

Certain countries have laws that prohibit the import or export of cryptographic software [2]. To avoid these issues, LibNRG will not implement its own encryption of data, but rather provide the ability for its users to add encryption if they so choose.

Bibliography

[1] Microsoft “Games for Windows – LIVE F.A.Q.” [Online] Available: http://www.xbox.com/en-GB/Live/PC/FAQ

[2] The Wassenaar Arangement “List of Dual-Use Goods and Technologies” [Online] Available: http://www.wassenaar.org/controllists/2011/WA-LIST%20%2811%29%201%20Corr/WA-LIST%20%2811%29%201%20Corr.pdf