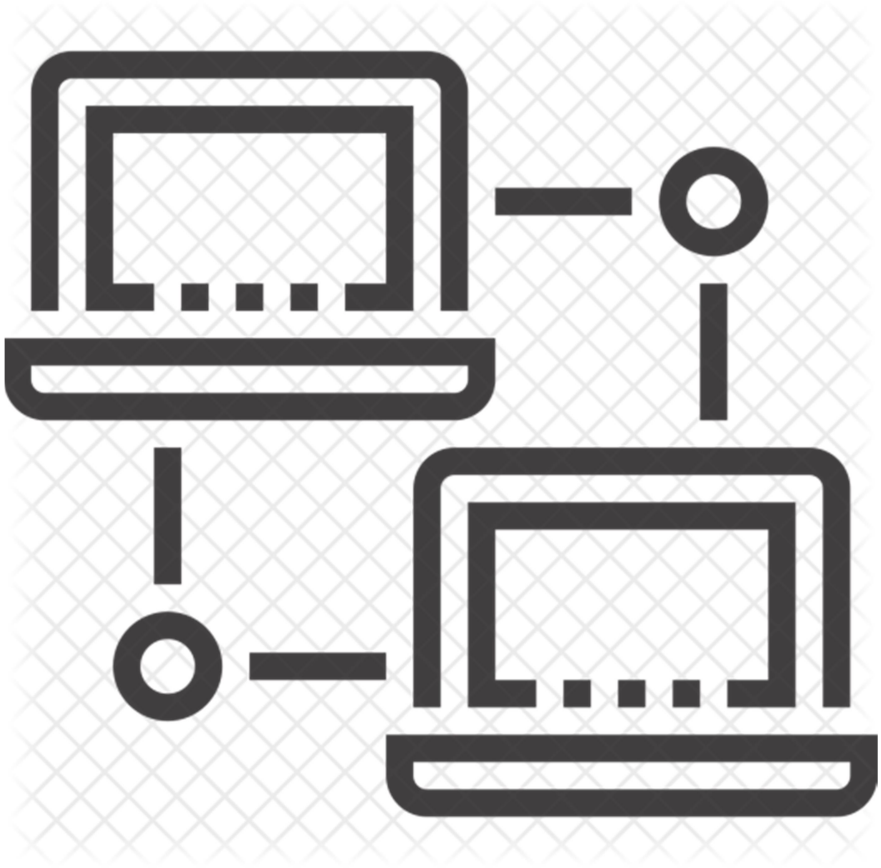
VERSION 0.2

MAY 5, 2018



NETWORK PARTY CHAT SYSTEM

COMPLEX GAME SYSTEMS ASSIGNMENT

CREATED BY: DANIEL MARTON

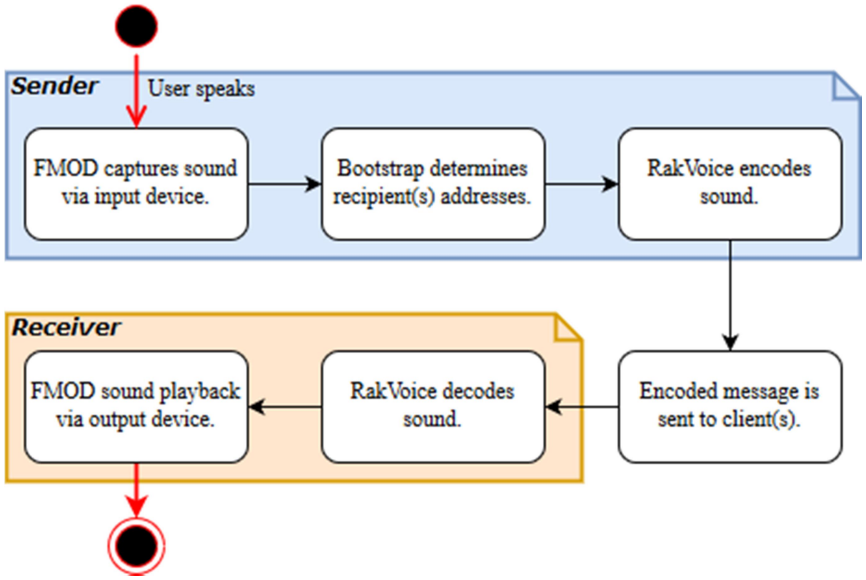
TABLE OF CONTENTS

SYSTEM OVERVIEW

For my complex game system assignment, I will develop a static LIB that can perform as a networked party chat for clients.

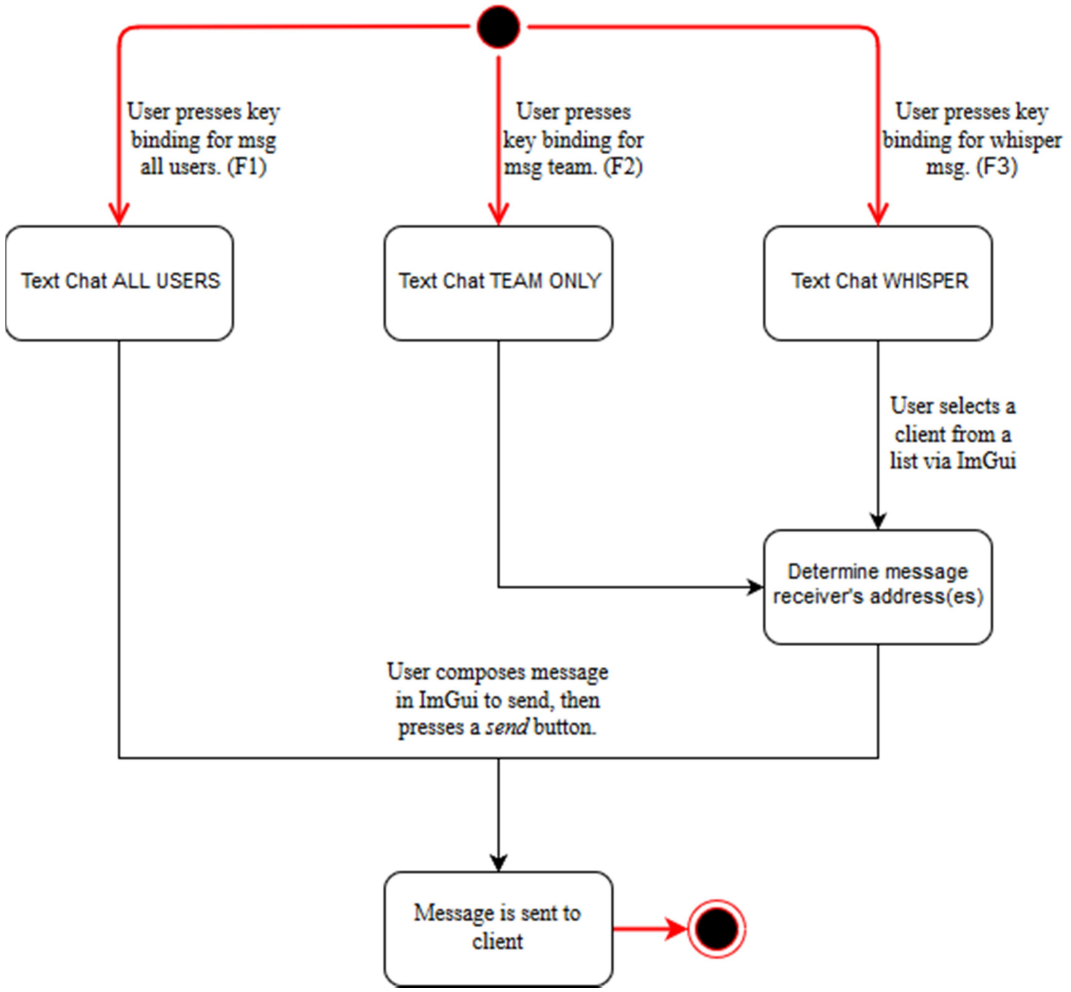
With a mixture of Server-Client authority & peer to peer models, the system will be able to perform a variety of features such as containing per client, information containing a profile name, team identifier & a list of all connected clients.
This is so that each client can send text messages either via a Server-Client broadcast, or to a specific client.
The reason I chose the text messaging service to follow a server authority, instead of peer to peer is so that if more resources were to be allocated on expanding the development of the system; a monitoring service could be attached seamlessly into the already built processes.
This could allow systems that would restrict clients from using vulgar language to dissipate toxic behaviors from clients using the system, as an example.

As well as text messages, clients can also communicate with each other by using a connected microphone to send voice messages to other clients who are matched on the same team as them.
The service will follow a peer to peer module as it is faster to send the data to the clients directly in this case. In order to maintain the microphone's recording processes be to run concurrently with the rest of the system, I will use multithreaded programming to achieve this. A dedicated thread will be made to record microphone input & from the main thread, only when a user chooses to; will the sounded that is being recorded will be sent over the network to its recipients. (Push to talk method).

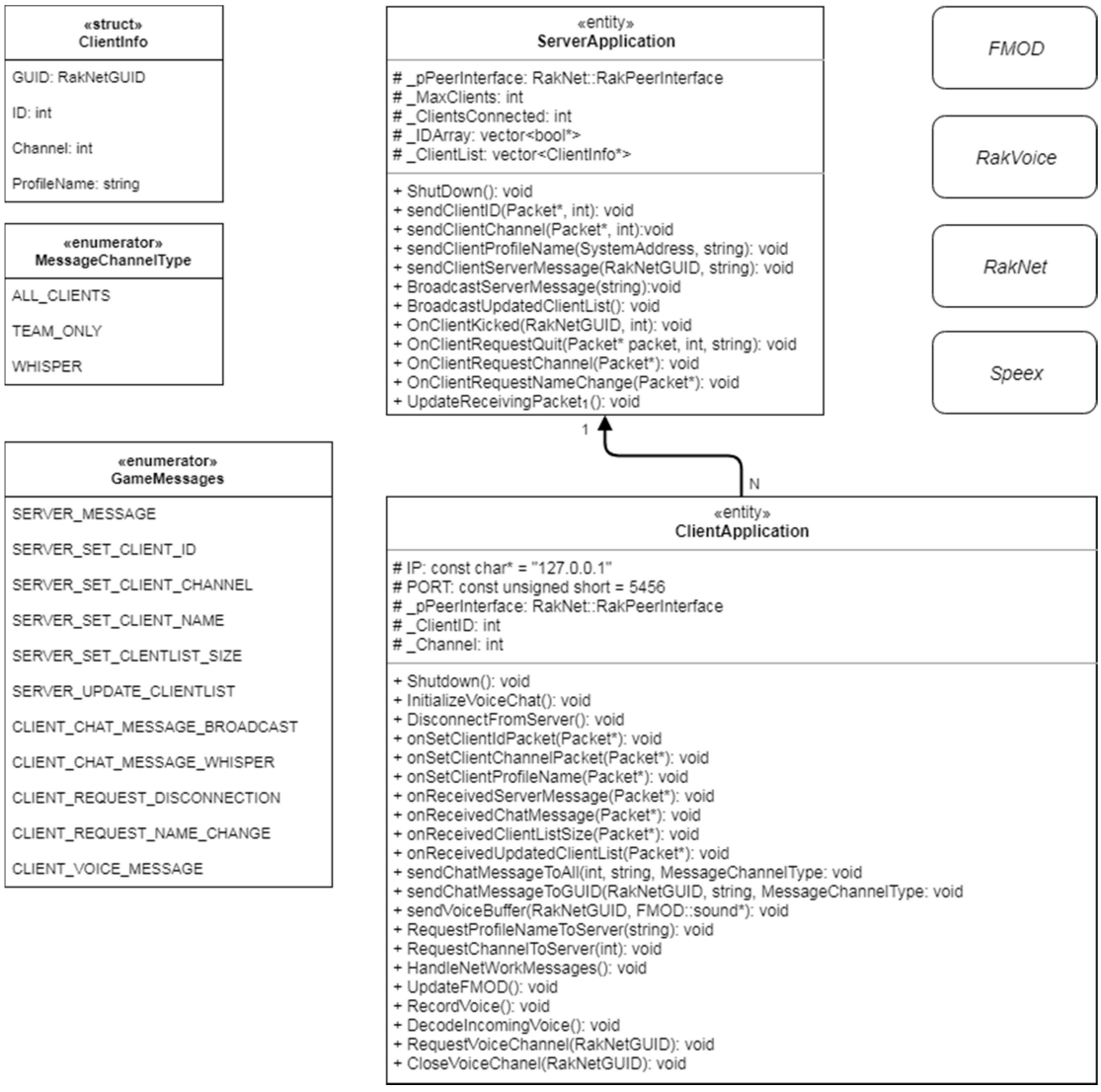


The event process for a client when communicating via voice channels to their teammates.

The event process for when a client selects a recipient(s) & then, composing a text message before sending it to the targeting client(s).



A simplified class diagram showing the main functionalities required in order to get the system to a minimum viable product level.



MINIMUM VIABLE PRODUCT

As a base system, the plugin will support text party chat (with the different channels to communicate on) in order to demonstrate its capabilities. Once this minimum target has been reached, I will expand on the system to support voice chat communication on its own processing thread, alongside the text communication component.

RISKS & CONCERNS

Encoding & decoding sounds through the RakVoice library, after they have been captured through FMOD; is a foreseeable obstacle. As I have not worked with either libraries & this particular programming problem is something different from my usual practices. However, as it has been mentioned in the reference material, the pitched process of voice communication through a network (FMOD > RakVoice > RakNet) is a viable solution to this issue.

Also as a concept of structuring a message to send over the network, the way the packets are created can be ambiguous as to how the system will work in the back end. For example, determining how large the chunks of sound should be when being processed through the input device, whilst it is capturing large, continuous streams of sound will prove to be a challenge.

Finally, networked voice quality is a particular concern I have for this project – Being able to retain a satisfactory standard of audio quality & minimizing the amount the network traffic associated with the system to allow network resources to be allocated to other systems. Essentially maximize quality whilst retaining efficiency.

THIRD-PARTY LIBRARIES

In order for the NPC system to function as an application, it will rely on the following external libraries to achieve its goals:

- C++ standard library for a code base to work with.
- RakNet for sending & receiving network packets.
- RakVoice for processing sound data on the network.
- Speex for encoding & decoding sounds.
- FMOD for recording & sound playback.

The demo application will also be dependent on the libraries that are list above, but will also rely on the following:

- AIE Bootstrap for a front-end to display party & client information.
- Networked Party Chat system for all the back-end functionalities.

FEATURE MAP

These are the main objective milestones of the software.

Milestone	Completed
MINIMUM VIABLE PRODUCT	
Communication channel / teams	Yes
Text chat to clients	Yes
EXTRA FEATURES	
Multithreaded voice recording	Yes
Voice chat to clients	In progress
Dynamic chat rooms	To be completed
Voice filtering option	To be completed
INTEGRATION DEMOS	
Aie bootstrap integration demo	Yes
Unity engine integration demo	To be completed
Unreal engine integration demo	To be completed

POSTMORTEM

Due to time constraints, I was not able to complete all the main milestones for the application. However, the minimum viable product was achieved, as is the demo application using the Aie bootstrap.

The voice chat process relies on the sample being created, at this current time; the user can hear their own voice as a result. Also, the decoding on the client's end is an uncompleted feature at this point and is disabled for the moment.

Also to note, creating the release build of the application caused some very strange linking errors. They indicate that bootstrap.lib is missing some definitions, however I am confident that the issues being outlined by visual studio's compiler are falsifying as due to previous applications relying on the same bootstrap library had no issues in compiling a release build. I believe the issue is hidden somewhere within the older RakNet libraries, since RakNet has been abandoned & is an outdated library itself.

RESOURCES

RakNet:

https://gafferongames.com/post/reading_and_writing_packets/

http://www.raknet.net/raknet/manual/tutorial.html

http://www.jenkinssoftware.com/raknet/manual/index.html.

http://www.jenkinssoftware.com/raknet/manual/programmingtips.html.

RakVoice:

http://www.raknet.net/raknet/manual/rakvoice.html

Speex:

https://www.speex.org/docs/manual/speex-manual.pdf

https://www.speex.org/docs/api/speex-api-reference.pdf

FMOD:

https://www.fmod.com/resources/documentation-api

https://www.fmod.com/resources/documentation-api?page=content/generated/overview/terminology.html#/

Multithreading:

https://aie.instructure.com/courses/40/pages/multithreaded-programming