Implementing a gather.town clone in Links

Weston Everett

MInf Project (Part 1) Report

Master of Informatics
School of Informatics
University of Edinburgh

2021

Abstract

This skeleton demonstrates how to use the `infthesis` style for undergraduate dissertations in the School of Informatics. It also emphasises the page limit, and that you must not deviate from the required style. The file `skeleton.tex` generates this document and can be used as a starting point for your thesis. The abstract should summarise your report and fit in the space on the first page.

Acknowledgements

Acknowledgements go here.

Table of Contents

Chapter 1

Introduction

The preliminary material of your report should contain:

- The title page.
- An abstract page.
- Optionally an acknowledgements page.
- The table of contents.

As in this example `skeleton.tex`, the above material should be included between:

```
\begin{preliminary}
    ...
\end{preliminary}
```

This style file uses roman numeral page numbers for the preliminary material.

The main content of the dissertation, starting with the first chapter, starts with page 1. ***The main content must not go beyond page 40.***

The report then contains a bibliography and any appendices, which may go beyond page 40. The appendices are only for any supporting material that's important to go on record. However, you cannot assume markers of dissertations will read them.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default 1.5 spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages such as `geometry`, `fullpage`, or `savetrees` change the margins of your document. Do not include them!

Over length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

1.1 Using Sections

Divide your chapters into sub-parts as appropriate.

1.2 Citations

Citations (such as [**?**] or [**?**]) can be generated using `BibTeX`. For more advanced usage, the `natbib` package is recommended. You could also consider the newer `biblatex` system.

These examples use a numerical citation style. You may also use (Author, Date) format if you prefer.

Chapter 2

Background Information

(introduction will of course cover that the project is a gather.town clone in Links)

2.1 Gather.town Overview

Gather.town (https://gather.town/app) is a website which allows a user to move around a map and speak to other users in the same ”room”, automatically beginning voice/video calls with anyone in their close proximity. It also includes other features such as areas where the range of your calls is changed (such as a table where you are only talking to other people at the table or a podium where everyone in the room can hear you), objects that allow you to load documents for others to see, and a space/avatar designer for users to customize their avatars and surroundings. Other quality of life features such as automatically muting and ending your video while tabbed off the screen, the implementation of visual emotes (through emojis in speech boxes on screen), and automatic echo-suppression in calls among others helps improve the platform further. The website contains multiple public gathering spaces for various topics where anyone is allowed, although it is also possible to create private or more specialized spaces that require a link. For these private and public spaces, Gather.town allows several tiers of “role” so that the space can be managed and built according to the creator’s liking, including roles like “Moderator” for managing any community that forms.

The application itself uses peer to peer connection over WebRTC (which will be discussed later) making it fairly cheap to run as extensive servers are unnecessary even though it advertises large events (as it charges for any spaces with more than 25 people) which create a large amount of traffic. Despite the peer to peer structure, it still seems to allow a large number of simultaneous calls (up to 9 video tracks simultaneously, and many more audio tracks), suggesting that the framework it runs calls over is not overly strenuous on a network. Gather.town also seems fairly robust over several different browsers and offers guidance on at least Google Chrome, Firefox, and Safari.

In theory, the design of gather.town allows workplaces and events to emulate the feel of an in-person meeting as small groups of people can easily breakaway from a larger meeting to talk about a certain topic, and it is then easy to move between these sub-

groups. It is also easy and intuitive to walk around the room and see conversations that are currently on-going. Certain events have even taken the step of designating areas for different topics using the room designer, which lets people wander over to conversations that are likely to be of interest to them. Gather.town also seems to want to attract people to use it as more of a place to hang out with friends, adding several games in the objects menu such as poker and popular web party games like Skribble.io and One Night Werewolf.

In practice, many of these benefits over traditional calls are realized, with the ability to see other conversations and change between them quickly featured chiefly among them. Many of the advertised features also seemed to work through my own admittedly limited testing, although this is supported by a lack of complaints of broken features (beyond the usual FAQ's about how to use certain aspects of the site) on online reviews along with their very positive bent (https://www.producthunt.com/posts/gather-town/reviews)

2.2 Gather.town Usage

In order to understand the strengths and weaknesses of a platform, it is also important to know how it is used and if it is expanding. For instance, Gather.town claims to have found a great deal of success in large events with gather.town alone claiming to have held events for over 10,000 companies and organizations (https://www.gather.town/conferences), with similar competitors claiming smaller but still significant numbers. It is somewhat harder to find evidence of the platform being used commonly for more casual gatherings beyond a few expired facebook game night groups from universities, which one can assume are likely to fade away even more once the pandemic is over. However, statistics on gather.town usage (https://www.similarweb.com/website/gather.town/) does suggest that its user base is increasing, with total monthly visits going from 2.6 million to about 4 million over the past 6 months. Although this is obviously in no small part due to the pandemic it also speaks to the advantages of the medium over traditional voice/video calls or chat rooms. Oddly enough, one of the top referring pages to gather.town is learn.ed.ac.uk, implying a considerable number of university-sponsored events.

2.3 Gather.town Issues

However, users also have complaints about the system (https://news.ycombinator.com/item?id=25039
For example, in gather.town it is difficult to tell who can hear your conversation as it does not show a radius on the screen of where you can be heard and the usual way of telling if someone can hear you (their video feed appearing on your screen) does not happen if there are already a large number of people in the conversation, which can lead to privacy issues. Its binary "if you are this close you are part of the conversation" also leads to multiple issues such as accidentally joining conversations if you walk too close, awkwardness about whether or not to join a conversation (as it provides no visual signals about how welcome new people are to a call), and a certain abruptness

to entering a conversation as opposed to real life. In addition, some users less experienced with the control system (which uses the WASD set-up common in games) have found it difficult to use, and some have even reported nausea due to the way the screen moves.

2.4 Gather.town Clones

The success of gather.town combined with the dissatisfaction of some users has meant there are a large number of clones/competitors with similar but distinct products such as Sococo (https://www.sococo.com/), Mozilla Hubs (https://hubs.mozilla.com/), and Remo (https://remo.co/), with all of these taking somewhat different approaches to the same idea.

For example, Sococo and Remo are both more abstract and accessible than gather.town and seem targeted towards remote workplaces and events respectively. They implement features such as click to move to be more natural to those less familiar with technology and a static screen to avoid nausea. In addition, Sococo implements simple dynamic avatars in order to quickly show what someone is doing (such as sharing their screen or in a video call) and "knocking" to ask permission to enter a conversation. Remo on the other hand with its focus on events implements "business cards" to make it easy to connect with people met at their events and separate modes for presentations. However in implementing a simpler interface to be more accessible, these platforms have made themselves less immersive (especially as Sococo and some like it have ditched most avatar customization beyond a name) which of course lessens the "feel" of an in-person event.

On the other hand, Mozilla Hubs takes almost the opposite approach with a full 3D VR-enabled setup with more complicated controls than gather.town (using specific keyboard inputs) and spatial audio. It also includes Spoke, which is its own room designer with a great deal of control. While this of course looks impressive and allows more flexibility than the more streamlined Sococo and Remo, it also comes at the cost of possibly alienating less technically-savvy people and increasing the barrier of entry.

Many of these platforms have found success due to the variety of features they offer, for instance Sococo claims to have multiple large companies currently using (and paying) for their services such as JetBlue (A large American Airline). Remo also claims some similar successes such as convincing Northstar and several Chambers of Commerces to use their platform (https://remo.co/success-stories).

2.5 Links

Links is a functional programming language for web applications that uses a single source code to generate code for all tiers of a web application, as opposed to the more traditional method of using separate languages for database and client (often SQL and Javascript respectively). Although Links is a relatively small language and therefore does not have the extensive set of libraries and API's of a more mature language there

are still some resources available. These are mostly in the form of several example projects from the Links github and Documentation found on Links wiki.

Links has a web server model that will be used in this project (detailed https://github.com/links-lang/links/wiki/AppServer) which allows Links to easily map URL's to Links functions and call those functions using that mapping when the appropriate URL is accessed. Of course, each function called using this routing table must return a page to be displayed.

Links also allows the use of functions from Javascript through the use of the Foreign Function Interface (FFI) which will be used in this project mainly to access WebRTC (detailed later). Although these functions are in Javascript and are therefore cannot be used with databases, they allow access to many functionalities that would otherwise be inaccessible in Links. This will be used initially to prototype video calling API before being replaced as much as possible by native code.

Various applications have been implemented in Links already that are relevant to this project in the examples (http://examples.links-lang.org:8080/). One of these is an application which includes draggable lists (useful for implementing mouse-based movement), and a different application where breakout is implemented which includes example code for keyboard-based movement. Another example app in this provides a way to use a login to authenticate users. Other provided example code (https://gist.github.com/SimonJ details a way to have a server respond to new users by broadcasting a message to all current users, which will be useful for updating the clients with data required to interact with other clients.

However, Links currently lacks an API to support video calling, which WebRTC will be used as a base to develop.

2.6 WebRTC

WebRTC (which stands for "Web Real Time Communication) is an API developed by Google which is designed to facilitate direct peer to peer communication of various types between browsers on different devices. According to the developers, it currently is supported by "All modern browsers". The API is available in Javascript and therefore should be accessible through Links foreign function interface mentioned earlier.

To understand how WebRTC works it can essentially be split up into two main segments; the MediaStream API which provides a way to access media on the users device, and the RTCPeerConnection which allows two different devices to make a peer to peer connection.

The MediaStream API allows "media capture" where in this case media is primarily the audio and video feed of a camera and microphone but also includes things like a screen capture for screen sharing. In order to use the video/audio feed (and to get the necessary control of the devices) the API requests access to devices that follow the constraints supplied, and (if given permission from the user) provides the output of the devices as a stream.

However, the API also has the functionality to request a list of all available devices

which can allow the user to choose which the application will use or the application to select one that best fits its parameters. The API also can control selection using it's constraints, such as setting requested resolution for video or requiring that the selected source has audio properties. In addition, the API can listen for devices changing during runtime, for example if the webcam being used by the application were unplugged or a new one were to be plugged in. Once this event happens, it can then trigger the proper response.

The stream output by this can then be used in a variety of ways, such as splitting just the audio or video "tracks" from it (which would for instance allow muting in a video calling app but retain the live video). Of course, this data can also be sent to another user, using the RTCPeerConnection segment of the API.

The RTCPeerConnections is somewhat more complex (for someone using the API at least) as it requires multiple steps to function correctly. Typically it is used for audio and video, although it can also send arbitrary binary if the clients support RTCDataChannel (a related API).

First, clients establish a connection through signalling which can be through various methods as WebRTC does not provide that functionality. However, as this project will be based around a website that all clients will be connected to it should not be an issue. This signalling is necessary so that both devices can be "on the same page" about how they are connecting, which typically is arranged using ICE servers.

Once the initial connection has occurred a client will store the connection as an object and an offer or accept behavior will be decided based on the configuration of the system (which is a design decision that will need to be made for this project). Once two peers have formed an initial connection an ICE service will search for and send them candidates for connecting until one is found and the peer connection is fully established.

Additionally, even before a peer to peer connection is fully established a media stream can be attached to the connection so that as soon as the connection is ready data will begin to be sent. Similarly, a Listener can be attached as well to be ready to immediately receive data.

Chapter 3

Conclusions

3.1 Final Reminder

The body of your dissertation, before the references and any appendices, *must* finish by page 40. The introduction, after preliminary material, should have started on page 1.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default 1.5 spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages such as `geometry`, `fullpage`, or `savetrees` change the margins of your document. Do not include them!

Over length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.