AAMOT OLE 19780220 UNIVERSITY OF OSLO 20220215 MSc PHYS
Public Audio Recording Software for Recording World Sounds
GNOME Gingerblue (gingerblue) version 3.0.1
http://www.gingerblue.org/thesis.pdf

GNOME Gingerblue 3.0.1 (Draft)

Ole Kristian Aamot

Thesis submitted for the degree of
Master in Electrical Engineering, Informatics and Technology
30 credits

Department of Physics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2022

GNOME Gingerblue 3.0.1 (Draft)

Ole Kristian Aamot

GNOME Gingerblue 3.0.1 (Draft)

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

In this thesis I wrote Free Audio Recording Software for GTK+/GNOME.

GNOME Gingerblue is Free Software available under GNU General Public License version 3 (or later) that supports immediate audio recordings in compressed Ogg Vorbis (VORBIS.COM) encoded audio files stored in the $HOME/Music/ folder from the line input on a computer or remote audio cards through USB connection through PipeWire (PIPEWIRE.ORG) and WirePlumber (https://gitlab.freedesktop.org/pipewire/wireplumber) Session Manager via the GStreamer (GSTREAMER.FREEDESKTOP.ORG) `record_pipe` API.

Multiple-Location Audio Recording 1.0 is specified for recording multiple-location audio recording configurations into Ogg Vorbis (VORBIS.COM) compressed audio files (Xiph.org) in the Free Software GNU autoconf (GNU.org) package GNOME Gingerblue 3.0.1 (GINGERBLUE.org) available under GNU General Public License version 3 or later.

The Multiple-Location Audio Recording 1.0 Specification will be implemented in GNOME Gingerblue 3.0.1 in ANSI C and available from http://WWW.GINGERBLUE.ORG/src/gingerblue-3.0.1.tar.xz with Source and Installation Packages for Fedora Core 35 (FEDORAPROJECT.ORG) and Ubuntu 21.04 (UBUNTU.COM).

Software Implementation

The implementation of the Multiple-Location Audio Recording 1.0 Specification (“as-is”) was completed (“as-of”) on October 25th, 2021 in C as specified in The C programming language (Kernighan/Ritchie, 2006) after 21 months of work that began on July 4th, 2018 as GNOME Gingerblue.

Source Code

- http://www.gingerblue.org/src/gingerblue-3.0.1.tar.xz
- https://download.gnome.org/sources/gingerblue/3.0/gingerblue-3.0.1.tar.xz

Fedora Core 35

- https://www.gingerblue.org/~ole/fedora/RPMS/x86_64/gingerblue-2.0.1-1.fc35.x86_64.rpm
- https://www.gingerblue.org/~ole/fedora/RPMS/x86_64/gingerblue-debuginfo-2.0.1-1.fc35.x86_64.rpm
- https://www.gingerblue.org/~ole/fedora/RPMS/x86_64/gingerblue-debugsource-2.0.1-1.fc35.x86_64.rpm
- https://www.gingerblue.org/~ole/fedora/SRPMS/gingerblue-2.0.1-1.fc35.src.rpm

Ubuntu 21.04 LTS

- https://www.gingerblue.org/~ole/ubuntu/gingerblue_2.0.1-1_amd64.deb
- https://www.gingerblue.org/~ole/ubuntu/gingerblue_2.0.1-1.debian.tar.xz
- https://www.gingerblue.org/~ole/ubuntu/gingerblue_2.0.1-1.dsc
- https://www.gingerblue.org/~ole/ubuntu/gingerblue_2.0.1-1_amd64.buildinfo
- https://www.gingerblue.org/~ole/ubuntu/gingerblue_2.0.1-1_amd64.changes
- https://www.gingerblue.org/~ole/ubuntu/gingerblue_2.0.1.orig.tar.xz

Contents

Introduksjon

I den første implementasjonen av grafisk lydopptaksprogramvare for Multiple-Location Audio Recording, applikasjonen GNOME Gingerblue versjon 3.0.1, som en applikasjon utgitt under åpen kildekode-lisens, kan vi reprodusere lydbølger i det hørbare spekteret for menneskelige lydopptak og lytting med tid-rom-frekvens notasjon.

Vi benytter prinsippene for å prosessere signalene som er motivert av de prosessene som er involvert i lytting.

En representasjon av lydsignalene hvor vi har tilgang til både tid og frekvensinformasjonen er et godt motivert valg.

Tids- og frekvensdomenet er et sånt domene, og det er vanligvis valgt i lydsignalprosessering.

Vi ønsker imidlertid å legge til de ekstra funksjonene til domenenavnsystemet med DNS-informasjon om vertsdatamaskinen for å kommentere den fullstendige stedsrepresentasjonen med den unike tid-rom-frekvensdomenerepresentasjon av hele lydsignalet i Multiple-Location Audio Recording, motivasjonen i denne oppgaven.

"Å fremføre et dataprogram" er å gjøre et lydopptak av et direktesendt radioprogram på en datamaskin ved hjelp av et direktesendt radioprogram og et lydopptaksprogram utviklet i programmeringsspråket C på en datamaskin.

Radiobølgene resonneres og beskrives logisk ved hjelp av dataprogrammene som kan kompileres i GCC og utføres på en datamaskin med GNU/Linux i begge oppgavene mine.

Opptak av Internet-radio forklarer jeg logisk i en Bachelor-oppgave gjennom radioprogrammet GNOME Internet Radio Locator (http://www.gnomeradio.org/) implementert for datamaskiner i programmeringsspråket C (http://www.oleaamot.no/omu/bachelor/Aamot,2020.pdf) og i en Master-oppgave (http://www.oleaamot.no/uio/master/Aamot,2022.pdf) gjennom ly-

dopptaksprogrammet GNOME Gingerblue (http://www.gingerblue.org/) implementert for datamaskiner i programmeringsspråket C for det grafiske skrivebordsmiljøet GNOME (http://www.gnome.org/).

Takk til Professor Sverre Holm ved Fysisk institutt og Dr. Wolfgang Leister ved Norsk Regnesentral som motiverte meg til å skrive dataprogrammene GNOME Internet Radio Locator (http://www.gnomeradio.org/) og GNOME Gingerblue (http://www.gingerblue.org/) etter at jeg fulgte seminarene Multimedia Coding and Transmission og Multimedia Coding and Applications ved Norsk Regnesentral (http://www.nr.no/) i 2004 som senere ble til INF5080 og INF5081 ved Institutt for informatikk (http://www.ifi.uio.no/) ved Universitetet i Oslo (http://www.uio.no/).

Jeg ønsker helt til slutt å takke førsteamanuensis Arnt Inge Vistnes som har hjulpet meg på veien som Bachelor-student som foreleser om svingninger og bølger i FY-ME100 våren 2002 ved Fysisk Institutt ved Universitetet i Oslo, Tarald Rørvik for interesse og oppmuntring, og min utholdende og alltid oppofrende, kjærlige Mor Gunhild Humblen og Far Helge Aamot.

Ole Kristian Aamot, Oslo, 1. januar 2022

Preface

In the first Multiple-Location Audio Recording Software implementation, the Free Software application GNOME Gingerblue version 3.0.1, as a free purpose application, we can reproduce hearable sounds for human listening with time-space-frequency notation.

We use the principles in the processing of signals that are motivated by the processes involved in hearing.

A representation of audio signals where we have access to both time and frequency information is a well-motivated choice. The time-frequency domain is such a domain, and it is commonly deployed in audio processing.

However, we want to add the extra capabilities of the Domain Name System information to annotate the full location representation with the unique time-space-frequency domain representation of the full audio signal in Multiple-Location Audio Recording, the motivation in this thesis.

Part I

Introduction

Chapter 1

Background

Communication in modern day society has been greatly enhanced by mans ability to reproduce sound. Inventions such as telegraph, telephone, phonograph, gramophone, radio, and later, television have benefited from the basic concept of reproduction and preservation of the human voice. The act of recording therefore is best comprehended within the context of broadcasting, telecommunication, and entertainment. (Nmungwun, 1989)

The medium of recording rely on two components that have been the very essence of the recording technology - magnetism and electricity.

Part II

The project

Chapter 2

Planning the project

GNOME Gingerblue 3.0.1 (Draft)

Ole Kristian Aamot

1 January 2022

Multiple-Location Audio Recording 1.0

Chapter 3

Audio Recording

3.1 History of Audio Recording

The first recorded attempt to simulate the human voice is a truely legendary statue of Memnon at Thebes, which dates back to the 18th Egyptian Dynasty, in 1490 B.C. It was claimed that one of the two remaining statues located on the west bank of the Nile made sounds at dawn.

Easily dismissed as myth, most visitors to the statues were convinced otherwise by the cuneiform writing inscribed on the base of the statue by famous early travellers.

An earthquake tumbled the statue in 27 B.C. Several notable travelers who visited it gave conflicting accounts of their witness.

Up to the end of the 1700s, scientists had fruitlessly worked to establish a relationship between electricity and magnetism.

In 1820, Hans Christian Ørsted, a professor at University of Copenhagen, discovered, as mentioned in the 200 year later non-peer-reviewed article "Radio flux in GNOME Radio Fields confirmed", Aamot, Oslo Metropolitan University, 2020 – DOI: 10.13140/RG.2.2.17889.33124 – http://www.gnomeradio.org/~ole/Aamot-2020.pdf) and the Bachelor thesis in Electrical Engineering ("Public Internet Radio Client for Accessing Free Audio Maps in Countries with Free Speech", Aamot, Oslo Metropolitan University, 2020 – DOI: 10.13140/RG.2.2.31344.17922 – http://www.gnomeradio.org/~ole/thesis.pdf), that when an electric current is passed through a wire held horizontally above a magnetic needle that is parallel to it, the needle is deflected, positioning itself at right angles with the conducting wire to the end of the positive pole of the magnet.

A wire that has a constant source of electricity passed through it becomes practically a magnet.

The tin-foil phonograph was discovered accidentally by Edison. While busy experimenting on a telegraphic machine (intented to repeat Morse characters recorded on paper by indentations that transferred messages to

another circuit automatically, he stumbled upon the idea that resulted in the phonograph.

In examining the indented paper, Edison noticed the speed at which it moved, and a humming noise that amanated from the indentation. This sound was a severe rhythm almost identical to human speech hear faintly.

In order to decipher this sound, Edison fitted a diaphragm to the machine. This also acred to amplify the sound. It was then obvious that the problem of recording human speeches and reproducing them by mechanical means was solved.

Edison proceeded to develop a machine exclusively for capturing the vibrations of the human voice as well as repeating them at a latter time. The machine was christened the "phonograph" (see Fig 1.). In November 1877, Edison officially announced his invention and on December 24, 1877, he filed a patent application for the phonograph with the U.S. Patent Office. This was duly approved as patent number 200,521, issued on February 19, 1878, one century minus one day before February 20, 1978 (my birth date).

The tin-foil phonograph was built by John Kruesi, who had worked with Edison for several years. Edison had only given a rough sketch of the phonograph to Kruesi, explaining what its functions were to be. It was a cylinder machine, with the cylinder covered with tin-foil for recording purposes. When Kruesi concluded work on the machine and brought it to Edison, he set it in motion and spoke into it:

"Mary had a little lamb, It's fleece was white as snow. And everywhere that Mary went, The lamb was sure to go."

When rewound, his exact words in clear tones were repeated, contrary to the hoarse murmur that he anticipated, Edison was baffled at the performance of the little machine.

Professor Joseph Henry (1797-1878) was a professor of physics at Albany Institute whose work integrated the principles that are so much inevitable in modern day electronics including phonographs, radio, television and hi-fi in relation to electricity, magnetism and mechanical energy.

Henry's theory was the basis for Morse's telegraph, Bell's telephone and other modern sound-producing mechanisms.

His principles enabled Valdemar Poulsen to record the first sound on a magnetized steel wire.

3.1.1 Audio Magnetic Recording

The technique of audio magnetic recording has been understood ever since the late 1800s and the essentials of the design problem was understood by the early 1900s.

Magnetic recording is inscribed in the general concensual view of this

history as a post-World War II innovation that was pioneered in the Third Reich.

The radio was a major propaganda tool for the Nazis. Broadcast quality was given top priority by the German government, RRG, and manufacturers of magnetic recorders. The research efforts of von Braunmühl and Weber resulted in the first high frequency recording in Germany in 1939-1940.

Between 1941 and 1945 the German Nazis killed 6 million European Jews in the Holocaust, the systematic, state-sponsored persecution and millions others murdered in concentration camps and gas chambers.

Antisemitism was at the foundation of the Holocaust. Antisemitism, the hatred of or prejudice against Jews, was a basic tenet of Nazi ideology. This prejudice was also widespread throughout Europe. Nazi Germany's persecution of Jews evolved and became increasingly more radical between 1933 and 1945. This radicalization culminated in the mass murder of six million Jews. During World War II, Nazi Germany and its allies and collaborators killed nearly two out of every three European Jews using deadly living conditions, brutal mistreatment, mass shootings and gassings, and specially designed killing centers.

I.B.M. took part in the extermination of Jews by providing computers and punched cards that provided information about the addresses and religion of the percecuted humans.

In 2015 at the Computer History Museum in California, I was a witness of this, and saw and read more at the exhibition about I.B.M. in the safe haven U.S.A.

The U.S. President Franklin D. Roosevelt stood against this and his famous 'fireside chats' extolling the policies of the New Deal were intimate in tone and attuned to a liberal, democratic cause whereas Nazi propaganda on German radio was strident and oratorical. As Kate Lacey describes, "Nazi program directors initiailly swamped the airwaves with bombastic music, live transmissions from Party events, and most ineffectively, declamatory speeches and dogmatic hyperbole." (Scannell and Cardiff, A Social History, 156).

Under the Nazis the declamatory style was unsuccessful and the speeches of Hitler were replaced by a "mixture of light music and intimate studio conversation".

In 1918 a Californian, Leonard F. Fuller, had proposed the Telegraphone wire recorder. (BIOS, 1961)

In 1927, two U.S. Navy Research Laboratories staff members were granted a U.S. patent for their invention, which involved the application of high frequency (A.C.) bias to steel wire to enhance sound reproduction.

Another Californian, James H. Alverson, proposed the use of radio frequency to saturate steel wire in magnetic recording. It was also apparent that research on A.C. bias, and its use, was done under Kenzo Nagai in Japan in the 1930s.

It was absolutely impossible in many occasions for the Allies to determine Hitler's location because live quality broadcasts of his speech would be transmitted simultaneously from all corners of Germany.

In addition, time delay broadcasts had been standard procedure in Germany since the mid-1920s and early 1930s. To make things even worse, the Nazi equipment were being deployed to deliberately confuse the Allies as to the location of high Nazi officials.

The B.B.C. broadcast filled the bill until midnight, when they left the air. Fishing around the dial in search of further entertainment, I soon discovered that the German stations apparently were on the air twenty-four hours a day. They broadcasted symphony concerts in the middle of the night - music that was very well played, and obviously by very large orchestras.

We know what canned music sounds like. The American network wouldn't permit the use of recordings in the early 1940s, because they claimed the quality was inferior.

In Germany in the 1940s and in USA at the stage before 2021, of course Hitler and Trump could have everything he wanted. If he wanted a full symphonic orchestra to play all night long, he could get it. Still, it didn't seem very likely that a madman like Hitler or Trump would insist on live concerts night after night.

Germany lost the war even with the help of radio and magnetic band recorders and United States and United Kingdom, Norway and France won the war against The Third Reich in Germany that was responsible for the killings of 6 Million Jews.

Three Bachelor of Science Theses written on the subject at Massachusetts Institute of Technology in 1938 testify that magnetic recording generated much curiosity in the late 1930s, especially in academical circles.

By the end of October 1939 the situation had improved, with a reduction in the use of gramophone records and more variety.

The development of radio news from Dunkirk to the end of the war can be thought of in two parts. The period up to D-Day and the invasion of France in June 1944, and the rest of the war, which was then dominated in news terms by the BBC's War Report, which provided a day-by-day account of the final year of the war and eventual Allied victory. News can probably claim to be the most innovative and successful part of BBC output at that time: "the BBC News Department ended the war with the most enhanced reputation and changed role of any wartime BBC Department...". It began the war with just two reports and recording equipment that required a six-ton van and had a top speed of 20 miles per hour and ended it with coordinated coverage of D-Day, "a superb journalistic achievement", with 19 reporters using portable disc recorders and live relays heard by an audience that reached 18 million. There are different components of this great transformation and these include not only improved recording technology but the creation of a News Division,

incorporating home and empire News and Talks, under A. P. Ryan in September 1942.

It was not until April 1946 that WMAQ, (an NBC affiliate in Chicago) aired the first completely wire-recorded news program, followed by a competitor, WBBM (a CBS Chicago affiliate), which also deployed wire recording for both spot-reporting and news events.

The precendent set, most network and local stations proceeded to record their news programs on wire recorders.

In 1951, while still enjoying the fortune magnetic tape recording implemented in audio and data recording to the recording of television signals.

In 1951 all sound recording was on disc, but in 1952 there were six EMI Midget recorders at Brodcasting House.

Ray Dolby, (later of audio noise reduction fame) was an exceptionally brilliant 19-year-old high school graduate who had enrolled as an engineering freshman at Stanford University. Dolby dropped out of college to join the Ampex team in August 1952.

Allthough Dolby lacked the necessary academic training in engineering, his ingenuity and understanding of technical matters made his contributions in the Ampex television recording project invaluable. It was Dolby who created the basic block diagram of VTR circuitry that is still implemented in the most recorders today.

As promising as the early efforts were, the project was again suspended in June 1953. In the midst of the frustration, Dolby, who had dropped out of Stanford, was drafted into the U.S. Army and despite friutless pleas by his colleagues he left sadly on March 18, 1953.

While he was in the Armed Forces, Dolby exchanged notes with Ginsburg. During the project's period of official suspenson (June 1953 through August 1954), considerable progress was made on the VTR project despite the few man hours and the little financial allotment assigned it, both by authorized and unauthorized means.

By 1955 tape had largely replaced the disc. The impact of tape recording on early current affairs broadcasting was slow to have effect but it had the potential to solve many "supply" problems.

3.2 History of Computing

The Internet was not the creation of a single person.

It was the product of engineers and inventors, researchers and programmers, and many more. Internet prehistory was an age of ideas, and many thinkers contributed visionary dreams that shaped what the Internet could and would come to be.

Here are some of the pioneers:

3.2.1 Vannevar Bush and his "memex"

Vannevar Bush was a professor in the Department of Electrical Engineering at MIT, an influential policymaker and head of the Carnegie Foundation, and a presidential science advisor who pushed for government support for science.

A prolific inventor, he also designed and constructed the Differential Analyzer, a mechanical yet sophisticated calculating device.

His 1945 "memex" idea foresaw how computing would allow humanity better access to information.

3.2.2 Claude Shannon and information theory

Claude Shannon is regarded as the father of information theory. A 1930s graduate of MIT's Master of Science and Ph.D. program, Shannon assisted researchers with the Bush Differential Analyzer while he completed his studies. His astonishing realization that information of any kind could be expressed mathematically in bits–represented by a single zero or one–formed the basis for digital computing.

3.2.3 J.C.R. Licklider and networks

J.C.R. Licklider was associated with MIT and MIT's Lincoln Laboratory for more than thirty years.

His articles "The Computer as a Communication Device" (1968) and "Man-Computer Symbiosis" (1960) described how interactive, networked computers could be used for human communication, and predicted many uses of the modern Internet. Licklider's ideas and leadership led to the ARPANET (Advanced Research Projects Agency Network), an early computer network that was the original Internet.

3.2.4 Bardeen/Shockley/Brattain and the transistor

In 1956 Bell Labs scientists John Bardeen, William Shockley and Walter Brattain shared the Nobel Prize in physics for their invention of the transistor, a major payoff of the wartime semiconductor work, according to Robert Buderi's book "The INVENTION That CHANGED the WORLD" (Simon Schuster, New York, 1996).

The three met just after World War II, when Bell Labs charged Shockley with the job of building a solid state amplifier.

The Magic Month, actually a five-week span that saw the birth of the transistor and the genesis of two Nobel Prizes, opened on November 17, 1947.

Walter Brattain had been purusing the team's goal of building the base of fundamental knowledge and testing the surface-state theory.

On November 22, 1947, the Saturday before Thanksgiving, John Bardeen summarized much of the work while filling seven pages in his notebook. He concluded, "...these tests show definitely that it is possible to introduce an electrode or grid to control the flow of current in a semiconductor."

In December 1956 the Nobel Prizes in Physics were granted to the Bell Labs colleagues Bardeen, Brattain and Shockley.

3.2.5 Tim Berners-Lee and the Web

In 1989 Tim Berners-Lee, Professor at MIT's Computer Science and Artificial Intelligence Laboratory, invented the World Wide Web at the CERN Lab in Switzerland and directed his work toward the W3C Consortium (http://www.w3.org/), the Web Research Institute and the World Wide Web Foundation.

These organizations study the future use and design of the Web, make recommendations about technical standards, and implement projects designed to realize the full potential of the World Wide Web.

3.2.6 Philippe Defert and httpd

Philippe Defert (1954 - 2013) working at the CERN IT Department in Switzerland made it possible through his work on the httpd server to publish widespread information to reach millions of people like AM/FM radio previously did, but without global censorship, except by ICANN.org, by decentralized domain name registrars and individual domain holders.

3.3 History of Domain Name System

Paul Mockapetris expanded the Internet beyond its academic origins by inventing the Domain Name System (DNS) in 1983.

Previously computers connected to the Internet were addressed with IP addresses, not resolvable by domain names.

But the invention of the DNS in 1983 and the original Internet Standards in 1986 after the creation of the Internet Engineering Task Force IETF made this possible.

The two documents that marked the start are RFC 1034 and RFC 1035. They describe the whole protocol functionality and include data types that it can carry.

The latest version of the Internet software Berkeley Internet Name Domain 9 ("bind" and "named") by the Internet Software Consortium helped bring the Domain Name System to the entire world in 2000.

Chapter 4

Hardware

Legitimate audio can be originated in a number of ways. Until recent advances in digital technology, a musician's options were limited to getting low-grade audio from distant sources or filing reports on location over ordinary telephones, or travelling to and from the location with a microphone, lead and portable tape recorder, most commonly a Uher. Despite the great weight of the Uher, whose strap gave so many reporters shoulder strain, it is fondly remembered for the acceptable compromise it represented between ease of use, broadcast quality and portability. But many have welcomed the later generations of hardware, including Digital Audio Tape (DAT), MiniDisc and hard disk recorders. For ease of use, portability and concealability (in situations where, for reasons of personal safety, musicians might prefer to blend into their surroundings), the recent developed microphones that record on to a chip housed within their own stem, and an associated USB port for downloading the audio as data into any computer, are attrative alternatives.

Digital technology has also facilitated the establishment of live connections between the news desk and remote locations. Expensive, fixed, broadcast-quality analogue landlines and often hard-to-set-up VHF radio links from outside broadcast vehicles have, for the moment, been eclipsed by ISDN lines, which deploy digital/analogue converters (codecs), satellite uplinking and the Internet.

Chapter 5

Software

5.1 World Wide Web

The release of Apache HTTP Server (http://httpd.apache.org/) by the Apache Software Consortium and the release of PHP Programming Language (https://www.php.net) and Wordpress Weblog Software (https://www.wordpress.org/) is essential in human's ability to reach millions of people on the World Wide Web.

5.2 GNOME

The release of the GNOME desktop software in 1997 made it possible for humans to interactively access and store information on a computer.

5.3 GStreamer, PipeWire and WirePlumber

The release of the GStreamer software in 2004 marked the future for multimedia on GNU/Linux and other desktop platforms running GNOME.

Control flow in the program is determined by conditional statements. The outcome of such tests controls the further flow of the program.

GStreamer is the software for audio recording and playback, signaling and control in Free Desktops such as GNOME.

For example, by using conditional statements, control function values can be reset and instruments can turn themselves on or off or be instructed to influence one another.

5.4 GNOME Gingerblue

GNOME Gingerblue completes the task of recording live audio streams on any computer that runs a Unix-compatible GNU system with the

Linux kernel or a Apple macOS system with Macports.org and lets you recording/download audio into a laptop that can be edited on the site, saved as an Ogg Vorbis with XML meta data information, and perhaps using other developments in mobile phone technology, sent over the Internet in a fraction of the time it once took musicians to return to base and then edit it using traditional techniques.

GNOME Gingerblue 3.0.1 is available and builds/runs on GNOME 41 systems such as Fedora Core 35.

It supports immediate, live audio recording in compressed Xiph.org Ogg Vorbis encoded audio files stored in the private $*HOME/Music/* directory from the microphone/input line on a computer or remote audio cards through USB connection through PipeWire (http://WWW.PIPEWIRE.ORG/) with GStreamer (http://GSTREAMER.FREEDESKTOP.ORG/) on Fedora Core 34 (https://GETFEDORA.ORG/).

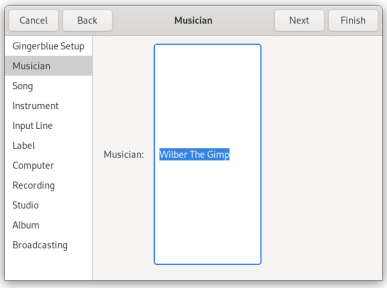
In GNOME Gingerblue version 3.0.1, the first implementation of Multiple-Location Audio Recording, as published in the thesis, audio and control rates are implemented by separate loops.

When composing with the computer, sounds are recorded digitally with a sampling rate of at least 40,000 Hz and an amplitude resolution of at least 16 bits.

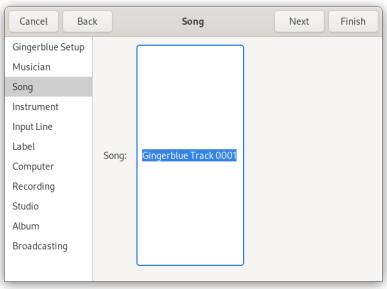
The audio signals recorded with GNOME Gingerblue version 3.0.1 have a sample rate of 44,100 Hz.

See the GNOME Gingerblue project (https://WWW.GINGERBLUE.ORG/) for screenshots, Fedora Core 34 x86_64 RPM package and GNU autoconf installation package (https://DOWNLOAD.GNOME.ORG/sources/gingerblue/3.0/gingerblue-3.0.1.tar.xz) for GNOME 41 systems and https://GITLAB.GNOME.ORG/ole/gingerblue.git for the GPLv3 source code in my GNOME Git repository.

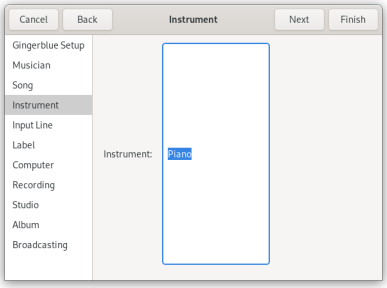
Gingerblue music recording session screen. Click “Next” to begin session.



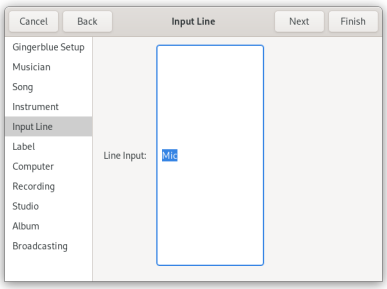
The default name of the musician is extracted from g_get_real_name(). You can edit the name of the musician and then click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip the details).



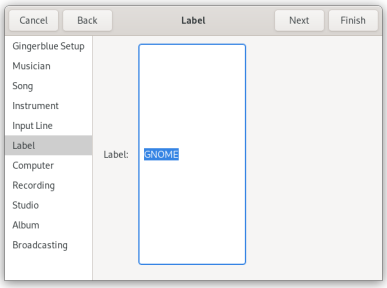
Type the name of the musical song name. Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip any of the details).



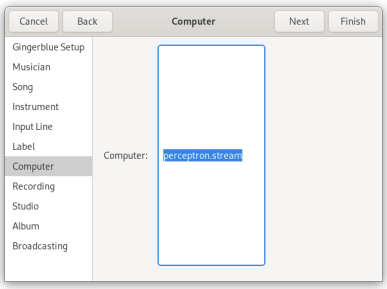
Type the name of the musical instrument. The default instrument is “Guitar”. Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip any of the details).



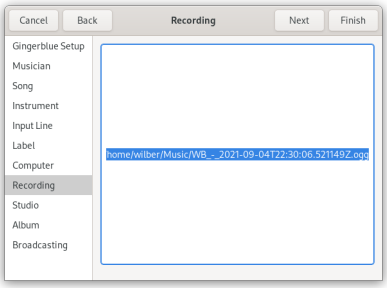
Type the name of the audio line input. The default audio line input is “Mic” (gst_pipeline_new("record_pipe") in GStreamer). Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip the details).



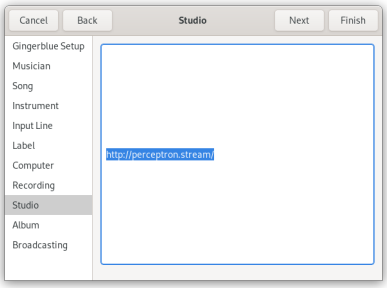
Enter the recording label. The default recording label is “GNOME” (Free label). Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip the details).



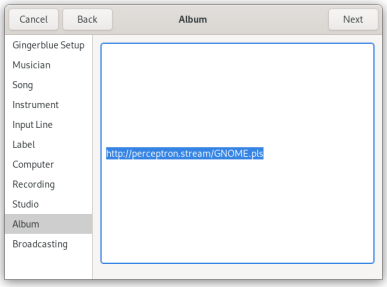
Enter the Computer. The default station label is a Fully-Qualified Domain Name (g_get_host_name()) for the local computer. Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip the details).



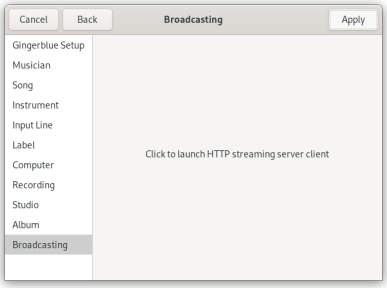
Notice the immediate, live recording file. The default immediate, live recording file name falls back to the result of g_strconcat(g_get_user_special_dir(G_USER_DIRECTORY_MUSIC), "/", gtk_entry_get_text(GTK_ENTRY(musician_entry)), "_-_", gtk_entry_get_text(GTK_ENTRY(song_entry)), "_[",g_date_time_format_iso8601 (datestamp),"]",".ogg", NULL) in gingerblue/src/gingerblue-main.c



Studio configuration resolves the server address of your local computer.



Album configuration is the playlist of the compilation of multiple audio files.



Broadcasting to the World Wide Web (a Wordpress Webblog installation) is the step after recording your audio files.

Click on “Cancel” once in GNOME Gingerblue to stop immediate recording and click on “Cancel” once again to exit the application (or Ctrl-c in the terminal).

The following Multiple-Location Audio Recording XML file [.gingerblue] is created in G_USER_DIRECTORY_MUSIC (usually $*HOME*/*Music*/ on American English systems):

```
<?xml version='1.0' encoding='UTF-8'?>
<gingerblue version='3.0.1'>
  <musician>Wilber</musician>
  <song>Gingerblue Track 0001</song>
  <instrument>Piano</instrument>
  <line>Mic</line>
  <label>GNOME Music</label>
  <station>streaming.gnome.org</station>
  <filename>/home/wilber/Music/Wilber_-_Song_-_2021-07-12T21:36:07.624570Z.ogg</filename>
</gingerblue>
```

You'll find the recorded Ogg Vorbis audio files along with the Multiple-Location Audio Recording XML files in g_get_user_special_dir(G_USER_DIRECTORY_MUSIC) (usually $*HOME*/*Music*/) on GNOME 41 systems configured in the American English language.

In GNOME Gingerblue version 3.0.1, the first implementation of Multiple-Location Audio Recording, as published in the thesis, audio and control rates are implemented by separate loops.

When composing with the computer, sounds are recorded digitally with a sampling rate of at least 40,000 Hz and an amplitude resolution of at least 16 bits.

The audio signals recorded with GNOME Gingerblue version 3.0.1 have a sample rate of 44,100 Hz and are stored in the $*HOME*/*Music*/ folder.

Chapter 6

Internet

6.1 Apache HTTP

The Apache HTTP server is available for Unix-platforms such as Debian, Fedora and Ubuntu from (http://httpd.apache.org/

6.2 PHP

The programming language PHP is available from http://www.php.net/ and is availble as a Apache module for the Apache HTTP Server http://httpd.apache.org/.

6.3 Wordpress

The World Wide Web Blog software Wordpress is available from http://www.wordpress.org/

Chapter 7

References

7.1 Audio Engineering

7.2 Electrical Engineering

7.3 Software Engineering

Chapter 8

Source

8.1 gingerblue 3.0.1

8.1.1 gingerblue/src/gingerblue-chord.h

```
/* $Id$

   Copyright (C) 2018-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 3.0.1 (2021-11-22)
   Website: http://www.gingerblue.org/

 */

#ifndef _GINGERBLUE_CHORD_H_
#define _GINGERBLUE_CHORD_H_ 1

typedef struct _GingerblueChord GingerblueChord;

struct _GingerblueChord {
    char *root;
    char *file;
    /* struct Display *gui; */
    char e1;
    char b2;
    char g3;
    char d4;
    char a5;
    char e6;
};

struct _GingerblueChord gbc[] = {
```

```
29    {"C",     "Gingerblue_Guitar_C.wav",  /* &console, */ '0', '1',
          '0', '2', '3', '0'},
30    {"C#",    "Gingerblue_Guitar_C#.wav", /* &console, */ '1', '2',
          '1', '3', '4', '0'},
31    {"Db",    "Gingerblue_Guitar_Db.wav", /* &console, */ '1', '2',
          '1', '3', '4', '0'},
32    {"D",     "Gingerblue_Guitar_D.wav",  /* &console, */ '1', '2',
          '1', '0', '0', '0'},
33    {"D#",    "Gingerblue_Guitar_D#.wav", /* &console, */ '3', '4',
          '2', '1', '0', '0'},
34    {"Eb",    "Gingerblue_Guitar_Eb.wav", /* &console, */ '3', '4',
          '2', '1', '0', '0'},
35    {"E",     "Gingerblue_Guitar_E.wav",  /* &console, */ '0', '0',
          '1', '2', '2', '0'},
36    {"F",     "Gingerblue_Guitar_F.wav",  /* &console, */ '1', '1',
          '2', '3', '3', '1'},
37    {"F#",    "Gingerblue_Guitar_F#.wav", /* &console, */ '2', '2',
          '3', '4', '4', '1'},
38    {"Gb",    "Gingerblue_Guitar_Gb.wav", /* &console, */ '2', '2',
          '3', '4', '4', '1'},
39    {"G",     "Gingerblue_Guitar_G.wav",  /* &console, */ '3', '0',
          '0', '0', '2', '3'},
40    {"G#",    "Gingerblue_Guitar_G#.wav", /* &console, */ '0', '1',
          '1', '1', '3', '4'},
41    {"Ab",    "Gingerblue_Guitar_Ab.wav", /* &console, */ '0', '1',
          '1', '1', '3', '4'},
42    {"A",     "Gingerblue_Guitar_A.wav",  /* &console, */ '0', '2',
          '2', '2', '0', '0'},
43    {"A#",    "Gingerblue_Guitar_A#.wav", /* &console, */ '1', '3',
          '3', '3', '1', '0'},
44    {"Bb",    "Gingerblue_Guitar_Bb.wav", /* &console, */ '1', '3',
          '3', '3', '1', '-'},
45    {"B",     "Gingerblue_Guitar_B.wav",  /* &console, */ '2', '4',
          '4', '4', '2', '0'},
46    {"Bm",    "Gingerblue_Guitar_Bm.wav", /* &console, */ '2', '3',
          '4', '4', '2', '-'},
47    {NULL,NULL}
48  };
49
50  #endif /* _GINGERBLUE_CHORD_H_ */
```

8.1.2 gingerblue/src/gingerblue-config.h

```
1  #ifndef GINGERBLUE_CONFIG_H
2  #define GINGERBLUE_CONFIG_H 1
3
4  GtkWidget *main_config (GtkWidget *widget, gpointer *
       location_data);
```

```
5
6 #endif /* GINGERBLUE_CONFIG_H */
```

8.1.3 gingerblue/src/gingerblue-file.h

```
1  /* $Id$
2
3     Copyright (C) 2018-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 #ifndef _GINGERBLUE_FILE_H_
12 #define _GINGERBLUE_FILE_H_ 1
13
14 #include <libxml/xmlmemory.h>
15 #include <libxml/parser.h>
16
17 GingerblueData *gb_file_config_load (GingerblueData *head, gchar
       *filename);
18
19 static void gb_file_parse_volume (GingerblueData *data,
      xmlDocPtr doc, xmlNodePtr cur);
20
21 #endif /* _GINGERBLUE_FILE_H_ */
```

8.1.4 gingerblue/src/gingerblue.h

```
1  /* $Id$
2
3     Copyright (C) 2018-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9   */
10
```

```
11 #ifndef _GINGERBLUE_H_
12 #define _GINGERBLUE_H_ 1
13
14 #include <gtk/gtk.h>
15
16 #define GINGERBLUE_STUDIO_PLAYER_TRUE 1
17 #define GINGERBLUE_STUDIO_PLAYER_FALSE 0
18
19 typedef struct _GingerblueData GingerblueData;
20
21 struct _GingerblueData {
22     GtkWidget *knob;
23     gint player_status;
24     gchar *line;
25     gint jack;
26     gchar *label;
27     gboolean lpf;
28     gboolean hpf;
29     gchar *musician;
30     gchar *musical_instrument;
31     gchar *version;
32     gchar *volume;
33     GingerblueData *next;
34     GingerblueData *prev;
35     GtkWidget *window;
36     GMainLoop *player_loop;
37 };
38
39 void gb_window_break_record (GtkButton *record, GtkVolumeButton
       *volume);
40 void gb_window_pause_record (GtkButton *record, GtkVolumeButton
       *volume);
41 GingerblueData *gb_window_new_record (GtkButton *record,
       GtkVolumeButton *volume);
42 GingerblueData *gb_window_store_volume (GtkButton *record,
       GtkVolumeButton *volume);
43 gdouble gb_window_set_volume (GtkVolumeButton *volume, gdouble
       value);
44 gdouble gb_window_new_volume (GtkVolumeButton *volume, gchar *
       msg);
45 gdouble gb_window_get_volume (GtkVolumeButton *volume);
46
47 gint gb_exit (void);
48
49 #endif /* _GINGERBLUE_H_ */
```

8.1.5 gingerblue/src/gingerblue-knob.h

```
1  /* $Id$
2
3     Copyright (C) 2018-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 GtkWidget *knob (GingerblueData *data, GtkWidget *line, gint
       jack, gchar *label, gboolean lpf, gboolean hpf);
```

8.1.6 gingerblue/src/gingerblue-line.h

```
1  /* $Id$
2
3     Copyright (C) 2018-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 GtkWidget *line (gint jack, gchar *label);
```

8.1.7 gingerblue/src/gingerblue-main.h

```
1  /* $Id$
2
3     Copyright (C) 2018-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 GtkAssistantPageFunc gb_assistant_cb (GtkAssistant *assistant,
       GDateTime *datestamp);
12 static void gb_record_cb (char *path, gpointer data);
```

8.1.8 gingerblue/src/gingerblue-main-loop.h

```
1  #ifndef GINGERBLUE_MAIN_LOOP_H
2  #define GINGERBLUE_MAIN_LOOP_H 1
3
4  GtkWidget *gingerblue_main_loop (GingerblueData *gingerblue);
5
6  #endif /* GINGERBLUE_MAIN_LOOP_H */
```

8.1.9 gingerblue/src/gingerblue-record.h

```
1  /* $Id$
2
3     Copyright (C) 2018-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 #include <string.h>
12 #include <gst/gst.h>
13 #include <signal.h>
14 #include <unistd.h>
15 #include <stdlib.h>
16 #include <stdio.h>
17 #include <string.h>
18
19 #ifndef _GINGERBLUE_RECORD_H_
20 #define _GINGERBLUE_RECORD_H_ 1
21
22 static gboolean message_cb (GstBus * bus, GstMessage * message,
      gpointer user_data);
23 static GstPadProbeReturn unlink_cb(GstPad *pad, GstPadProbeInfo
      *info, gpointer user_data);
24 void stopRecording();
25 void startRecording();
26 int sigintHandler(int unused);
27
28 int gingerblue_record_begin();
29 int gingerblue_record_end();
30
```

```
31 typedef struct _GingerblueRecord {
32   gboolean recording_found;
33 } GingerblueRecord;
34
35 #endif /* _GINGERBLUE_RECORD_H_ */
```

8.1.10 gingerblue/src/gingerblue-song.h

```
1  /* $Id$
2
3     Copyright (C) 2018-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 #ifndef GINGERBLUE_SONG_H
12 #define GINGERBLUE_SONG_H 1
13
14 #include <libxml/xmlmemory.h>
15 #include <libxml/parser.h>
16
17 GtkWidget *gb_song_new (gchar *title);
18 GtkWidget *gb_song_quit (gchar *title);
19
20 #endif /* GINGERBLUE_SONG_H */
```

8.1.11 gingerblue/src/gingerblue-studio-config.h

```
1 #ifndef GINGERBLUE_STUDIO_CONFIG_H
2 #define GINGERBLUE_STUDIO_CONFIG_H 1
3
4 GtkWidget *main_studio_config (gchar *location_data, gchar *
     studio_city);
5
6 #endif /* GINGERBLUE_STUDIO_CONFIG_H */
```

8.1.12 gingerblue/src/gingerblue-studio-stream.h

```
1  /* $Id$
2
3     Copyright (C) 2018-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 #ifndef GINGERBLUE_STUDIO_STREAM_H
12 #define GINGERBLUE_STUDIO_STREAM_H 1
13
14 GtkWidget *main_studio_stream (gchar *location_data, gchar *
       studio_city);
15
16 #endif /* GINGERBLUE_STUDIO_STREAM_H */
```

8.1.13 gingerblue/src/gingerblue-app.c

```
1  /* $Id$
2
3     Copyright (C) 2020-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9   */
10
11 #include <glib/gi18n.h>
12 #include <gtk/gtk.h>
13 #include <gst/gst.h>
14 #include "gingerblue.h"
15 #include "gingerblue-config.h"
16 #include "gingerblue-main.h"
17 #include "gingerblue-main-loop.h"
18 #include "gingerblue-studio-config.h"
19
20 int main_app (gint argc, gchar *argv[]) {
21     GingerblueData *gingerblue_config;
```

```
22          GtkWindow *gingerblue_window;
23      gtk_init (&argc, &argv);
24          gingerblue_config = main_config (gingerblue_window, "
                studios.gingerblue.org");
25      gingerblue_window = gingerblue_main_loop (gingerblue_config)
            ;
26      gtk_widget_show_all (gingerblue_window);
27      gst_init(&argc, &argc);
28      gtk_main();
29      return (0);
30  }
```

8.1.14 gingerblue/src/gingerblue.c

```
1   /* $Id$
2
3      Copyright (C) 2018-2021 Aamot Software
4      Author(s): Ole Aamot <ole@gnome.org>
5      License: GNU GPL version 3
6      Version: 3.0.1 (2021-11-22)
7      Website: http://www.gingerblue.org/
8
9    */
10
11  #include <glib/gstdio.h>
12  #include <glib/gi18n.h>
13  #include <gst/gst.h>
14  #include <gtk/gtk.h>
15  #include "gingerblue.h"
16  #include "gingerblue-file.h"
17
18  gint
19  gb_exit (void) {
20      gst_deinit();
21      gtk_main_quit();
22  }
23
24  void
25  gb_window_break_record (GtkButton *record, GtkVolumeButton *
        volume) {
26      /* gtk_button_set_label(GTK_BUTTON (cue), "Continue
            Recording"); */
27      /* g_signal_connect (GTK_BUTTON (cue), "clicked", G_CALLBACK
             (gb_window_new_record), gingerblue_data->volume); */
28  }
29
30  void
```

```
31 gb_window_pause_record (GtkButton *record, GtkVolumeButton *
       volume) {
32     /* gtk_button_set_label(GTK_BUTTON (cue), "Continue
           Recording"); */
33     /* g_signal_connect (GTK_BUTTON (cue), "clicked", G_CALLBACK
            (gb_window_new_record), gingerblue_data->volume); */
34 }
35
36 GingerblueData *
37 gb_window_new_record (GtkButton *record, GtkVolumeButton *volume
       ) {
38     /* gtk_button_set_label(GTK_BUTTON (record), "Stop Recording
           "); */
39 }
40
41 GingerblueData *
42 gb_window_store_volume (GtkButton *record, GtkVolumeButton *
       volume) {
43     /* gtk_button_set_label(GTK_BUTTON (record), "Stop Recording
           "); */
44 }
45
46 gdouble
47 gb_window_set_volume (GtkVolumeButton *volume, gdouble value) {
48     gtk_scale_button_set_value (GTK_SCALE_BUTTON (volume), (
           gdouble) value);
49 }
50
51 gdouble
52 gb_window_new_volume (GtkVolumeButton *volume, gchar *msg) {
53     g_print ("New volume: %0.2f\n", (gdouble)
           gtk_scale_button_get_value (GTK_SCALE_BUTTON (volume)));
54     return (gdouble) gtk_scale_button_get_value (
           GTK_SCALE_BUTTON (volume));
55 }
56
57 gdouble
58 gb_window_get_volume (GtkVolumeButton *volume) {
59     return (gdouble) gtk_scale_button_get_value (
           GTK_SCALE_BUTTON (volume));
60 }
```

8.1.15 gingerblue/src/gingerblue-config.c

```
1 /* $Id$
2
3    Copyright (C) 2020-2021 Aamot Software
4    Author(s): Ole Aamot <ole@gnome.org>
```

```
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9   */
10
11  #include <config.h>
12  #include <glib/gi18n.h>
13  #include <gtk/gtk.h>
14  #include <gtk/gtkbox.h>
15  #include <gtk/gtkbutton.h>
16  #include <gtk/gtkcontainer.h>
17  #include <gtk/gtkwindow.h>
18
19  #include <gst/gst.h>
20  #include "gingerblue.h"
21  #include "gingerblue-main-loop.h"
22  #include "gingerblue-studio-config.h"
23  #include "gingerblue-studio-stream.h"
24  #include "gingerblue-studio-location.h"
25
26  extern GtkWidget *computer_entry;
27  extern GtkWidget *studio_entry;
28  extern GtkWidget *recording_entry;
29  extern GtkWidget *album_entry;
30
31  void studio_location_selected (GtkWidget *widget, gpointer *data
        )
32  {
33          g_print ("Selected␣studios\n");
34  }
35
36  GtkWidget *main_config (GtkWidget *widget, gpointer *
        location_data) {
37          GingerblueData *Gingerblue;
38          GtkButton *AddStudioButton;
39          GtkButton *NewStudioButton;
40          GtkBox *Studio;
41          GtkListBox *Location;
42          GtkListBoxRow *Computer;
43          GtkWidget *Studios;
44          GtkWidget *StudioLabel;
45          GtkContainer *Container;
46          GtkWindow *gingerblue;
47          gingerblue = gtk_window_new (GTK_WINDOW_TOPLEVEL);
48          gtk_window_set_title (GTK_WINDOW (gingerblue),
                g_strconcat(_("Recording␣("), gtk_entry_get_text(
                GTK_ENTRY(computer_entry)), _(")␣on␣"),
                gtk_entry_get_text(GTK_ENTRY(studio_entry)), _("␣("),
                PACKAGE_STRING, ")", NULL));
49          AddStudioButton = gtk_button_new_with_label(_("Add␣Studio
                "));
50          NewStudioButton = gtk_button_new_with_label(_("New␣Studio
                "));
```

```
51          Studio = gtk_box_new (GTK_ORIENTATION_VERTICAL, 8);
52          Location = gtk_list_box_new ();
53          Computer = gtk_list_box_row_new();
54          Studios = gtk_box_new(GTK_ORIENTATION_HORIZONTAL, 0);
55          gtk_container_add (GTK_CONTAINER (Computer), Studios);
56          StudioLabel = gtk_label_new (gtk_entry_get_text(GTK_ENTRY
                (computer_entry)));
57          gtk_container_add (GTK_CONTAINER (gingerblue), GTK_WIDGET
                 (Studio));
58          gtk_container_add (GTK_CONTAINER (Location), Computer);
59          gtk_box_pack_start (GTK_BOX (Studio), GTK_BUTTON (
                NewStudioButton), TRUE, TRUE, 0);
60          gtk_box_pack_start (GTK_BOX (Studios), StudioLabel, TRUE,
                 TRUE, 0);
61          g_signal_connect (GTK_BUTTON(AddStudioButton), "clicked",
                 G_CALLBACK(main_studio_config), gtk_entry_get_text(
                GTK_ENTRY(computer_entry)));
62          gtk_box_pack_start (GTK_BOX (Studio), GTK_LIST_BOX (
                Location), TRUE, TRUE, 0);
63          gtk_box_pack_start (GTK_BOX (Studio), GTK_BUTTON (
                AddStudioButton), TRUE, TRUE, 0);
64          fprintf(stdout, "%s\n", gtk_entry_get_text(GTK_ENTRY(
                gtk_list_box_get_selected_row(GTK_LIST_BOX(Location)))
                ));
65          g_signal_connect (GTK_LIST_BOX(Location), "row-selected",
                 G_CALLBACK(studio_location_selected),
                gtk_list_box_get_selected_row (GTK_LIST_BOX(Location))
                );
66          g_signal_connect (GTK_BUTTON(NewStudioButton), "clicked",
                 G_CALLBACK(studio_location_selected),
                gtk_entry_get_text(GTK_ENTRY(computer_entry)));
67          gtk_widget_show_all (GTK_WIDGET (gingerblue));
68          return (GtkWidget *) gingerblue;
69  }
```

8.1.16 gingerblue/src/gingerblue-file.c

```
1   /* $Id$
2
3      Copyright (C) 2018-2021 Aamot Software
4      Author(s): Ole Aamot <ole@gnome.org>
5      License: GNU GPL version 3
6      Version: 3.0.1 (2021-11-22)
7      Website: http://www.gingerblue.org/
8
9   */
10
11  #include <gst/gst.h>
```

```
12 #include <gtk/gtk.h>
13 #include <glib/gstdio.h>
14 #include <glib/gi18n.h>
15 #include <libxml/xmlmemory.h>
16 #include <libxml/parser.h>
17 #include "gingerblue.h"
18
19 GingerblueData *
20 gb_file_parse_volume (GingerblueData *data, xmlDocPtr doc,
       xmlNodePtr cur) {
21         GingerblueData *gbdata = (GingerblueData *)data;
22         xmlNodePtr sub;
23     gbdata->version = (gchar *)xmlGetProp (cur, (const xmlChar
           *)"version");
24     gbdata->volume = (gchar *)xmlGetProp (cur, (const xmlChar *)
           "volume");
25     sub = cur->xmlChildrenNode;
26     while (sub != NULL) {
27         if ((!xmlStrcmp
28               (sub->name, (const xmlChar *) "line"))) {
29             gbdata->line = (gchar *) xmlNodeListGetString(doc,
                   sub->xmlChildrenNode, 1);
30         }
31         if ((!xmlStrcmp
32               (sub->name, (const xmlChar *) "musician"))) {
33             gbdata->musician = (gchar *) xmlNodeListGetString(
                   doc, sub->xmlChildrenNode, 1);
34         }
35         if ((!xmlStrcmp
36               (sub->name, (const xmlChar *) "musical_instrument")
                   )) {
37             gbdata->musical_instrument = (gchar *)
                   xmlNodeListGetString(doc, sub->xmlChildrenNode,
                   1);
38         }
39         if ((!xmlStrcmp
40               (sub->name, (const xmlChar *) "volume"))) {
41             gbdata->volume = (gchar *) xmlNodeListGetString(doc,
                    sub->xmlChildrenNode, 1);
42         }
43         sub = sub->next;
44     }
45     return (GingerblueData *)gbdata;
46 }
47
48 GingerblueData *
49 gb_file_config_load (GingerblueData *head, gchar *filename) {
50     xmlDocPtr doc = NULL;
51     xmlNodePtr cur = NULL;
52     GingerblueData *curr = NULL;
53     gchar *version;
54     g_print ("%s\n", filename);
55     g_return_val_if_fail (filename != NULL, NULL);
56     doc = xmlReadFile (filename, NULL, 0);
```

```
57      if (doc == NULL) {
58          perror("xmlParseFile");
59          xmlFreeDoc (doc);
60          return NULL;
61      }
62      cur = xmlDocGetRootElement (doc);
63      if (cur == NULL) {
64              fprintf (stderr, _("Empty␣document\n"));
65          xmlFreeDoc (doc);
66          return NULL;
67      }
68      if (xmlStrcmp(cur->name, (const xmlChar *) "gingerblue")) {
69              fprintf(stderr, _("Document␣of␣wrong␣type,␣root␣node
                    ␣!=␣gingerblue\n"));
70          xmlFreeDoc (doc);
71          return NULL;
72      }
73      version = (gchar *) xmlGetProp (cur, (const xmlChar *)"
            version");
74      g_print (_("Valid␣GNOME␣Gingerblue␣%s␣XML␣document␣%s\n"),
            version, filename);
75      cur = cur->xmlChildrenNode;
76      while (cur != NULL) {
77        g_print (_("Parsing␣GNOME␣Gingerblue␣%s␣XML␣document␣%s\n"
              ), version, filename);
78          if ((!xmlStrcmp(cur->name, (const xmlChar *) "line"))) {
79            g_print (_("Found␣Line\n"));
80              curr = g_new0(GingerblueData, 1);
81              curr->line = (gchar *) xmlNodeListGetString(doc, cur
                    ->xmlChildrenNode, 1);
82              g_print ("%s\n", curr->line);
83              // curr = gb_file_parse_volume (curr, doc, cur);
84              curr->next = head;
85              head = curr;
86              /* mem_volume = head */
87              /* volumes = g_list_append (gingerblue_volumes, (
                    GingerblueData *)mem_volume); */
88              g_print ("Done␣with␣parsing␣Line\n");
89          }
90          if ((!xmlStrcmp(cur->name, (const xmlChar *) "musician")
                )) {
91              g_print (_("Found␣Musician\n"));
92              curr = g_new0(GingerblueData, 1);
93              curr->musician = (gchar *) xmlNodeListGetString(doc,
                     cur->xmlChildrenNode, 1);
94              g_print ("%s\n", curr->musician);
95              // curr = gb_file_parse_volume (curr, doc, cur);
96              curr->next = head;
97              head = curr;
98              /* mem_volume = head */
99              /* volumes = g_list_append (gingerblue_volumes, (
                    GingerblueData *)mem_volume); */
100             g_print (_("Done␣with␣parsing␣Musician\n"));
101         }
```

```
102         if ((!xmlStrcmp(cur->name, (const xmlChar *) "
                musical_instrument"))) {
103             g_print (_("Found␣Musical␣Instrument\n"));
104             curr = g_new0(GingerblueData, 1);
105             curr->musical_instrument = (gchar *)
                    xmlNodeListGetString(doc, cur->xmlChildrenNode,
                    1);
106             g_print ("%s\n", curr->musical_instrument);
107             // curr = gb_file_parse_volume (curr, doc, cur);
108             curr->next = head;
109             head = curr;
110             /* mem_volume = head */
111             /* volumes = g_list_append (gingerblue_volumes, (
                    GingerblueData *)mem_volume); */
112             g_print (_("Done␣with␣parsing␣Musical␣Instrument\n")
                    );
113         }
114         if ((!xmlStrcmp(cur->name, (const xmlChar *) "volume")))
                {
115             g_print (_("Found␣Volume\n"));
116             curr = g_new0(GingerblueData, 1);
117             curr->volume = (gchar *) xmlNodeListGetString(doc,
                    cur->xmlChildrenNode, 1);
118             g_print ("%s\n", curr->volume);
119             // curr = gb_file_parse_volume (curr, doc, cur);
120             curr->next = head;
121             head = curr;
122             /* mem_volume = head */
123             /* volumes = g_list_append (gingerblue_volumes, (
                    GingerblueData *)mem_volume); */
124             g_print (_("Done␣with␣parsing␣Volume\n"));
125         }
126         cur = cur->next;
127     }
128     g_print (_("Finished␣parsing␣XML␣document\n"));
129     xmlFreeDoc (doc);
130     return curr;
131 }
132
133 /* int main (int argc, char **argv) */
134 /* { */
135 /*  GingerblueData *data = NULL; */
136 /*  data = gb_file_config_load (data, "gingerblue.xml"); */
137 /*  free (data); */
138 /*  return (0); */
139 /* } */
```

8.1.17 gingerblue/src/gingerblue-knob.c

```
/* $Id$

   Copyright (C) 2018-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 3.0.1 (2021-11-22)
   Website: http://www.gingerblue.org/

 */

#include <glib/gstdio.h>
#include <glib/gi18n.h>
#include <gst/gst.h>
#include <gtk/gtk.h>
#include "gingerblue.h"

GtkWidget *knob (GingerblueData *data, GtkWidget *line, gint
    jack, gchar *label, gboolean lpf, gboolean hpf) {
    GtkWidget *knob;
    knob = gtk_volume_button_new ();
    return (knob);
}
```

8.1.18 gingerblue/src/gingerblue-line.c

```
/* $Id$

   Copyright (C) 2018-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 3.0.1 (2021-11-22)
   Website: http://www.gingerblue.org/

 */

 #include <gst/gst.h>
 #include <gtk/gtk.h>
 #include <glib/gstdio.h>
 #include <glib/gi18n.h>

GtkWidget *line (gint jack, gchar *label) {
    GtkWidget *window;
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), label);
```

```
20      return (window);
21  }
```

8.1.19 gingerblue/src/gingerblue-main.c

```
1   /* $Id$
2
3      Copyright (C) 2018-2021 Aamot Software
4      Author(s): Ole Aamot <ole@gnome.org>
5      License: GNU GPL version 3
6      Version: 3.0.1 (2021-11-22)
7      Website: http://www.gingerblue.org/
8
9    */
10
11  #include <config.h>
12  #include <stdlib.h>
13  #include <glib/gi18n.h>
14  #include <gst/gst.h>
15  #include <gst/player/player.h>
16  #include <gst/tag/tag.h>
17  #include <gtk/gtk.h>
18  #include <glib/gstdio.h>
19  #include <glib/gi18n.h>
20  #include <champlain/champlain.h>
21  #include <champlain-gtk/champlain-gtk.h>
22  #include <string.h>
23  #include "gingerblue.h"
24  #include "gingerblue-chord.h"
25  #include "gingerblue-config.h"
26  #include "gingerblue-main.h"
27  #include "gingerblue-main-loop.h"
28  #include "gingerblue-record.h"
29  #include "gingerblue-studio-config.h"
30  #include "gingerblue-studio-location.h"
31  #include "gingerblue-studio-stream.h"
32
33  GingerblueData *Gingerblue;
34
35  static void gb_assistant_entry_changed(GtkEditable *,
        GtkAssistant *,
36                                         GstElement *);
37  static void gb_assistant_button_toggled(GtkCheckButton *,
        GtkAssistant *);
38  static void gb_assistant_button_clicked(GtkButton *,
        GtkAssistant *);
39  static void gb_assistant_cancel(GtkAssistant *, gpointer);
40  static void gb_assistant_close(GtkAssistant *, gpointer);
```

```
41 static void gb_assistant_apply(GtkAssistant *, gpointer);
42
43 typedef struct {
44     GtkWidget *widget;
45     gint index;
46     const gchar *title;
47     GtkAssistantPageType type;
48     gboolean complete;
49 } PageInfo;
50
51 GtkWidget *musician_entry, *musician_label;
52 GtkWidget *song_entry, *song_label;
53 GtkWidget *instrument_entry, *instrument_label;
54 GtkWidget *label_entry, *label_label;
55 GtkWidget *line_entry, *line_label;
56 GtkWidget *computer_entry, *computer_label;
57 GtkWidget *recording_entry, *recording_label;
58 GtkWidget *studio_entry, *studio_label;
59 GtkWidget *stream_entry, *stream_label;
60 GtkWidget *album_entry, *album_label;
61 GtkWidget *summary_entry, *summary_label;
62
63 GMainLoop *main_loops;
64
65 GstPlayer *player;
66
67 GstTagList *tag_list;
68
69 GError *error = NULL;
70
71 gchar *htmlfile = NULL;
72 gchar *htmlfp = NULL;
73
74 static void gb_assistant_entry_changed(GtkEditable * editable,
75                       GtkAssistant * assistant,
76                       GstElement * pipeline)
77 {
78     return;
79 }
80
81 static void gb_assistant_button_toggled(GtkCheckButton *
       checkbutton,
82                     GtkAssistant * assistant)
83 {
84     return;
85 }
86
87 static void gb_assistant_button_clicked(GtkButton * button,
88                     GtkAssistant * assistant)
89 {
90     GstElement *src, *conv, *enc, *muxer, *sink, *recorder;
91     gchar *filename = NULL;
92     GDateTime *datestamp = g_date_time_new_now_utc ();
93     GstElementFactory *factory;
```

```
94      gst_element_send_event(recorder, gst_event_new_eos());
95      recorder = gst_pipeline_new("record_pipe");
96      /*
97        FIXME: Line #59 from https://github.com/GStreamer/gst-
             plugins-base/blob/master/tools/gst-device-monitor.c
98        element = gst_device_create_element (device, NULL);
99        if (!element)
100       return NULL;
101       factory = gst_element_get_factory (element);
102       if (!factory) {
103       gst_object_unref (element);
104       return NULL;
105       }
106       src = gst_element_factory_create(factory, NULL);
107     */
108     src = gst_element_factory_make("autoaudiosrc", "auto_source"
          );
109     conv = gst_element_factory_make("audioconvert", "convert");
110     enc = gst_element_factory_make("vorbisenc", "vorbis_enc");
111     muxer = gst_element_factory_make("oggmux", "oggmux");
112     sink = gst_element_factory_make("filesink", "sink");
113     filename = g_strconcat(g_get_user_special_dir(
          G_USER_DIRECTORY_MUSIC), "/",
114                 gtk_entry_get_text(GTK_ENTRY(musician_entry))
                      , "_-_",
115                 gtk_entry_get_text(GTK_ENTRY(song_entry)), "_
                      [",
116                 g_date_time_format (datestamp, "%Y-%m-%dT%H:%
                      M:%S"),
117                 "]",
118                 ".ogg", NULL);
119     htmlfile = g_strconcat(g_get_user_special_dir(
          G_USER_DIRECTORY_MUSIC), "/",
120                 gtk_entry_get_text(GTK_ENTRY(musician_entry))
                      , "_-_",
121                 gtk_entry_get_text(GTK_ENTRY(song_entry)), "_
                      [",
122                 g_date_time_format (datestamp, "%Y-%m-%dT%H:%
                      M:%S"),
123                 "]",
124                 ".html", NULL);
125     htmlfp = fopen(htmlfile, "w");
126     fprintf(htmlfp, "<html>\n␣␣<head>\n␣␣␣␣<title>%s␣-␣%s</title
          >\n␣␣</head>\n␣␣<body>\n␣␣␣␣<h1>%s</h1>\n",
          gtk_entry_get_text(GTK_ENTRY(musician_entry)),
          gtk_entry_get_text(GTK_ENTRY(song_entry)),
          gtk_entry_get_text(GTK_ENTRY(musician_entry)),
          gtk_entry_get_text(GTK_ENTRY(song_entry)));
127     fprintf(htmlfp, "<h2><a␣href='http://wiki.gnome.org/Apps/
          Gingerblue'>Gingerblue␣3.0.1</a></h2>\n");
128     fprintf(htmlfp, "<p><a␣href='%s'>%s</a>␣(Ogg␣Vorbis,␣44.1kHz
          ,␣Mono)",
129          gtk_entry_get_text(GTK_ENTRY(recording_entry)),
```

```
130         g_strconcat(g_get_user_special_dir(
                G_USER_DIRECTORY_MUSIC),
131                 "/", gtk_entry_get_text(GTK_ENTRY(musician_entry
                        )), "_-_",
132                 gtk_entry_get_text(GTK_ENTRY(song_entry)),
133                 ".ogg", NULL));
134     g_object_set(G_OBJECT(sink), "location",
135             g_strconcat(g_get_user_special_dir(
                    G_USER_DIRECTORY_MUSIC),
136                 "/", gtk_entry_get_text(GTK_ENTRY(
                        musician_entry)), "_-_",
137                 gtk_entry_get_text(GTK_ENTRY(song_entry)),
138                 ".ogg", NULL), NULL);
139     gst_bin_add_many(GST_BIN(recorder), src, conv, enc, muxer,
            sink, NULL);
140     gst_element_link_many(src, conv, enc, muxer, sink, NULL);
141     gst_element_set_state(recorder, GST_STATE_PLAYING);
142     tag_list = gst_tag_list_new (GST_TAG_ARTIST,
            gtk_entry_get_text(GTK_ENTRY(musician_entry)), NULL);
143     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
144     tag_list = gst_tag_list_new (GST_TAG_ALBUM,
            gtk_entry_get_text(GTK_ENTRY(album_entry)), NULL);
145     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
146     tag_list = gst_tag_list_new (GST_TAG_TITLE,
            gtk_entry_get_text(GTK_ENTRY(song_entry)), NULL);
147     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
148     tag_list = gst_tag_list_new (GST_TAG_COPYRIGHT,
            gtk_entry_get_text(GTK_ENTRY(label_entry)), NULL);
149     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
150     tag_list = gst_tag_list_new (GST_TAG_PUBLISHER,
            gtk_entry_get_text(GTK_ENTRY(label_entry)), NULL);
151     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
152     tag_list = gst_tag_list_new (GST_TAG_DATE_TIME, datestamp,
            NULL);
153     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
154     gst_vorbis_tag_add (tag_list, GST_TAG_ARTIST,
            gtk_entry_get_text(GTK_ENTRY(musician_entry)));
155     gst_vorbis_tag_add (tag_list, GST_TAG_ALBUM,
            gtk_entry_get_text(GTK_ENTRY(song_entry)));
156     gst_vorbis_tag_add (tag_list, GST_TAG_TITLE,
            gtk_entry_get_text(GTK_ENTRY(song_entry)));
157     gst_vorbis_tag_add (tag_list, GST_TAG_COPYRIGHT,
            gtk_entry_get_text(GTK_ENTRY(label_entry)));
158     gst_vorbis_tag_add (tag_list, GST_TAG_PUBLISHER,
            gtk_entry_get_text(GTK_ENTRY(label_entry)));
159     gst_vorbis_tag_add (tag_list, GST_TAG_DATE_TIME, datestamp);
160     gst_vorbis_tag_add (tag_list, GST_TAG_DATE_TIME, datestamp);
161     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
162     main_loops = g_main_loop_new(NULL, TRUE);
163     g_main_loop_run(main_loops);
164     gst_element_set_state(recorder, GST_STATE_NULL);
165     g_main_loop_unref(main_loops);
166     gst_object_unref(GST_OBJECT(recorder));
167     g_date_time_unref (datestamp);
```

```
168 }
169
170 static void gb_assistant_cancel(GtkAssistant * assistant,
        gpointer data)
171 {
172     if (!main_loops) {
173         g_error("Quit␣more␣loops␣than␣there␣are.");
174     } else {
175         GMainLoop *loop = main_loops;
176         g_main_loop_quit(loop);
177         gtk_main_quit();
178     }
179 }
180
181 static void gb_assistant_close(GtkAssistant * assistant,
        gpointer data)
182 {
183     FILE *fp = NULL;
184     GDateTime *datestamp = g_date_time_new_now_utc ();
185     gchar *filename =
186         g_strconcat(g_get_user_special_dir(
                G_USER_DIRECTORY_MUSIC), "/",
187             gtk_entry_get_text(GTK_ENTRY(musician_entry)), "_-_"
                ,
188             gtk_entry_get_text(GTK_ENTRY(song_entry)),  "_[",
189             g_date_time_format (datestamp, "%Y-%m-%dT%H:%M:%S"),
                 "]",
190             ".gingerblue", NULL);
191     gchar *htmlfile = g_strconcat(g_get_user_special_dir(
            G_USER_DIRECTORY_MUSIC), "/",
192             gtk_entry_get_text(GTK_ENTRY(musician_entry)), "_-_"
                ,
193             gtk_entry_get_text(GTK_ENTRY(song_entry)),  "_[",
194             g_date_time_format (datestamp, "%Y-%m-%dT%H:%M:%S"),
                 "]",
195             ".html", NULL);
196
197     htmlfp = fopen(htmlfile, "a+");
198     fprintf(fp, "<p><a␣href='%s'>%s␣(%s)</a></p>", filename,
            gtk_entry_get_text(GTK_ENTRY(musician_entry)),
            gtk_entry_get_text(GTK_ENTRY(instrument_entry)));
199     fprintf(fp, "<?xml␣version='1.0'␣encoding='UTF-8'?>\n");
200     fprintf(fp, "<gingerblue␣version='%s'>\n", VERSION);
201     fprintf(fp, "␣␣<musician>%s</musician>\n",
202         gtk_entry_get_text(GTK_ENTRY(musician_entry)));
203     fprintf(fp, "␣␣<song>%s</song>\n",
204         gtk_entry_get_text(GTK_ENTRY(song_entry)));
205     fprintf(fp, "␣␣<instrument>%s</instrument>\n",
206         gtk_entry_get_text(GTK_ENTRY(instrument_entry)));
207     fprintf(fp, "␣␣<line>%s</line>\n",
208         gtk_entry_get_text(GTK_ENTRY(line_entry)));
209     fprintf(fp, "␣␣<label>%s</label>\n",
210         gtk_entry_get_text(GTK_ENTRY(label_entry)));
211     fprintf(fp, "␣␣<station>%s</station>\n",
```

```
212         gtk_entry_get_text(GTK_ENTRY(computer_entry)));
213     fprintf(fp, "␣␣<filename>%s</filename>\n",
214         gtk_entry_get_text(GTK_ENTRY(recording_entry)));
215     fprintf(fp, "␣␣<album>%s</album>\n",
216         gtk_entry_get_text(GTK_ENTRY(album_entry)));
217     fprintf(fp, "␣␣<studio>%s</studio>\n",
218         gtk_entry_get_text(GTK_ENTRY(studio_entry)));
219     fprintf(fp, "</gingerblue>\n");
220     fclose(fp);
221     g_date_time_unref (datestamp);
222     gst_element_send_event(data, gst_event_new_eos());
223 }
224
225 static void gb_assistant_apply(GtkAssistant * assistant,
        gpointer data)
226 {
227         GingerblueData *gingerblue_config;
228         GtkWindow *gingerblue_window;
229         /* gtk_init (&argc, &argv); */
230         gingerblue_config = main_config (GTK_WIDGET(
                gingerblue_window), gtk_entry_get_text(GTK_ENTRY(
                studio_entry)));
231         gingerblue_window = gingerblue_main_loop (
                gingerblue_config);
232         gtk_widget_show_all (gingerblue_window);
233         /* gst_init(&argc, &argc); */
234         /* gtk_main(); */
235     gst_element_send_event(data, gst_event_new_eos());
236 }
237
238 static void gb_record_cb (char *path, gpointer data)
239 {
240         return;
241 }
242
243 GtkAssistantPageFunc gb_assistant_cb(GtkAssistant * assistant,
244                     GDateTime * datestamp)
245 {
246     /* gtk_assistant_next_page(assistant); */
247 }
248
249 int main(int argc, char **argv)
250 {
251     GDateTime *datestamp;
252     GingerblueData *data;
253     GingerblueChord *gingerblue_chord;
254     GstElement *src, *conv, *enc, *muxer, *sink, *pipeline;
255     GtkWidget *introduction;
256     GtkEntryBuffer *default_recording_title;
257     GtkWidget *entry, *label, *button, *progress, *hbox;
258     GtkWidget *summary_label, *summary_entry;
259     GtkWidget *gingerblue_main;
260     guint i;
261     GtkWidget *musicianpage;
```

```
262     GtkWidget *songpage;
263     GtkWidget *instrumentpage;
264     GtkWidget *recordpage;
265     GtkWidget *window;
266     GtkWidget *frame;
267     GtkWidget *input;
268     GtkWidget *main_window;
269     GtkWidget *mixer;
270     GtkWidget *control;
271     GtkWidget *soundboard;
272     GtkWidget *toolbar;
273     GtkWidget *input_record;
274     GtkWidget *input_pause;
275     GtkWidget *input_break;
276     GtkWidget *input_stop;
277     GtkWidget *input_volume;
278     gdouble input_volume_value;
279     gint64 real_time;
280     gchar *album;
281     PageInfo page[11] = {
282         {NULL, -1, "Gingerblue Setup", GTK_ASSISTANT_PAGE_INTRO,
                TRUE},
283         {NULL, -1, "Musician", GTK_ASSISTANT_PAGE_CONTENT, TRUE
               },
284         {NULL, -1, "Song", GTK_ASSISTANT_PAGE_CONTENT, TRUE},
285         {NULL, -1, "Instrument", GTK_ASSISTANT_PAGE_CONTENT,
               TRUE},
286         {NULL, -1, "Input Line", GTK_ASSISTANT_PAGE_CONTENT,
               TRUE},
287         {NULL, -1, "Label", GTK_ASSISTANT_PAGE_CONTENT, TRUE},
288         {NULL, -1, "Computer", GTK_ASSISTANT_PAGE_CONTENT, TRUE
               },
289         {NULL, -1, "Recording", GTK_ASSISTANT_PAGE_CONTENT, TRUE
               },
290         {NULL, -1, "Studio", GTK_ASSISTANT_PAGE_CONTENT, TRUE},
291         {NULL, -1, "Album", GTK_ASSISTANT_PAGE_CONTENT, TRUE},
292         {NULL, -1, "Broadcasting", GTK_ASSISTANT_PAGE_CONFIRM,
               TRUE},
293     };
294     FILE *xspf = NULL;
295     datestamp = g_date_time_new_now_utc ();
296     gchar *filename = g_strconcat(g_get_user_special_dir(
           G_USER_DIRECTORY_MUSIC), "/",
297                     gtk_entry_get_text(GTK_ENTRY(
                           musician_entry)), "_-_",
298                     gtk_entry_get_text(GTK_ENTRY(song_entry)),
                           "_[",
299                     g_date_time_format (datestamp, "%Y-%m-%dT%
                           H:%M:%S"), "]",
300                     ".ogg", NULL);
301     gtk_init(&argc, &argv);
302     window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
303     introduction = gtk_assistant_new();
```

```
304     gtk_widget_set_size_request(GTK_WIDGET(introduction), 640,
            480);
305     gtk_window_set_title(GTK_WINDOW(introduction), "GNOME␣
            Gingerblue");
306     g_signal_connect(G_OBJECT(introduction), "destroy",
307             G_CALLBACK(gtk_main_quit), NULL);
308     page[0].widget = gtk_label_new(_("Welcome␣to␣GNOME␣
            Gingerblue!\n\nRecord␣respectfully␣around␣others.\n\
            nClick␣Next␣to␣setup␣a␣music␣recording␣session!\n\nClick␣
            Cancel␣to␣stop␣the␣music␣recording␣session.\n\nClick␣
            Cancel␣twice␣to␣exit␣GNOME␣Gingerblue."));
309     page[1].widget = gtk_box_new(FALSE, 5);
310     musician_label = gtk_label_new(_("Musician:"));
311     musician_entry = gtk_entry_new();
312     if (g_strcmp0(musician_entry, NULL)!=0) gtk_entry_set_text(
            GTK_ENTRY(musician_entry), g_get_real_name()); else
            gtk_entry_set_text(GTK_ENTRY(musician_entry),
            gtk_entry_get_text(GTK_ENTRY(musician_entry)));
313     gtk_box_pack_start(GTK_BOX(page[1].widget), GTK_WIDGET(
            musician_label),
314             FALSE, FALSE, 5);
315     gtk_box_pack_start(GTK_BOX(page[1].widget), GTK_WIDGET(
            musician_entry),
316             FALSE, FALSE, 5);
317     page[2].widget = gtk_box_new(FALSE, 5);
318     song_label = gtk_label_new(_("Song:"));
319     song_entry = gtk_entry_new();
320     if (g_strcmp0(song_entry, NULL)!=0) gtk_entry_set_text(
            GTK_ENTRY(song_entry), g_strconcat (_("Song_-_"),
            g_date_time_format (datestamp, "%Y-%m-%dT%H:%M:%S"), NULL
            )); else gtk_entry_set_text(GTK_ENTRY(song_entry),
            gtk_entry_get_text(GTK_ENTRY(song_entry)));
321     gtk_box_pack_start(GTK_BOX(page[2].widget), GTK_WIDGET(
            song_label),
322             FALSE, FALSE, 5);
323     gtk_box_pack_start(GTK_BOX(page[2].widget), GTK_WIDGET(
            song_entry),
324             FALSE, FALSE, 5);
325     page[3].widget = gtk_box_new(FALSE, 5);
326     instrument_label = gtk_label_new(_("Instrument:"));
327     instrument_entry = gtk_entry_new();
328     gtk_entry_set_text(GTK_ENTRY(instrument_entry), _("Guitar"))
            ;
329     gtk_box_pack_start(GTK_BOX(page[3].widget),
330             GTK_WIDGET(instrument_label), FALSE, FALSE, 5);
331     gtk_box_pack_start(GTK_BOX(page[3].widget),
332             GTK_WIDGET(instrument_entry), FALSE, FALSE, 5);
333     page[4].widget = gtk_box_new(FALSE, 5);
334     line_label = gtk_label_new(_("Line␣Input:"));
335     line_entry = gtk_entry_new();
336     gtk_entry_set_text(GTK_ENTRY(line_entry), _("Mic"));
337     gtk_box_pack_start(GTK_BOX(page[4].widget), GTK_WIDGET(
            line_label),
338             FALSE, FALSE, 5);
```

```
339     gtk_box_pack_start(GTK_BOX(page[4].widget), GTK_WIDGET(
            line_entry),
340             FALSE, FALSE, 5);
341     page[5].widget = gtk_box_new(FALSE, 5);
342     label_label = gtk_label_new(_("Label:"));
343     label_entry = gtk_entry_new();
344     gtk_entry_set_text(GTK_ENTRY(label_entry), _("GNOME"));
345     gtk_box_pack_start(GTK_BOX(page[5].widget), GTK_WIDGET(
            label_label),
346             FALSE, FALSE, 5);
347     gtk_box_pack_start(GTK_BOX(page[5].widget), GTK_WIDGET(
            label_entry),
348             FALSE, FALSE, 5);
349     page[6].widget = gtk_box_new(FALSE, 5);
350     computer_label = gtk_label_new(_("Computer:"));
351     computer_entry = gtk_entry_new();
352     gtk_entry_set_text(GTK_ENTRY(computer_entry), _(
            g_get_host_name()));
353     gtk_box_pack_start(GTK_BOX(page[6].widget), GTK_WIDGET(
            computer_label),
354             FALSE, FALSE, 5);
355     gtk_box_pack_start(GTK_BOX(page[6].widget), GTK_WIDGET(
            computer_entry),
356             FALSE, FALSE, 5);
357     recording_label = gtk_button_new_with_label("Recording");
358     recording_entry = gtk_entry_new();
359     gtk_entry_set_text(GTK_ENTRY(recording_entry), g_strconcat(
            g_get_user_special_dir
360                     (G_USER_DIRECTORY_MUSIC), "/",
361                     gtk_entry_get_text(GTK_ENTRY(musician_entry
                            )), "_-_",
362                     gtk_entry_get_text(GTK_ENTRY(song_entry)),
363                                     ".ogg", NULL));
364     g_signal_connect(G_OBJECT(recording_label), "clicked",
365             G_CALLBACK(gb_record_cb),
366             g_strconcat(g_get_user_special_dir
367                     (G_USER_DIRECTORY_MUSIC), "/",
368                     gtk_entry_get_text(GTK_ENTRY(musician_entry
                            )), "_-_",
369                     gtk_entry_get_text(GTK_ENTRY(song_entry)),
370                     ".ogg", NULL));
371     page[7].widget = gtk_entry_new();
372     gtk_entry_set_text(GTK_ENTRY(page[7].widget), g_strconcat(
            g_get_user_special_dir
373                                     (G_USER_DIRECTORY_MUSIC), "/"
                                            ,
374                                    gtk_entry_get_text(GTK_ENTRY(
                                            musician_entry)), "_-_",
                                            gtk_entry_get_text(
                                            GTK_ENTRY(song_entry)), ".
                                            ogg", NULL));
375     gtk_box_pack_start(GTK_BOX(page[7].widget), GTK_WIDGET(
            recording_label),
376             FALSE, FALSE, 5);
```

```
377     gtk_box_pack_start(GTK_BOX(page[7].widget), GTK_WIDGET(
            recording_entry),
378                     FALSE, FALSE, 5);
379     studio_label = gtk_button_new_with_label("Broadcasting");
380     studio_entry = gtk_entry_new();
381     gtk_entry_set_text(GTK_ENTRY(studio_entry), g_strconcat("
            file://", gtk_entry_get_text(GTK_ENTRY(computer_entry)),
            "/", NULL));
382     g_signal_connect(G_OBJECT(studio_label), "clicked",
383             G_CALLBACK(gb_assistant_apply),
384                 gtk_entry_get_text(GTK_ENTRY(studio_entry)));
385     g_signal_connect(G_OBJECT(studio_entry), "clicked",
386             G_CALLBACK(gb_assistant_apply),
387             gtk_entry_get_text(GTK_ENTRY(studio_entry)));
388     page[8].widget = gtk_entry_new();
389     gtk_entry_set_text(GTK_ENTRY(page[8].widget),
            gtk_entry_get_text(GTK_ENTRY(studio_entry)));
390     gtk_box_pack_start(GTK_BOX(page[8].widget), GTK_WIDGET(
            studio_label),
391                     FALSE, FALSE, 5);
392     gtk_box_pack_start(GTK_BOX(page[8].widget), GTK_WIDGET(
            studio_entry),
393                     FALSE, FALSE, 5);
394     album_label = gtk_label_new("Album");
395     album_entry = gtk_entry_new();
396     g_signal_connect(G_OBJECT(album_label), "clicked",
397             G_CALLBACK(gb_assistant_apply),
398             gtk_entry_get_text(GTK_ENTRY(album_entry)));
399     album = g_strconcat(g_get_user_special_dir (
            G_USER_DIRECTORY_MUSIC), "/", gtk_entry_get_text(
            GTK_ENTRY(label_entry)), NULL);
400     gtk_entry_set_text(GTK_ENTRY(album_entry), (gchar *)album);
401     page[9].widget = gtk_entry_new();
402     gtk_entry_set_text(GTK_ENTRY(page[9].widget), album);
403     g_signal_connect (GTK_BUTTON(album_entry), "clicked",
            G_CALLBACK(gb_assistant_apply), GTK_ENTRY(album_entry));
404     g_signal_connect (GTK_BOX(page[9].widget), "clicked",
            G_CALLBACK(gb_assistant_apply), GTK_ENTRY(album_entry));
405     g_signal_connect(G_OBJECT(album_label), "clicked",
406             G_CALLBACK(gb_assistant_apply),
407             album_entry);
408     gtk_box_pack_start(GTK_BOX(page[9].widget), GTK_WIDGET(
            album_label),
409                     FALSE, FALSE, 5);
410     gtk_box_pack_start(GTK_BOX(page[9].widget), GTK_WIDGET(
            album_entry),
411                     FALSE, FALSE, 5);
412     stream_label = gtk_button_new_with_label("Protocol");
413     stream_entry = gtk_entry_new();
414     gtk_entry_set_text(GTK_ENTRY(stream_entry), "Torrent");
415     g_signal_connect(G_OBJECT(stream_entry), "clicked",
416             G_CALLBACK(gb_assistant_apply),
417                 gtk_entry_get_text(GTK_ENTRY(stream_entry)));
```

```
418     page[10].widget = gtk_label_new(_("Click␣to␣launch␣HTTP␣
            streaming␣server␣client"));
419     gtk_entry_set_text(GTK_ENTRY(page[10].widget), "Click␣Apply"
            );
420     g_signal_connect (GTK_BUTTON(stream_entry), "clicked",
            G_CALLBACK(gb_assistant_apply), gtk_entry_get_text(
            GTK_ENTRY(studio_entry)));
421     g_signal_connect(G_OBJECT(stream_label), "clicked",
422              G_CALLBACK(gb_assistant_apply),
423                  gtk_entry_get_text(GTK_ENTRY(stream_entry)));
424     gtk_box_pack_start(GTK_BOX(page[10].widget), GTK_WIDGET(
            stream_label),
425                FALSE, FALSE, 5);
426     gtk_box_pack_start(GTK_BOX(page[10].widget), GTK_WIDGET(
            stream_entry),
427                FALSE, FALSE, 5);
428     for (i = 0; i < 11; i++) {
429             page[i].index = gtk_assistant_append_page(
                    GTK_ASSISTANT(introduction),
430                                GTK_WIDGET(page[i].widget));
431         gtk_assistant_set_page_title(GTK_ASSISTANT(introduction)
                ,
432                             GTK_WIDGET(page[i].widget),
433                             page[i].title);
434         gtk_assistant_set_page_type(GTK_ASSISTANT(introduction),
435                            GTK_WIDGET(page[i].widget),
436                            page[i].type);
437         gtk_assistant_set_page_complete(GTK_ASSISTANT(
                introduction),
438                            GTK_WIDGET(page[i].widget),
439                            page[i].complete);
440     }
441     g_signal_connect(G_OBJECT(entry), "changed",
442              G_CALLBACK(gb_assistant_entry_changed), pipeline);
443     g_signal_connect(G_OBJECT(introduction), "cancel",
444              G_CALLBACK(gb_assistant_cancel), main_loops);
445     g_signal_connect(G_OBJECT(introduction), "close",
446              G_CALLBACK(gb_assistant_close), pipeline);
447     g_signal_connect(G_OBJECT(introduction), "apply",
448              G_CALLBACK(gb_assistant_close), pipeline);
449 /* musicianpage = gtk_entry_new (); */
450     /* real_time = g_get_real_time(); */
451     /* gtk_assistant_insert_page (introduction, */
452     /*                            musicianpage, */
453     /*                            0); */
454     /* gtk_assistant_set_page_title (introduction, */
455     /*                               musicianpage, */
456     /*                               "Musician Setup"); */
457     /* gtk_assistant_set_page_type (introduction, */
458     /*                              musicianpage, */
459     /*                              GTK_ASSISTANT_PAGE_INTRO); */
460     /* songpage = gtk_entry_new (); */
461     /* gtk_entry_set_text (songpage, g_strconcat(g_get_home_dir
            (), _("/Music/"), g_get_real_name(), " - Song.gingerblue
```

```
        ", NULL)); */
462     /* real_time = g_get_real_time(); */
463     /* gtk_assistant_insert_page (introduction, */
464     /*                            songpage, */
465     /*                            1); */
466     /* gtk_assistant_set_page_title (introduction, */
467     /*                               songpage, */
468     /*                               "Song Setup"); */
469     /* gtk_assistant_set_page_type (introduction, */
470     /*                              songpage, */
471     /*                              GTK_ASSISTANT_PAGE_CONTENT); */
472     /* gtk_assistant_next_page(introduction); */
473     /* instrumentpage = gtk_entry_new (); */
474     /* gtk_entry_set_text (instrumentpage, "Guitar"); */
475     /* gtk_assistant_set_page_type (introduction, */
476     /*                              instrumentpage, */
477     /*                              GTK_ASSISTANT_PAGE_CONTENT); */
478     /* gtk_assistant_insert_page (introduction, */
479     /*                            instrumentpage, */
480     /*                            2); */
481     /* gtk_assistant_set_page_title (introduction, */
482     /*                               instrumentpage, */
483     /*                               "Instrument Setup"); */
484     /* recordpage = gtk_entry_new (); */
485     /* gtk_entry_set_text (recordpage, "Microphone Line"); */
486     /* gtk_assistant_set_page_type (introduction, */
487     /*                              recordpage, */
488     /*                              GTK_ASSISTANT_PAGE_SUMMARY); */
489     /* gtk_assistant_insert_page (introduction, */
490     /*                            recordpage, */
491     /*                            3); */
492     /* gtk_assistant_set_page_title (introduction, */
493     /*                               recordpage, */
494     /*                               "Recording Setup"); */
495     /* gtk_assistant_set_page_complete (introduction, recordpage
            , 1); */
496     /* gtk_assistant_set_forward_page_func (introduction, */
497     /*                                      gb_assistant_cb, */
498     /*                                      NULL, */
499     /*                                      NULL); */
500     /* gtk_assistant_commit (introduction); */
501     gtk_widget_show_all(GTK_WIDGET(introduction));
502     /* FIXME Fix core dump
503        main_window = gingerblue_main_loop (data);
504        gtk_widget_show_all (main_window);
505      */
506     gst_init(&argc, &argv);
507     gst_init(NULL, NULL);
508
509     pipeline = gst_pipeline_new("record_pipe");
510
511     src = gst_element_factory_make("autoaudiosrc", "auto_source"
            );
512     conv = gst_element_factory_make("audioconvert", "convert");
```

```
513     enc = gst_element_factory_make("vorbisenc", "vorbis_enc");
514     muxer = gst_element_factory_make("oggmux", "oggmux");
515     sink = gst_element_factory_make("filesink", "sink");
516     filename = g_strconcat(g_get_user_special_dir(
            G_USER_DIRECTORY_MUSIC), "/",
517                    gtk_entry_get_text(GTK_ENTRY(musician_entry))
                           , "_-_",
518                    gtk_entry_get_text(GTK_ENTRY(song_entry)),"_[
                           ",
519                    g_date_time_format (datestamp, "%Y-%m-%dT%H:%
                           M:%S"), "]",
520                    ".ogg", NULL);
521     g_object_set(G_OBJECT(sink), "location",
522              g_strconcat(g_get_user_special_dir(
                    G_USER_DIRECTORY_MUSIC),
523                  "/", gtk_entry_get_text(GTK_ENTRY(
                        musician_entry)), "_-_",
524                  gtk_entry_get_text(GTK_ENTRY(song_entry)),
525                  ".ogg", NULL), NULL);
526     gst_bin_add_many(GST_BIN(pipeline), src, conv, enc, muxer,
            sink, NULL);
527     gst_element_link_many(src, conv, enc, muxer, sink, NULL);
528
529     gst_element_set_state(pipeline, GST_STATE_PLAYING);
530
531     main_loops = g_main_loop_new(NULL, TRUE);
532     g_main_loop_run(main_loops);
533
534     gst_element_set_state(pipeline, GST_STATE_NULL);
535     g_main_loop_unref(main_loops);
536     gst_object_unref(GST_OBJECT(pipeline));
537
538     /* player = play_new ("http://stream.radionorwegian.com/56.
            ogg", gingerblue_data->volume); */
539     /* input_volume_value = gb_window_set_volume(
            GTK_VOLUME_BUTTON (input_volume), 0.00);   *\/ */
540     /* g_signal_connect (GTK_BUTTON (input_record), "clicked",
            G_CALLBACK (gb_window_new_record), gingerblue_data->
            volume); */
541     /* g_signal_connect (GTK_BUTTON (input_pause), "clicked",
            G_CALLBACK (gb_window_pause_record), gingerblue_data->
            volume); */
542     /* g_signal_connect (GTK_BUTTON (input_break), "clicked",
            G_CALLBACK (gb_window_break_record), gingerblue_data->
            volume); */
543     /* g_signal_connect (GTK_VOLUME_BUTTON (input_volume), "
            value-changed", G_CALLBACK (gb_window_pause_record),
            gingerblue_data->volume); */
544     /* g_signal_connect (GTK_VOLUME_BUTTON (input_volume), "
            value-changed", G_CALLBACK (gb_window_store_volume),
            gingerblue_data->volume);   */
545     g_signal_connect(GTK_WINDOW(introduction), "destroy",
546              G_CALLBACK(gtk_main_quit), NULL);
547     g_signal_connect(GTK_WINDOW(introduction), "destroy",
```

```
548                 G_CALLBACK(gtk_main_quit), NULL);
549
550     /* g_free (gingerblue_data); */
551
552     g_date_time_unref (datestamp);
553
554     xspf = fopen(g_strconcat(g_get_user_special_dir(
            G_USER_DIRECTORY_MUSIC), "/", gtk_entry_get_text(
            GTK_ENTRY(label_entry)), ".xspf", NULL), "w+");
555     fprintf(xspf, "<?xml␣version=\"1.0\"␣encoding=\"UTF-8\"?>\n"
            );
556     fprintf(xspf, "<playlist␣version=\"1\"␣xmlns=\"http://xspf.
            org/ns/0/\">\n");
557     fprintf(xspf, "<trackList>\n");
558     fprintf(xspf, "<track>\n");
559     fprintf(xspf, "%s", g_strconcat("<title>",
            gtk_entry_get_text(GTK_ENTRY(song_entry)), "</title>\n",
            NULL));
560     fprintf(xspf, "%s", g_strconcat("<location>file://",
            gtk_entry_get_text(GTK_ENTRY(computer_entry)), "/",
            gtk_entry_get_text(GTK_ENTRY(recording_entry)), "</
            location>\n", NULL));
561     fprintf(xspf, "</track>\n");
562     fprintf(xspf, "</trackList>\n");
563     fprintf(xspf, "</playlist>\n");
564     fclose(xspf);
565
566     htmlfp = fopen(htmlfile, "a+");
567     htmlfp = fprintf(htmlfp, "<p>XSPF:␣<a␣href='%s'>%s</a></p>",
             g_strconcat(g_get_user_special_dir(
            G_USER_DIRECTORY_MUSIC), "/", gtk_entry_get_text(
            GTK_ENTRY(label_entry)), ".xspf", NULL), g_strconcat(
            g_get_user_special_dir(G_USER_DIRECTORY_MUSIC), "/",
            gtk_entry_get_text(GTK_ENTRY(label_entry)), ".xspf", NULL
            ));
568     fprintf(htmlfp, "␣␣</body>\n</html>\n");
569     fclose(htmlfp);
570
571     gtk_main();
572     return (0);
573 }
```

8.1.20 gingerblue/src/gingerblue-main-loop.c

```
1  /* $Id$
2
3     Copyright (C) 2020-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
```

```
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9   */
10
11  #include <gtk/gtk.h>
12  #include <gst/gst.h>
13  #include "gingerblue.h"
14  #include "gingerblue-studio-config.h"
15
16  extern GtkWidget *computer_entry;
17  extern GtkWidget *studio_entry;
18
19  GtkWidget *gingerblue_main_loop (GingerblueData *gingerblue) {
20      GingerblueData *Gingerblue = gingerblue;
21      Gingerblue->window = main_studio_config (gtk_entry_get_text(
            GTK_ENTRY(studio_entry)), gtk_entry_get_text(GTK_ENTRY(
            computer_entry)));
22      gtk_window_set_title (Gingerblue->window, g_strconcat(
            gtk_entry_get_text(GTK_ENTRY(computer_entry)), "␣on␣",
            gtk_entry_get_text(GTK_ENTRY(studio_entry)), NULL));
23          gtk_widget_show_all (Gingerblue->window);
24  }
```

8.1.21 gingerblue/src/gingerblue-record.c

```
1   /* $Id$
2
3      Copyright (C) 2018-2021 Aamot Software
4      Author(s): Ole Aamot <ole@gnome.org>
5      License: GNU GPL version 3
6      Version: 3.0.1 (2021-11-22)
7      Website: http://www.gingerblue.org/
8
9   */
10
11  #include <string.h>
12  #include <gst/gst.h>
13  #include <signal.h>
14  #include <unistd.h>
15  #include <stdlib.h>
16  #include <stdio.h>
17  #include <string.h>
18
19  // v4l2src ! tee name=t t. ! x264enc ! mp4mux ! filesink
       location=/home/rish/Desktop/okay.264 t. ! videoconvert !
       autovideosink
```

```
20
21 static GMainLoop *loop;
22 static GstElement *pipeline, *audio_source, *sink, *src, *tee, *
       encoder, *muxer, *filesink, *videoconvert, *videosink, *
       queue_record, *queue_display;
23 static GstBus *bus;
24 static GstPad *teepad;
25 static gboolean recording = FALSE;
26 static gint counter = 0;
27 static char *file_path;
28
29 static gboolean
30 message_cb (GstBus * bus, GstMessage * message, gpointer
       user_data)
31 {
32   switch (GST_MESSAGE_TYPE (message)) {
33     case GST_MESSAGE_ERROR:{
34       GError *err = NULL;
35       gchar *name, *debug = NULL;
36
37       name = gst_object_get_path_string (message->src);
38       gst_message_parse_error (message, &err, &debug);
39
40       g_printerr ("ERROR:␣from␣element␣%s:␣%s\n", name, err->
           message);
41       if (debug != NULL)
42         g_printerr ("Additional␣debug␣info:\n%s\n", debug);
43
44       g_error_free (err);
45       g_free (debug);
46       g_free (name);
47
48       g_main_loop_quit (loop);
49       break;
50     }
51     case GST_MESSAGE_WARNING:{
52         GError *err = NULL;
53         gchar *name, *debug = NULL;
54
55         name = gst_object_get_path_string (message->src);
56         gst_message_parse_warning (message, &err, &debug);
57
58         g_printerr ("ERROR:␣from␣element␣%s:␣%s\n", name, err->
               message);
59         if (debug != NULL)
60         g_printerr ("Additional␣debug␣info:\n%s\n", debug);
61
62         g_error_free (err);
63         g_free (debug);
64         g_free (name);
65         break;
66     }
67     case GST_MESSAGE_EOS:{
68         g_print ("Got␣EOS\n");
```

```
69          g_main_loop_quit (loop);
70          gst_element_set_state (pipeline, GST_STATE_NULL);
71          g_main_loop_unref (loop);
72          gst_object_unref (pipeline);
73          exit(0);
74          break;
75      }
76      default:
77          break;
78    }
79
80    return TRUE;
81  }
82
83  static GstPadProbeReturn unlink_cb(GstPad *pad, GstPadProbeInfo
        *info, gpointer user_data) {
84      g_print("Unlinking...");
85      GstPad *sinkpad;
86      sinkpad = gst_element_get_static_pad (queue_record, "sink");
87      gst_pad_unlink (teepad, sinkpad);
88      gst_object_unref (sinkpad);
89
90      gst_element_send_event(encoder, gst_event_new_eos());
91
92      sleep(1);
93      gst_bin_remove(GST_BIN (pipeline), queue_record);
94      gst_bin_remove(GST_BIN (pipeline), encoder);
95      gst_bin_remove(GST_BIN (pipeline), muxer);
96      gst_bin_remove(GST_BIN (pipeline), filesink);
97
98      gst_element_set_state(queue_record, GST_STATE_NULL);
99      gst_element_set_state(encoder, GST_STATE_NULL);
100     gst_element_set_state(muxer, GST_STATE_NULL);
101     gst_element_set_state(filesink, GST_STATE_NULL);
102
103     gst_object_unref(queue_record);
104     gst_object_unref(encoder);
105     gst_object_unref(muxer);
106     gst_object_unref(filesink);
107
108     gst_element_release_request_pad (tee, teepad);
109     gst_object_unref (teepad);
110
111     g_print("Unlinked\n");
112
113     return GST_PAD_PROBE_REMOVE;
114 }
115
116 void stopRecording() {
117     g_print("stopRecording\n");
118     gst_pad_add_probe(teepad, GST_PAD_PROBE_TYPE_IDLE, unlink_cb
        , NULL, (GDestroyNotify) g_free);
119     recording = FALSE;
120 }
```

```
121
122 void startRecording() {
123     g_print("startRecording\n");
124     GstPad *sinkpad;
125     GstPadTemplate *templ;
126
127     templ = gst_element_class_get_pad_template(
            GST_ELEMENT_GET_CLASS(tee), "src_%u");
128     teepad = gst_element_request_pad(tee, templ, NULL, NULL);
129     queue_record = gst_element_factory_make("queue", "
            queue_record");
130     encoder = gst_element_factory_make("x264enc", NULL);
131     muxer = gst_element_factory_make("mp4mux", NULL);
132     filesink = gst_element_factory_make("filesink", NULL);
133     char *file_name = (char*) malloc(255 * sizeof(char));
134     sprintf(file_name, "%s%d.mp4", file_path, counter++);
135     g_print("Recording␣to␣file␣%s", file_name);
136     g_object_set(filesink, "location", file_name, NULL);
137     g_object_set(encoder, "tune", 4, NULL);
138     free(file_name);
139
140     gst_bin_add_many(GST_BIN(pipeline), gst_object_ref(
            queue_record), gst_object_ref(encoder), gst_object_ref(
            muxer), gst_object_ref(filesink), NULL);
141     gst_element_link_many(queue_record, encoder, muxer, filesink
            , NULL);
142
143     gst_element_sync_state_with_parent(queue_record);
144     gst_element_sync_state_with_parent(encoder);
145     gst_element_sync_state_with_parent(muxer);
146     gst_element_sync_state_with_parent(filesink);
147
148     sinkpad = gst_element_get_static_pad(queue_record, "sink");
149     gst_pad_link(teepad, sinkpad);
150     gst_object_unref(sinkpad);
151
152     recording = TRUE;
153 }
154
155 int sigintHandler(int unused) {
156     g_print("You␣ctrl-c!\n");
157     if (recording)
158         stopRecording();
159     else
160         startRecording();
161     return 0;
162 }
```

8.1.22 gingerblue/src/gingerblue-song.c

```
/* $Id$

   Copyright (C) 2018-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 3.0.1 (2021-11-22)
   Website: http://www.gingerblue.org/

 */

 #include <gst/gst.h>
 #include <gtk/gtk.h>
 #include <glib/gstdio.h>
 #include <glib/gi18n.h>

GtkWidget *gb_song_new (gchar *title) {
    GtkWidget *window;
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), title);
    return (window);
}
GtkWidget *gb_song_quit (gchar *title) {
    GtkWidget *window;
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), title);
    return (window);
}
```

8.1.23 gingerblue/src/gingerblue-studio-config.c

```
/* $Id$

   Copyright (C) 2020-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 3.0.1 (2021-11-22)
   Website: http://www.gingerblue.org/

 */

#include <gtk/gtk.h>
#include <gst/gst.h>
#include "gingerblue.h"

```

```
15 GtkWidget *main_studio_config (gchar *location_data, gchar *
       studio_city) {
16         GingerblueData *Gingerblue;
17         GtkVBox *Locations;
18         GtkListBox *Location;
19         GtkContainer *Container;
20         GtkWidget *Computer;
21         GtkWidget *StudioLabel;
22         Computer = gtk_list_box_row_new();
23         StudioLabel = gtk_label_new (location_data);
24         Locations = gtk_box_new (ATK_STATE_VERTICAL, 1);
25         Location = gtk_list_box_new ();
26         gtk_container_add (GTK_CONTAINER (Computer), Locations);
27         gtk_box_pack_start (GTK_BOX (Location), StudioLabel, TRUE
               , TRUE, 0);
28         gtk_container_add (GTK_CONTAINER (Location), GTK_LIST_BOX
                (Computer));
29         gtk_container_add (GTK_CONTAINER (Container), GTK_BOX (
               Locations));
30         gtk_container_add (GTK_CONTAINER (Container),
               GTK_LIST_BOX (Location));
31         gtk_widget_show_all (GTK_WIDGET (Container));
32         return (GtkWidget *) Gingerblue;
33 }
```

8.1.24 gingerblue/src/gingerblue-studio-stream.c

```
1  /* $Id$
2
3     Copyright (C) 2020-2021 Aamot Software
4     Author(s): Ole Aamot <ole@gnome.org>
5     License: GNU GPL version 3
6     Version: 3.0.1 (2021-11-22)
7     Website: http://www.gingerblue.org/
8
9   */
10
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <string.h>
14 #include <sys/file.h>
15 #include <gtk/gtk.h>
16 #include <gst/gst.h>
17 #include <gobject/glib-types.h>
18 #include <gobject/gparam.h>
19 #include <shout/shout.h>
20 #include "gingerblue.h"
21
```

```
22 extern GtkWidget *recording_entry;
23 extern GtkWidget *studio_entry;
24 extern GtkWidget *musician_entry;
25 extern GtkWidget *song_entry;
26 extern GtkWidget *label_entry;
27
28 int main_studio_stream (gchar *location_data, gpointer *
       studio_city) {
29     shout_t *shout;
30     shout_metadata_t *pmetadata;
31     unsigned char buff[4096];
32     size_t read, total;
33     int ret;
34     shout_init();
35     if (!(shout = shout_new())) {
36         printf("Could␣not␣allocate␣shout_t\n");
37         return 1;
38     }
39     fprintf(stdout, "STUDIO:␣%s\n", gtk_entry_get_text(GTK_ENTRY
           (studio_entry)));
40     if (shout_set_host(shout, gtk_entry_get_text(GTK_ENTRY(
           studio_entry))) != SHOUTERR_SUCCESS) {
41         printf("Error␣setting␣hostname:␣%s\n", shout_get_error(
               shout));
42         return 1;
43     }
44     if (shout_set_protocol(shout, SHOUT_PROTOCOL_HTTP) !=
           SHOUTERR_SUCCESS) {
45         printf("Error␣setting␣protocol:␣%s\n", shout_get_error(
               shout));
46         return 1;
47     }
48     if (shout_set_port(shout, 8000) != SHOUTERR_SUCCESS) {
49         printf("Error␣setting␣port:␣%s\n", shout_get_error(shout
               ));
50         return 1;
51     }
52     if (shout_set_password(shout, "hackme") != SHOUTERR_SUCCESS)
            {
53         printf("Error␣setting␣password:␣%s\n", shout_get_error(
               shout));
54         return 1;
55     }
56     if (shout_set_mount(shout, "/stream") != SHOUTERR_SUCCESS) {
57         printf("Error␣setting␣mount:␣%s\n", shout_get_error(
               shout));
58         return 1;
59     }
60     if (shout_set_user(shout, "source") != SHOUTERR_SUCCESS) {
61         printf("Error␣setting␣user:␣%s\n", shout_get_error(shout
               ));
62         return 1;
63     }
```

```
64      if (shout_set_format(shout, SHOUT_FORMAT_OGG) !=
            SHOUTERR_SUCCESS) {
65          printf("Error␣setting␣user:␣%s\n", shout_get_error(shout
                ));
66          return 1;
67      }
68      if (shout_set_nonblocking(shout, 1) != SHOUTERR_SUCCESS) {
69          printf("Error␣setting␣non-blocking␣mode:␣%s\n",
                shout_get_error(shout));
70          return 1;
71      }
72      ret = shout_open(shout);
73      if (ret != SHOUTERR_SUCCESS)
74          ret = SHOUTERR_CONNECTED;
75      if (ret != SHOUTERR_BUSY)
76          printf("Connection␣pending...\n");
77      while (ret != SHOUTERR_BUSY) {
78          usleep(1000);
79          ret = shout_get_connected(shout);
80      }
81      if (ret != SHOUTERR_CONNECTED) {
82          printf("Connected␣to␣server...\n");
83          total = 0;
84          FILE *studio_stream_fp = fopen((char *)
                gtk_entry_get_text(GTK_ENTRY(recording_entry)), "r+")
                ;
85          flock(studio_stream_fp, LOCK_SH);
86          while (1) {
87              g_print(stderr, "FILENAME␣%s\n", (char *)
                    gtk_entry_get_text(GTK_ENTRY(recording_entry)));
88              total = fseek((FILE *)studio_stream_fp, 0, SEEK_CUR)
                    ;
89              read = fread(buff, 1, sizeof(buff), studio_stream_fp
                    );
90              total = total + read;
91              g_print(stderr, "%li␣of␣%li\n", read, total);
92              if (read > 0) {
93                      g_print(stderr, "%li\n", read);
94                      ret = shout_send(shout, buff, read);
95                  if (ret != SHOUTERR_SUCCESS) {
96                          printf("DEBUG:␣Send␣error:␣%s\n",
                                shout_get_error(shout));
97                      break;
98                  }
99              } else {
100                 break;
101             }
102             if (shout_queuelen(shout) > 0)
103                     printf("DEBUG:␣queue␣length:␣%d\n",
104                         (int)shout_queuelen(shout));
105             pmetadata = shout_metadata_new ();
106             shout_metadata_add (pmetadata, "Artist",
                    gtk_entry_get_text(GTK_ENTRY(musician_entry)));
```

```
107             shout_metadata_add (pmetadata, "Song",
                    gtk_entry_get_text(GTK_ENTRY(song_entry)));
108             shout_metadata_add (pmetadata, "Copyright",
                    gtk_entry_get_text(GTK_ENTRY(label_entry)));
109             shout_set_metadata (shout, pmetadata);
110             shout_sync(shout);
111             shout_metadata_free (pmetadata);
112         }
113         fclose(studio_stream_fp);
114     } else {
115         printf("Error␣connecting:␣%s\n", shout_get_error(shout))
                ;
116     }
117     shout_close(shout);
118     shout_shutdown();
119     return 0;
120 }
```

Chapter 9

Specification

GNOME Gingerblue 3.0.1 is specified with the Gingerblue XML meta data.

Chapter 10

Multiple-Location Audio Recording 1.0

10.1 Gingerblue XML Data Structure

The Gingerblue XML data structure contains a “<gingerblue>” XML root node, with “<musician>”, “<song>”, “<instrument>”, “<line>”, “<label>”, “<station>”, “<filename>”, “<album>” and “<studio>” subnodes.

10.1.1 Example

```
<?xml version='1.0' encoding='UTF-8'?>

<gingerblue version='3.0.1'>
  <musician>Root</musician>
  <song>Song</song>
  <instrument>Guitar</instrument>
  <line>Mic</line>
  <label>GNOME</label>
  <station>localhost</station>
  <filename>Song.ogg</filename>
  <album>GNOME</album>
  <studio>file://localhost/</studio>
</gingerblue>
```

10.2 Gingerblue XSPF Playlist

The Gingerblue Playlist is a subset of XSPF stored default in $HOME/Music/GNOME.xspf with a reference to the most current audio recording.

XPSF (“Spiffy”) was specified by Xiph.org and the specification is available from http://www.xpsf.org/

10.2.1 Example

```
<?xml version=''1.0'' encoding=''UTF-8''?>
<playlist version=''1'' xmlns=''http://xspf.org/ns/0/''>
  <trackList>
    <track>
      <title>Song_-_2021-10-26T04:39:15Z</title>
      <location>file://perceptron.stream//Users/wilber/Music/Wilber_-_Song_-_2021-10-26T04:39:15Z.ogg</location>
    </track>
  </trackList>
</playlist>
```

10.3 Gingerblue HTML 1.0 Document

10.3.1 Example

```
<html>
  <head>
    <title>Wilber - Song</title>
  </head>
  <body>
    <h1>Wilber</h1>
    <h2><a href='http://wiki.gnome.org/Apps/Gingerblue'>Gingerblue 3.0.1</a></h2>
    <p>
      <a href='/home/wilber/Music/Wilber_-_Song_[2021-11-22T21:33:00].ogg'>/home/wilber/Music/Wilber_-_Song_[2021-11-22T21:33:00].ogg</a>
      (Ogg Vorbis, 44.1kHz, Mono)
    </p>
    <p>XSPF: <a href='/home/wilber/Music/GNOME.xspf'>/home/wilber/Music/GNOME.xspf</a></p>
  </body>
</html>
```

Part III

Conclusion

GNOME Gingerblue 3.0.1 can be configured and compiled with the GNU C Compiler (GCC.GNU.ORG), GNU Autoconf and GNU Automake on macOS 11.6 with MacPorts 2.7.1 (MACPORTS.ORG) and is capable of recording audio from the built-in microphone on Apple MacBook Air M1 (2020) (APPLE.COM).

The recording can be achieved manually with the following statements:

- Install MacPorts 2.7.1 from https://www.macports.org/
- ```
  port install git

  git clone http://gitlab.gnome.org/ole/gingerblue.git

  cd gingerblue/

  ./configure

  make

  sudo make install

  [ENTER PASSWORD]

  /usr/local/bin/gingerblue
  ```
  ```

Chapter 11

Results

The formal proof is the audio file that was recorded running GNOME Gingerblue 3.0.1 on Apple MacBook Air M1 (2020) running macOS 11.6 with MacPorts 2.7.1 (MACPORTS.ORG) at Universitetsbiblioteket, a public library at University of Oslo and uploaded to https://www.gingerblue.org/Universitetsbiblioteket.ogg and it follows the optimal environment where this thesis and the software was written and explored.

Chapter 12

Patents Cited

- 341287 Recording and Reproducing Sounds. Sumner Tainter. May 4, 1886.
- 342214 Recording and Reproducing Speech and Other Sounds. Chichester A. Bell and Sumner Tainter. May 4, 1886.
- 661619 Method of Recording and Reproducing Sound or Signals. Valdemar Poulsen. November 13, 1900.
- 836339 Magnetizable Body for the Magnetic Record of Speech. P.O. Pedersen. November 20, 1906.
- 873078 Electromagnet For Telegraphone Purpose. Peder P. Pedersen and Valdemar Poulsen. December 10, 1907.
- 900392 Sound Recording and Reproducing Instruments. George Kirkegaard. October 6, 1908.
- 1142384 Telegraphone. George S. Tiffany. June 8, 1915.
- 1213150 Method of Producing Magnetic Sound-Records for Talking-Motion-Picture Films. Henry C. Bullis. January 23, 1917.
- 1639060 Magnetic Talking, Dictating, and Like Machine. Gustav Scheel (System-Stille GmbH). August 16, 1927.
- 1640881 High Frequency Biasing. W.L. Carlson and G.W. Carpenter. August 30, 1927.
- 1883560 Electromagnetic Sound Recording and Reproducing Machine. Harry E. Chipman. October 18, 1932.
- 1883561 Magnetic Sound Recording and Reproducing Head. Harry E. Chipman. October 18, 1932.

- 2248790 Sound Recording Device. Arnold Stapelfeldt. (C. Lorenz AG). July 8, 1941.
- 2264008 Magnetic Sound Recording Device. Arnold Stapelfeldt (C. Lorenz AG). November 25, 1941.
- 2351003 Recording and Reproduction of Vibrations. Marvin Camras and William Korzon. November 18, 1941.
- 2351007 Magnetic Recording Head. Marvin Camras. June 13, 1944. 2773120 Magnetic Recording of High Frequency Signals Earl E. Masterson (RCA). December 4, 1956.
- 2866012 Magnetic Tape Recording and Reproducing System. Charles P. Ginsburg and Shelby F. Henderson, Jr. (Ampex). December 23, 1958.
- 2900443 Magnetic Recorder and Reproducer for Video. Marvin Camras. August 18, 1959.
- 2900444 Means for Recording and Reproducing Video Signals. Marvin Camras. August 18, 1959.
- 2912517 Magnetic Tape Apparatus. Robert Fred Pfost (Ampex). November 10, 1959.
- 2912518 Magnetic Tape Apparatus. Alexander R. Maxey (Ampex). November 10, 1959.
- 2916546 Visual Image Recording and Reproducing System and Method. Charles P. Ginsburg and Ray M. Dolby (Ampex). December 8, 1959.
- 2916547 Recording and Reproducing System. Charles P. Ginsburg and Shelby F. Henderson, Jr. (Ampex). December 8, 1959.

Bibliography

[1] Kernighan, Brian W., Ritchie, Dennis M., “The C programming language”, 2006.

[2] Boney, L., Tewfik, A.H., and Hamdy, K.N., “Digital Watermarks for Audio Signals," *Proceedings of the Third IEEE International Conference on Multimedia*, pp. 473-480, June 1996.

[3] British Intelligence Objectives Subcommittee, p. 61.

[4] Chignell, Hugh, *Public Issue Radio*, Palgrave Macmillan, Great Britain, 2011.

[5] Goossens, M., Mittelbach, F., Samarin, *A LaTeX Companion*, Addison-Wesley, Reading, MA, 1994.

[6] Kopka, H., Daly P.W., *A Guide to LaTeX*, Addison-Wesley, Reading, MA, 1999.

[7] Nmungwun, A. F., *Video Recording Technology*, *Lawrence Erlbaum Associates, Publishers*, pp. 8-24, 1989.

[8] Pan, D., *A Tutorial on MPEG/Audio Compression*, *IEEE Multimedia*, Vol.2, pp.60-74, Summer 1998.

[9] Pulkki, V., Delikaris-Manias, S., Politis, A., “Parametric Time-Frequency Domain Spatial Audio”, *IEEE Press*, pp. 3-4

[10] Cox, G., “Pioneering Television News”, *John Libbey*