AAMOT OLE 19780220 UNIVERSITY OF OSLO 20220215 BSc PHYS
Public Audio Recording Software for Recording World Sounds
GNOME Gingerblue (gingerblue) version 2.0.1 (20211025)
http://www.gingerblue.org/thesis.pdf

GNOME Gingerblue 2.0.1

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 2.0.1

Ole Kristian Aamot

GNOME Gingerblue 2.0.1

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 2.0.1 (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 2.0.1 in ANSI C and available from http://WWW.GINGERBLUE.ORG/src/gingerblue-2.0.1.tar.xz with Source and Installation Packages for Fedora Core 35 (FEDORAPROJECT.ORG) and Ubuntu 21.04 (UBUNTU.COM).

The Source and Installation packages of GNOME Gingerblue 2.0.1 were tested for recording on a Hewlett Packard laptop computer and the MacPorts Installation package of 2.0.1 worked on Apple MacBook Air 2020 (M1) with macOS 11.6 Big Sur..

The Apple/HP-tested Source package of GNOME Gingerblue 2.0.1 is available from http://DOWNLOAD.GNOME.ORG/sources/gingerblue/2.0/gingerblue-2.0.1.tar.xz and a Binary package is available for MacPorts (https://www.macports.org/) on Apple macOS (https://ports.macports.org/port/gingerblue/):

```
sudo port install gingerblue
```

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, 1978) after 21 months of work that began on July 4th, 2018 as GNOME Gingerblue.

Source Code

- http://www.gingerblue.org/src/gingerblue-2.0.1.tar.xz
- https://download.gnome.org/sources/gingerblue/2.0/gingerblue-2.0.1.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

- 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 2.0.1, 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 i programmeringsspråket C (Kernighan/Ritchie, 1978).

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), gjennom ly-

dopptaksprogrammet GNOME Gingerblue (http://www.gingerblue.org/) implementert for datamaskiner i programmeringsspråket C for det grafiske skrivebordsmiljøet GNOME (http://www.gnome.org/) beskrevet i denne oppgaven (http://www.oleaamot.no/uio/bachelor/Aamot,2022.pdf) og i en Bachelor-oppgave om GNOME Voice (http://www.oleaamot.no/ntnu/bachelor/Aamot,2024.pdf) med planlagt leveranse 24. juni 2024 ved NTNU.

Takk til Professor Sverre Holm ved Fysisk institutt, Universitetet i Oslo 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 mye på veien som Bachelor-student som foreleser om svingninger og bølger i FY-ME100 våren 2002 og senere emne FYS2130 ved Fysisk Institutt ved Universitetet i Oslo, Knut Tomren og Tarald Rørvik for interesse og oppmuntring, Inger Johanne Seielstad Haugli og min tante Astrid Hanken som utvidet min interesse for lydopptak, og min utholdende og alltid oppofrende, kjærlige Mor Gunhild Humblen og stødige, vennlige Far Helge Aamot som gav meg min første kassettopptaker Julen 1985 og albumet "Face Another Day" med The Monroes på magnetbånd (Compact Cassette).

Endelig kan vi gjøre kontinuerlig, grafisk lydopptak under GStreamer til lagring på solid-state drive (SSD) på en moderne datamaskin med fri Unix som Ubuntu 21.04, Fedora 35 og macOS 11.6.

Du kan følge prosjektene på http://www.gingerblue.org/ og http://wiki.gnome.org/Apps/Gingerblue i videre utvikling.

Ole Kristian Aamot, Oslo, 15. februar 2022

Preface

In the first Multiple-Location Audio Recording Software implementation, the Free Software application GNOME Gingerblue version 2.0.1, 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 2.0.1

Ole Kristian Aamot

15 February 2022

Multiple-Location Audio Recording 1.0

Chapter 3

Historic Notes

3.1 History of Recording

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, minus one century and 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.

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.

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 2.0.1 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/).

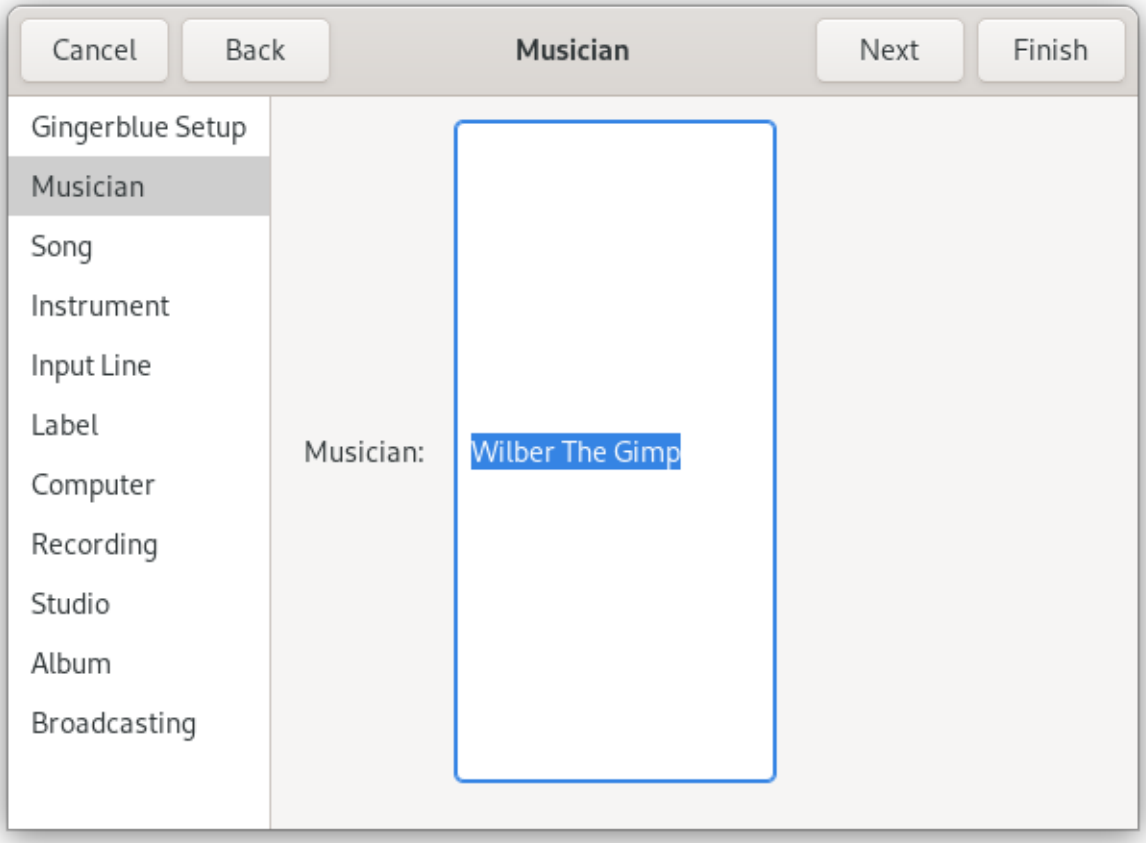
In GNOME Gingerblue version 2.0.1, the first implementation of Multiple-Location Audio Recording, as published in the thesis, audio and control rates are implemented by separate loops.

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.

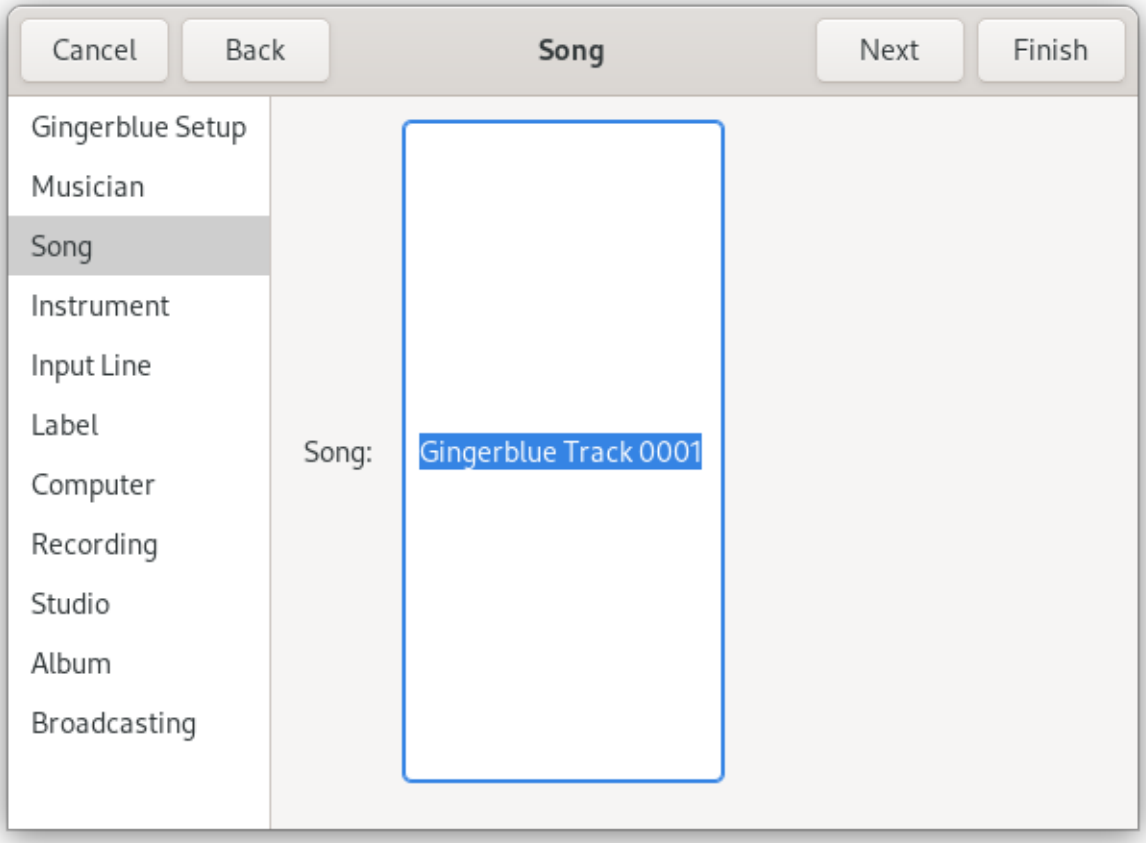
The audio signals recorded with GNOME Gingerblue version 2.0.1 have a sample rate of 44,100 Hz.

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/2.0/gingerblue-2.0.1.tar.xz) for GNOME 41 systems and https://GITLAB.GNOME.ORG/ole/gingerblue.git for the GPLv3 source code in my GNOME Git repository.

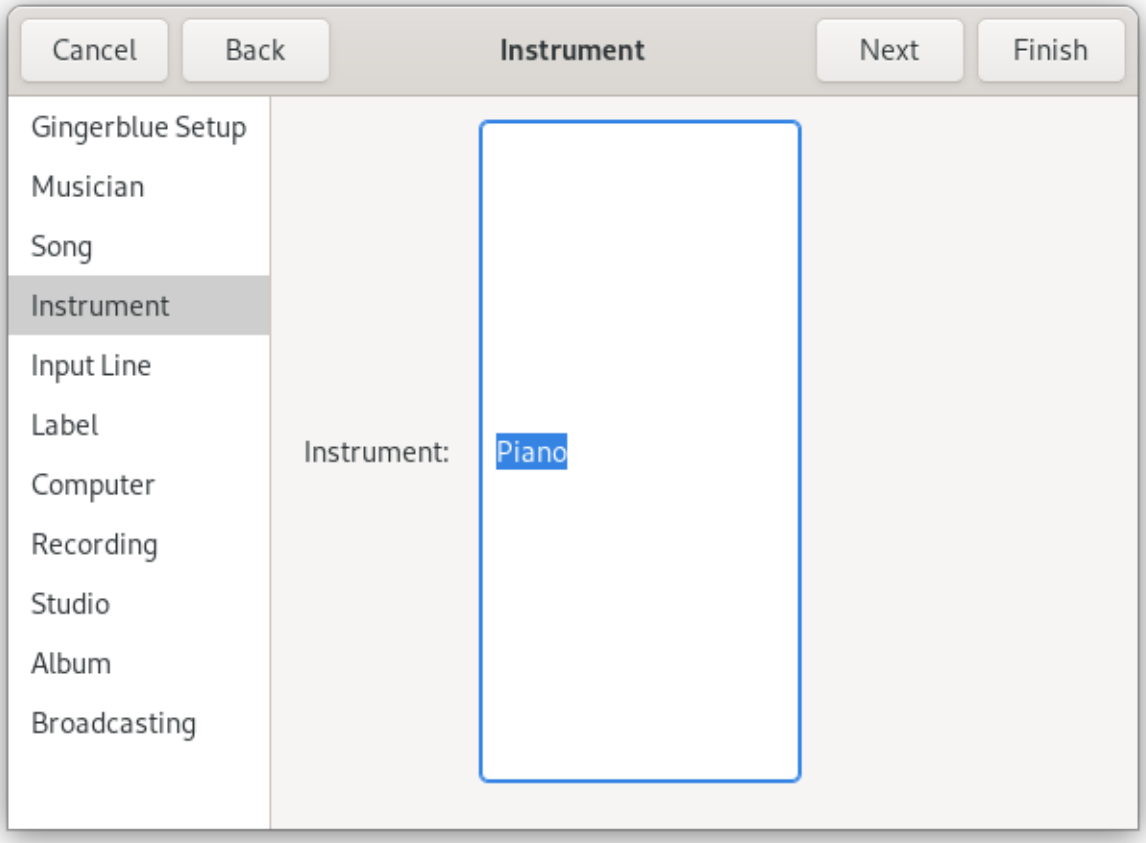
Gingerblue music recording session screen. Click “Next” to begin session.



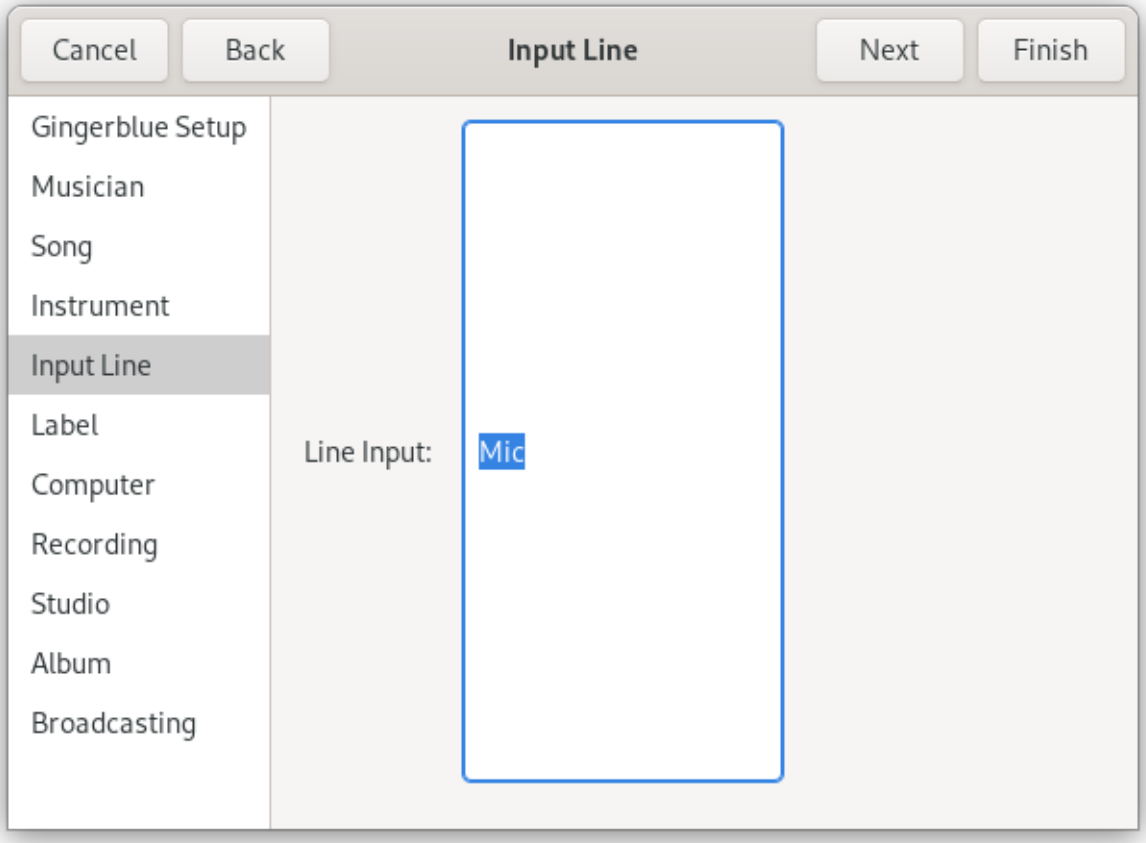
The default name of the musician is extracted from g_get_real_name(). You can edit the name of the musician and then click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip the details).



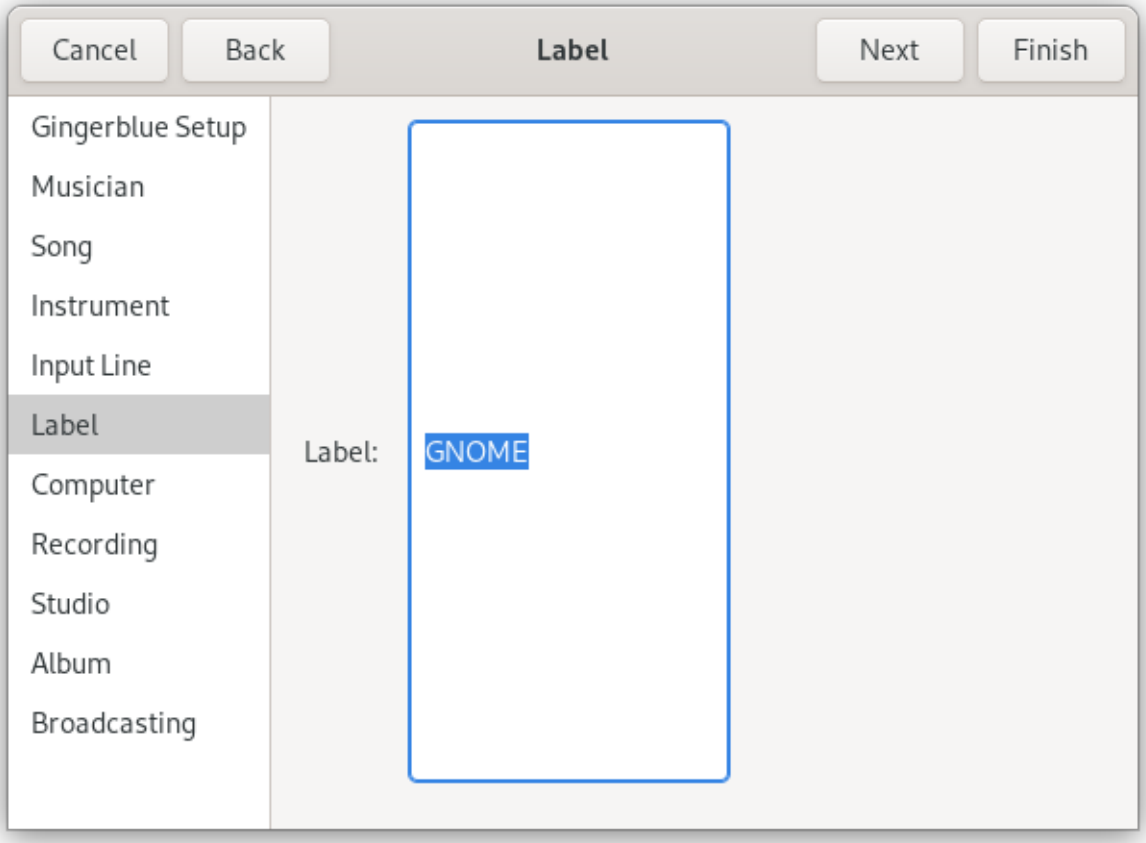
Type the name of the musical song name. Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip any of the details).



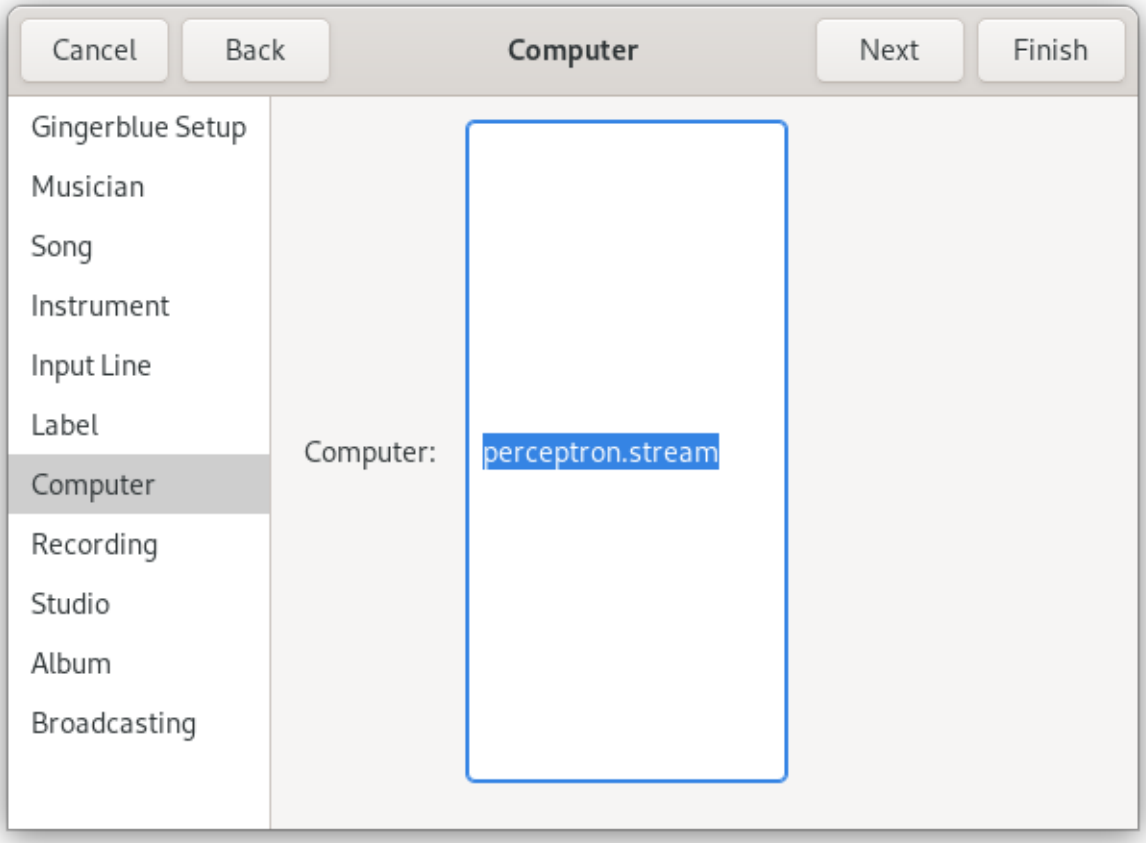
Type the name of the musical instrument. The default instrument is "Guitar". Click "Next" to continue ((or "Back" to start all over again) or "Finish" to skip any of the details).



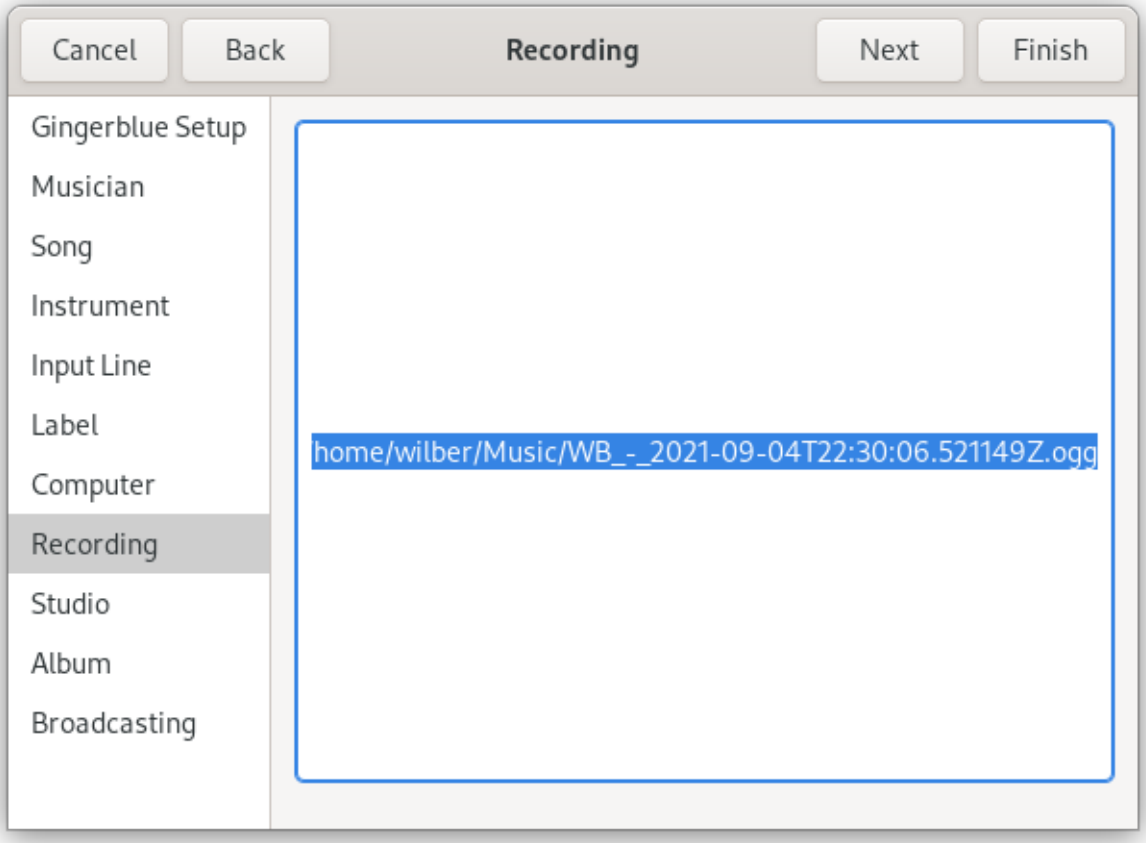
Type the name of the audio line input. The default audio line input is “Mic” (gst_pipeline_new("record_pipe") in GStreamer). Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip the details).



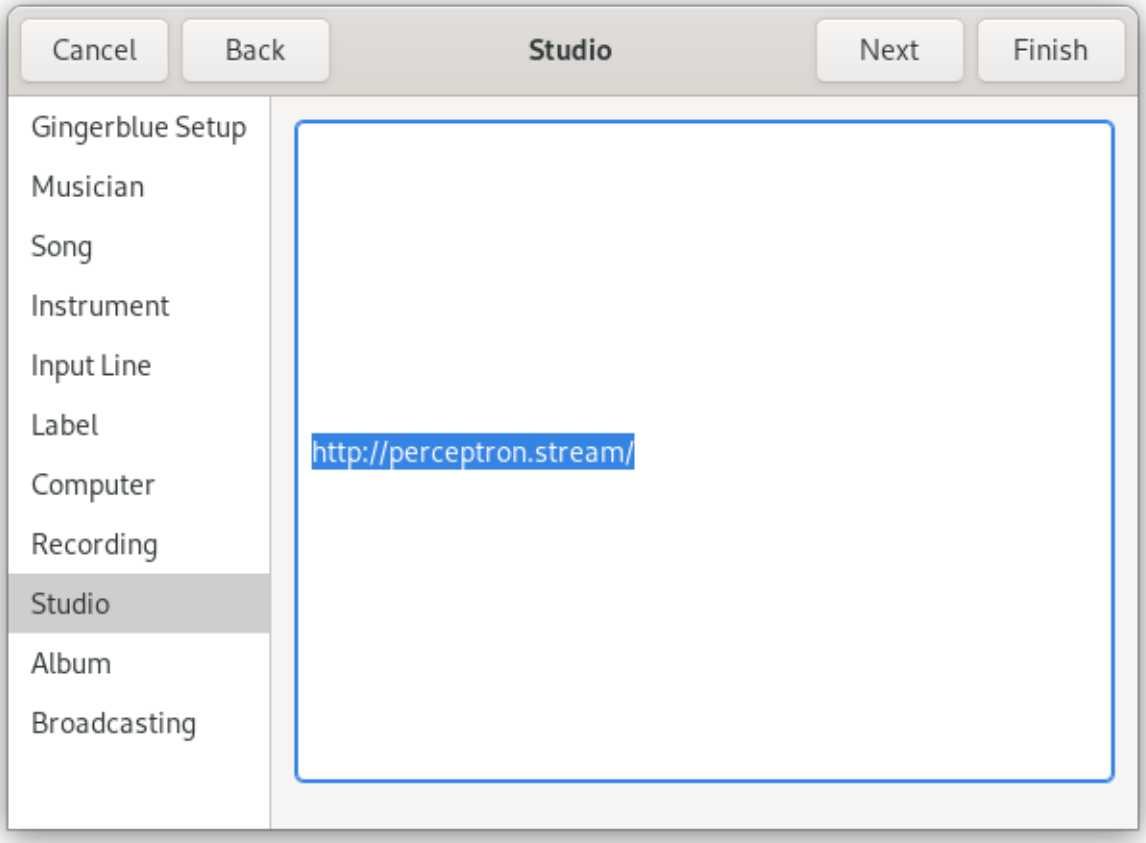
Enter the recording label. The default recording label is “GNOME” (Free label). Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip the details).



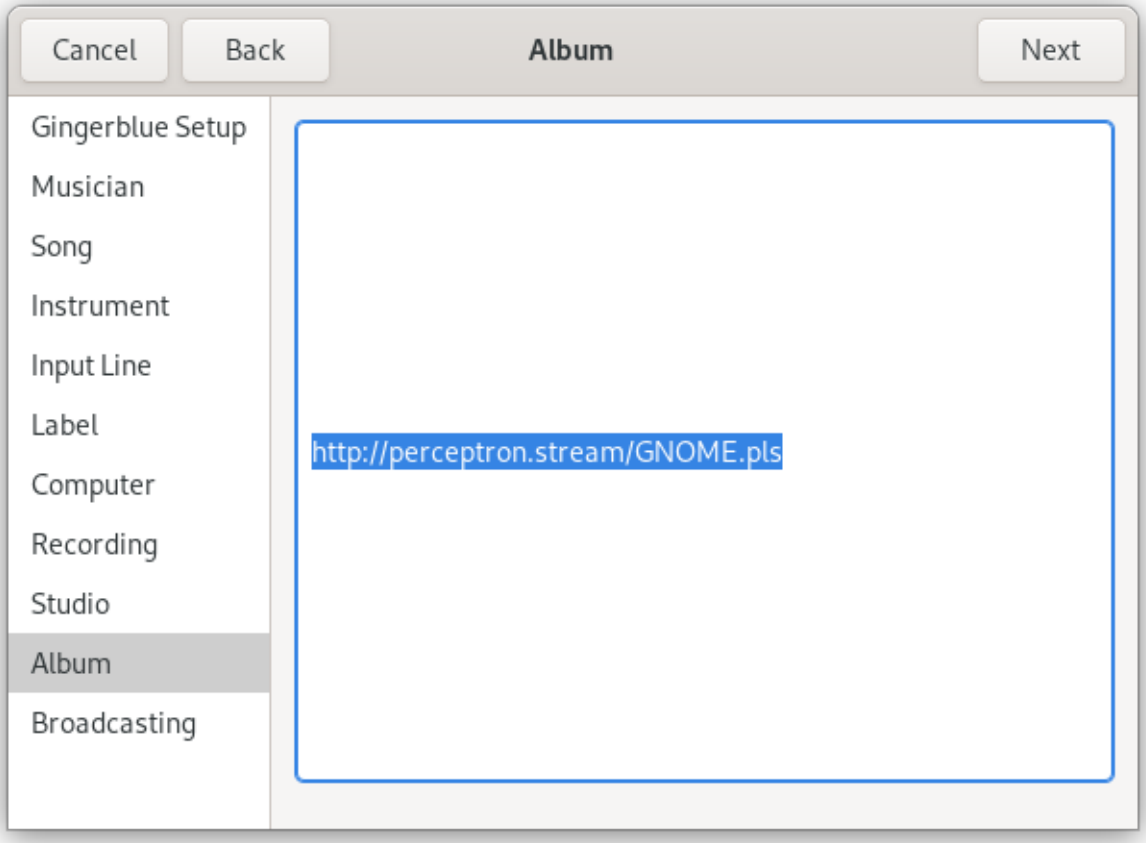
Enter the Computer. The default station label is a Fully-Qualified Domain Name (g_get_host_name()) for the local computer. Click “Next” to continue ((or “Back” to start all over again) or “Finish” to skip the details).



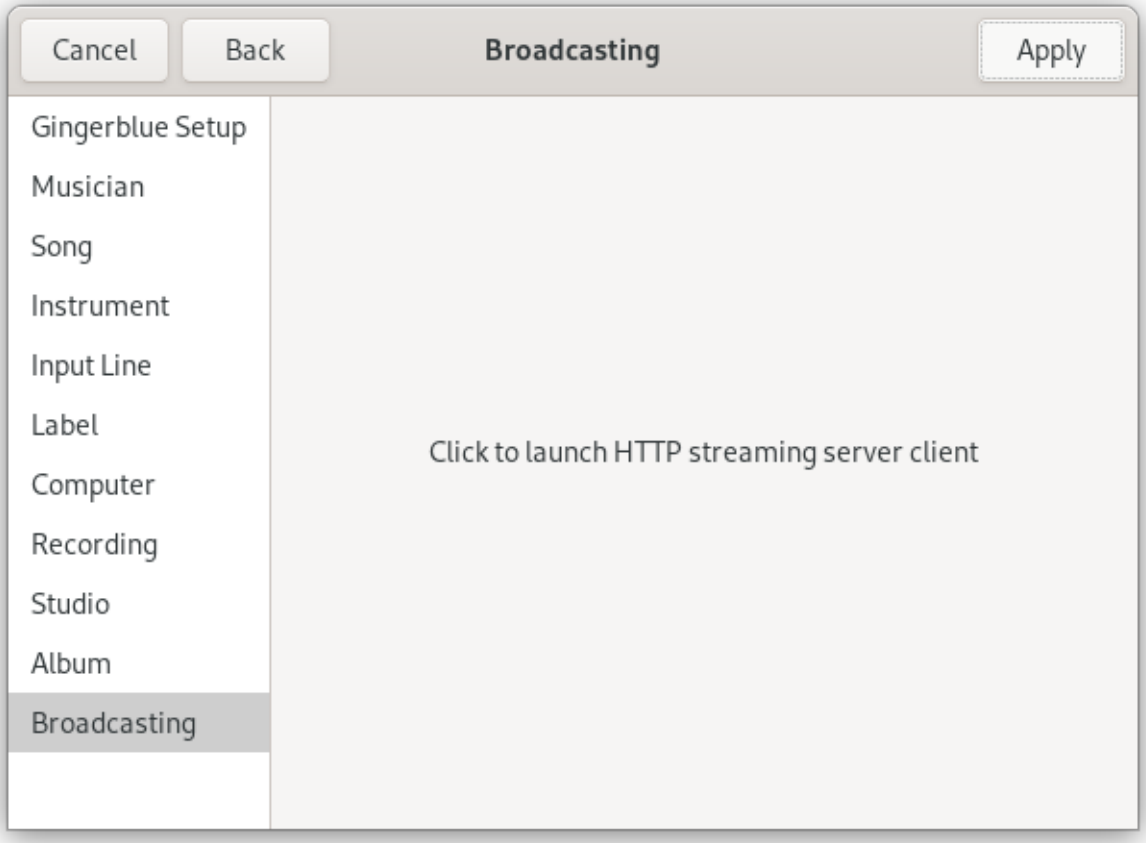
Notice the immediate, live recording file. The default immediate, live recording file name falls back to the result of g_strconcat(g_get_user_special_dir(G_USER_DIRECTORY_MUSIC), "/", gtk_entry_get_text(GTK_ENTRY(musician_entry)), "_-_", gtk_entry_get_text(GTK_ENTRY(song_entry)), "_[",g_date_time_format_iso8601 (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='2.0.1'>
  <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 2.0.1, 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 2.0.1 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

Source

7.1 gingerblue 2.0.1

7.1.1 gingerblue-2.0.1/src/gingerblue-chord.h

```
/* $Id$

   Copyright (C) 2018-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 2.0.1 (2021-10-25)
   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_ */
```

7.1.2 gingerblue-2.0.1/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 */
```

7.1.3 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 #include <libxml/xmlmemory.h>
12 #include <libxml/parser.h>
13
14 GingerblueData *gb_file_config_load (GingerblueData *head, gchar
       *filename);
15
16 static void gb_file_parse_volume (GingerblueData *data,
     xmlDocPtr doc, xmlNodePtr cur);
```

7.1.4 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
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_ */
```

7.1.5 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 GtkWidget *knob (GingerblueData *data, GtkWidget *line, gint
      jack, gchar *label, gboolean lpf, gboolean hpf);
```

7.1.6 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 GtkWidget *line (gint jack, gchar *label);
```

7.1.7 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 GtkAssistantPageFunc gb_assistant_cb (GtkAssistant *assistant,
      GDateTime *datestamp);
```

7.1.8 gingerblue-2.0.1/src/gingerblue-main-loop.h

```
#ifndef GINGERBLUE_MAIN_LOOP_H
#define GINGERBLUE_MAIN_LOOP_H 1

GtkWidget *gingerblue_main_loop (GingerblueData *gingerblue);

#endif /* GINGERBLUE_MAIN_LOOP_H */
```

7.1.9 gingerblue-2.0.1/src/gingerblue-record.h

```
/* $Id$

   Copyright (C) 2018-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 2.0.1 (2021-10-25)
   Website: http://www.gingerblue.org/

*/

#include <string.h>
#include <gst/gst.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static gboolean message_cb (GstBus * bus, GstMessage * message,
    gpointer user_data);
static GstPadProbeReturn unlink_cb(GstPad *pad, GstPadProbeInfo
    *info, gpointer user_data);
void stopRecording();
void startRecording();
int sigintHandler(int unused);
int gb_record_cb (gchar *path);

int gingerblue_record_begin();
int gingerblue_record_end();

typedef struct _GingerblueRecord {
  gboolean recording_found;
} GingerblueRecord;
```

7.1.10 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
7     Website: http://www.gingerblue.org/
8
9  */
10
11 #include <libxml/xmlmemory.h>
12 #include <libxml/parser.h>
13
14 GtkWidget *gb_song_new (gchar *title);
15 GtkWidget *gb_song_quit (gchar *title);
```

7.1.11 gingerblue-2.0.1/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 */
```

7.1.12 gingerblue-2.0.1/src/gingerblue-studio-stream.h

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

7.1.13 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
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 }
```

7.1.14 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
```

```
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 }
```

7.1.15 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
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 }
```

7.1.16 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
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 /* } */
```

7.1.17 gingerblue-2.0.1/src/gingerblue-knob.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: 2.0.1 (2021-10-25)
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
17 GtkWidget *knob (GingerblueData *data, GtkWidget *line, gint
       jack, gchar *label, gboolean lpf, gboolean hpf) {
18     GtkWidget *knob;
19     knob = gtk_volume_button_new ();
20     return (knob);
21 }
```

7.1.18 gingerblue-2.0.1/src/gingerblue-line.c

```
/* $Id$

   Copyright (C) 2018-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 2.0.1 (2021-10-25)
   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);
    return (window);
}
```

7.1.19 gingerblue-2.0.1/src/gingerblue-main.c

```
/* $Id$

   Copyright (C) 2018-2021 Aamot Software
   Author(s): Ole Aamot <ole@gnome.org>
   License: GNU GPL version 3
   Version: 2.0.1 (2021-10-25)
   Website: http://www.gingerblue.org/

 */

#include <config.h>
#include <stdlib.h>
#include <glib/gi18n.h>
#include <gst/gst.h>
#include <gst/player/player.h>
#include <gst/tag/tag.h>
#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  static void gb_assistant_entry_changed(GtkEditable * editable,
72                         GtkAssistant * assistant,
73                         GstElement * pipeline)
74  {
75      return;
76  }
77
78  static void gb_assistant_button_toggled(GtkCheckButton *
        checkbutton,
79                      GtkAssistant * assistant)
80  {
81      return;
82  }
83
84  static void gb_assistant_button_clicked(GtkButton * button,
85                      GtkAssistant * assistant)
86  {
87      GstElement *src, *conv, *enc, *muxer, *sink, *recorder;
88      gchar *filename = NULL;
89      GDateTime *datestamp = g_date_time_new_now_utc ();
90      GstElementFactory *factory;
91      gst_element_send_event(recorder, gst_event_new_eos());
92      recorder = gst_pipeline_new("record_pipe");
93      /*
94        FIXME: Line #59 from https://github.com/GStreamer/gst-
              plugins-base/blob/master/tools/gst-device-monitor.c
95        element = gst_device_create_element (device, NULL);
96        if (!element)
97        return NULL;
98        factory = gst_element_get_factory (element);
99        if (!factory) {
100       gst_object_unref (element);
101       return NULL;
102       }
103       src = gst_element_factory_create(factory, NULL);
104     */
105     src = gst_element_factory_make("autoaudiosrc", "auto_source"
          );
106     conv = gst_element_factory_make("audioconvert", "convert");
107     enc = gst_element_factory_make("vorbisenc", "vorbis_enc");
108     muxer = gst_element_factory_make("oggmux", "oggmux");
109     sink = gst_element_factory_make("filesink", "sink");
110     filename = g_strconcat(g_get_user_special_dir(
          G_USER_DIRECTORY_MUSIC), "/",
111                    gtk_entry_get_text(GTK_ENTRY(musician_entry))
                       , "_-_",
112                    gtk_entry_get_text(GTK_ENTRY(song_entry)), "_
                       [",
113                    g_date_time_format_iso8601 (datestamp),
114                    "]",
115                    ".ogg", NULL);
116     g_object_set(G_OBJECT(sink), "location",
```

```
117             g_strconcat(g_get_user_special_dir(
                    G_USER_DIRECTORY_MUSIC),
118                 "/", gtk_entry_get_text(GTK_ENTRY(
                        musician_entry)), "_-_",
119                 gtk_entry_get_text(GTK_ENTRY(song_entry)),
120                 ".ogg", NULL), NULL);
121     gst_bin_add_many(GST_BIN(recorder), src, conv, enc, muxer,
            sink, NULL);
122     gst_element_link_many(src, conv, enc, muxer, sink, NULL);
123     gst_element_set_state(recorder, GST_STATE_PLAYING);
124     tag_list = gst_tag_list_new (GST_TAG_ARTIST,
            gtk_entry_get_text(GTK_ENTRY(musician_entry)), NULL);
125     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
126     tag_list = gst_tag_list_new (GST_TAG_ALBUM,
            gtk_entry_get_text(GTK_ENTRY(album_entry)), NULL);
127     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
128     tag_list = gst_tag_list_new (GST_TAG_TITLE,
            gtk_entry_get_text(GTK_ENTRY(song_entry)), NULL);
129     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
130     tag_list = gst_tag_list_new (GST_TAG_COPYRIGHT,
            gtk_entry_get_text(GTK_ENTRY(label_entry)), NULL);
131     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
132     tag_list = gst_tag_list_new (GST_TAG_PUBLISHER,
            gtk_entry_get_text(GTK_ENTRY(label_entry)), NULL);
133     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
134     tag_list = gst_tag_list_new (GST_TAG_DATE_TIME, datestamp,
            NULL);
135     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
136     gst_vorbis_tag_add (tag_list, GST_TAG_ARTIST,
            gtk_entry_get_text(GTK_ENTRY(musician_entry)));
137     gst_vorbis_tag_add (tag_list, GST_TAG_ALBUM,
            gtk_entry_get_text(GTK_ENTRY(song_entry)));
138     gst_vorbis_tag_add (tag_list, GST_TAG_TITLE,
            gtk_entry_get_text(GTK_ENTRY(song_entry)));
139     gst_vorbis_tag_add (tag_list, GST_TAG_COPYRIGHT,
            gtk_entry_get_text(GTK_ENTRY(label_entry)));
140     gst_vorbis_tag_add (tag_list, GST_TAG_PUBLISHER,
            gtk_entry_get_text(GTK_ENTRY(label_entry)));
141     gst_vorbis_tag_add (tag_list, GST_TAG_DATE_TIME, datestamp);
142     gst_vorbis_tag_add (tag_list, GST_TAG_DATE_TIME, datestamp);
143     gst_stream_set_tags (GST_STREAM (recorder), tag_list);
144     main_loops = g_main_loop_new(NULL, TRUE);
145     g_main_loop_run(main_loops);
146     gst_element_set_state(recorder, GST_STATE_NULL);
147     g_main_loop_unref(main_loops);
148     gst_object_unref(GST_OBJECT(recorder));
149     g_date_time_unref (datestamp);
150 }
151
152 static void gb_assistant_cancel(GtkAssistant * assistant,
        gpointer data)
153 {
154     if (!main_loops) {
155         g_error("Quit␣more␣loops␣than␣there␣are.");
```

```
156     } else {
157         GMainLoop *loop = main_loops;
158         g_main_loop_quit(loop);
159         gtk_main_quit();
160     }
161 }
162
163 static void gb_assistant_close(GtkAssistant * assistant,
        gpointer data)
164 {
165     FILE *fp = NULL;
166     GDateTime *datestamp = g_date_time_new_now_utc ();
167     gchar *filename =
168         g_strconcat(g_get_user_special_dir(
              G_USER_DIRECTORY_MUSIC), "/",
169             gtk_entry_get_text(GTK_ENTRY(musician_entry)), "_-_"
                 ,
170             gtk_entry_get_text(GTK_ENTRY(song_entry)),  "_[",
171             g_date_time_format_iso8601 (datestamp), "]",
172             ".gingerblue", NULL);
173     fp = fopen(filename, "w");
174     fprintf(fp, "<?xml␣version='1.0'␣encoding='UTF-8'?>\n");
175     fprintf(fp, "<gingerblue␣version='%s'>\n", VERSION);
176     fprintf(fp, "␣␣<musician>%s</musician>\n",
177         gtk_entry_get_text(GTK_ENTRY(musician_entry)));
178     fprintf(fp, "␣␣<song>%s</song>\n",
179         gtk_entry_get_text(GTK_ENTRY(song_entry)));
180     fprintf(fp, "␣␣<instrument>%s</instrument>\n",
181         gtk_entry_get_text(GTK_ENTRY(instrument_entry)));
182     fprintf(fp, "␣␣<line>%s</line>\n",
183         gtk_entry_get_text(GTK_ENTRY(line_entry)));
184     fprintf(fp, "␣␣<label>%s</label>\n",
185         gtk_entry_get_text(GTK_ENTRY(label_entry)));
186     fprintf(fp, "␣␣<station>%s</station>\n",
187         gtk_entry_get_text(GTK_ENTRY(computer_entry)));
188     fprintf(fp, "␣␣<filename>%s</filename>\n",
189         gtk_entry_get_text(GTK_ENTRY(recording_entry)));
190     fprintf(fp, "␣␣<album>%s</album>\n",
191         gtk_entry_get_text(GTK_ENTRY(album_entry)));
192     fprintf(fp, "␣␣<studio>%s</studio>\n",
193         gtk_entry_get_text(GTK_ENTRY(studio_entry)));
194     fprintf(fp, "</gingerblue>\n");
195     fclose(fp);
196     g_date_time_unref (datestamp);
197     gst_element_send_event(data, gst_event_new_eos());
198 }
199
200 static void gb_assistant_apply(GtkAssistant * assistant,
        gpointer data)
201 {
202         GingerblueData *gingerblue_config;
203         GtkWindow *gingerblue_window;
204         /* gtk_init (&argc, &argv); */
```

```
205         gingerblue_config = main_config (GTK_WIDGET(
                gingerblue_window), gtk_entry_get_text(GTK_ENTRY(
                studio_entry)));
206         gingerblue_window = gingerblue_main_loop (
                gingerblue_config);
207         gtk_widget_show_all (gingerblue_window);
208         /* gst_init(&argc, &argc); */
209         /* gtk_main(); */
210     gst_element_send_event(data, gst_event_new_eos());
211 }
212
213 GtkAssistantPageFunc gb_assistant_cb(GtkAssistant * assistant,
214                     GDateTime * datestamp)
215 {
216     /* gtk_assistant_next_page(assistant); */
217 }
218
219 int main(int argc, char **argv)
220 {
221     GDateTime *datestamp;
222     GingerblueData *data;
223     GingerblueChord *gingerblue_chord;
224     GstElement *src, *conv, *enc, *muxer, *sink, *pipeline;
225     GtkWidget *introduction;
226     GtkEntryBuffer *default_recording_title;
227     GtkWidget *entry, *label, *button, *progress, *hbox;
228     GtkWidget *summary_label, *summary_entry;
229     GtkWidget *gingerblue_main;
230     guint i;
231     GtkWidget *musicianpage;
232     GtkWidget *songpage;
233     GtkWidget *instrumentpage;
234     GtkWidget *recordpage;
235     GtkWidget *window;
236     GtkWidget *frame;
237     GtkWidget *input;
238     GtkWidget *main_window;
239     GtkWidget *mixer;
240     GtkWidget *control;
241     GtkWidget *soundboard;
242     GtkWidget *toolbar;
243     GtkWidget *input_record;
244     GtkWidget *input_pause;
245     GtkWidget *input_break;
246     GtkWidget *input_stop;
247     GtkWidget *input_volume;
248     gdouble input_volume_value;
249     gint64 real_time;
250     gchar *album;
251     PageInfo page[11] = {
252         {NULL, -1, "Gingerblue␣Setup", GTK_ASSISTANT_PAGE_INTRO,
                TRUE},
253         {NULL, -1, "Musician", GTK_ASSISTANT_PAGE_CONTENT, TRUE
                },
```

```
254         {NULL, -1, "Song", GTK_ASSISTANT_PAGE_CONTENT, TRUE},
255         {NULL, -1, "Instrument", GTK_ASSISTANT_PAGE_CONTENT,
               TRUE},
256         {NULL, -1, "Input␣Line", GTK_ASSISTANT_PAGE_CONTENT,
               TRUE},
257         {NULL, -1, "Label", GTK_ASSISTANT_PAGE_CONTENT, TRUE},
258         {NULL, -1, "Computer", GTK_ASSISTANT_PAGE_CONTENT, TRUE
               },
259         {NULL, -1, "Recording", GTK_ASSISTANT_PAGE_CONTENT, TRUE
               },
260         {NULL, -1, "Studio", GTK_ASSISTANT_PAGE_CONTENT, TRUE},
261         {NULL, -1, "Album", GTK_ASSISTANT_PAGE_CONTENT, TRUE},
262         {NULL, -1, "Broadcasting", GTK_ASSISTANT_PAGE_CONFIRM,
               TRUE},
263     };
264     FILE *xspf = NULL;
265     datestamp = g_date_time_new_now_utc ();
266     gchar *filename = g_strconcat(g_get_user_special_dir(
           G_USER_DIRECTORY_MUSIC), "/",
267                     gtk_entry_get_text(GTK_ENTRY(
                           musician_entry)), "_-_",
268                     gtk_entry_get_text(GTK_ENTRY(song_entry)),
                           "_[",
269                     g_date_time_format_iso8601 (datestamp), "]
                           ",
270                     ".ogg", NULL);
271     gtk_init(&argc, &argv);
272     window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
273     introduction = gtk_assistant_new();
274     gtk_widget_set_size_request(GTK_WIDGET(introduction), 640,
           480);
275     gtk_window_set_title(GTK_WINDOW(introduction), "GNOME␣
           Gingerblue");
276     g_signal_connect(G_OBJECT(introduction), "destroy",
277             G_CALLBACK(gtk_main_quit), NULL);
278     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."));
279     page[1].widget = gtk_box_new(FALSE, 5);
280     musician_label = gtk_label_new(_("Musician:"));
281     musician_entry = gtk_entry_new();
282     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)));
283     gtk_box_pack_start(GTK_BOX(page[1].widget), GTK_WIDGET(
           musician_label),
284             FALSE, FALSE, 5);
285     gtk_box_pack_start(GTK_BOX(page[1].widget), GTK_WIDGET(
           musician_entry),
286             FALSE, FALSE, 5);
287     page[2].widget = gtk_box_new(FALSE, 5);
```

```
288     song_label = gtk_label_new(_("Song:"));
289     song_entry = gtk_entry_new();
290     if (g_strcmp0(song_entry, NULL)!=0) gtk_entry_set_text(
            GTK_ENTRY(song_entry), g_strconcat (_("Song_-_"),
            g_date_time_format_iso8601 (datestamp), NULL)); else
            gtk_entry_set_text(GTK_ENTRY(song_entry),
            gtk_entry_get_text(GTK_ENTRY(song_entry)));
291     gtk_box_pack_start(GTK_BOX(page[2].widget), GTK_WIDGET(
            song_label),
292             FALSE, FALSE, 5);
293     gtk_box_pack_start(GTK_BOX(page[2].widget), GTK_WIDGET(
            song_entry),
294             FALSE, FALSE, 5);
295     page[3].widget = gtk_box_new(FALSE, 5);
296     instrument_label = gtk_label_new(_("Instrument:"));
297     instrument_entry = gtk_entry_new();
298     gtk_entry_set_text(GTK_ENTRY(instrument_entry), _("Guitar"))
            ;
299     gtk_box_pack_start(GTK_BOX(page[3].widget),
300             GTK_WIDGET(instrument_label), FALSE, FALSE, 5);
301     gtk_box_pack_start(GTK_BOX(page[3].widget),
302             GTK_WIDGET(instrument_entry), FALSE, FALSE, 5);
303     page[4].widget = gtk_box_new(FALSE, 5);
304     line_label = gtk_label_new(_("Line␣Input:"));
305     line_entry = gtk_entry_new();
306     gtk_entry_set_text(GTK_ENTRY(line_entry), _("Mic"));
307     gtk_box_pack_start(GTK_BOX(page[4].widget), GTK_WIDGET(
            line_label),
308             FALSE, FALSE, 5);
309     gtk_box_pack_start(GTK_BOX(page[4].widget), GTK_WIDGET(
            line_entry),
310             FALSE, FALSE, 5);
311     page[5].widget = gtk_box_new(FALSE, 5);
312     label_label = gtk_label_new(_("Label:"));
313     label_entry = gtk_entry_new();
314     gtk_entry_set_text(GTK_ENTRY(label_entry), _("GNOME"));
315     gtk_box_pack_start(GTK_BOX(page[5].widget), GTK_WIDGET(
            label_label),
316             FALSE, FALSE, 5);
317     gtk_box_pack_start(GTK_BOX(page[5].widget), GTK_WIDGET(
            label_entry),
318             FALSE, FALSE, 5);
319     page[6].widget = gtk_box_new(FALSE, 5);
320     computer_label = gtk_label_new(_("Computer:"));
321     computer_entry = gtk_entry_new();
322     gtk_entry_set_text(GTK_ENTRY(computer_entry), _(
            g_get_host_name()));
323     gtk_box_pack_start(GTK_BOX(page[6].widget), GTK_WIDGET(
            computer_label),
324             FALSE, FALSE, 5);
325     gtk_box_pack_start(GTK_BOX(page[6].widget), GTK_WIDGET(
            computer_entry),
326             FALSE, FALSE, 5);
327     recording_label = gtk_button_new_with_label("Recording");
```

```
328     recording_entry = gtk_entry_new();
329     gtk_entry_set_text(GTK_ENTRY(recording_entry), g_strconcat(
            g_get_user_special_dir
330                     (G_USER_DIRECTORY_MUSIC), "/",
331                     gtk_entry_get_text(GTK_ENTRY(musician_entry
                            )), "_-_",
332                     gtk_entry_get_text(GTK_ENTRY(song_entry)),
333                                     ".ogg", NULL));
334     g_signal_connect(G_OBJECT(recording_label), "clicked",
335             G_CALLBACK(gb_record_cb),
336             g_strconcat(g_get_user_special_dir
337                     (G_USER_DIRECTORY_MUSIC), "/",
338                     gtk_entry_get_text(GTK_ENTRY(musician_entry
                            )), "_-_",
339                     gtk_entry_get_text(GTK_ENTRY(song_entry)),
340                     ".ogg", NULL));
341     page[7].widget = gtk_entry_new();
342     gtk_entry_set_text(GTK_ENTRY(page[7].widget), g_strconcat(
            g_get_user_special_dir
343                                     (G_USER_DIRECTORY_MUSIC), "/"
                                        ,
344                                    gtk_entry_get_text(GTK_ENTRY(
                                       musician_entry)), "_-_",
                                       gtk_entry_get_text(
                                       GTK_ENTRY(song_entry)), ".
                                       ogg", NULL));
345     gtk_box_pack_start(GTK_BOX(page[7].widget), GTK_WIDGET(
            recording_label),
346                     FALSE, FALSE, 5);
347     gtk_box_pack_start(GTK_BOX(page[7].widget), GTK_WIDGET(
            recording_entry),
348                     FALSE, FALSE, 5);
349     studio_label = gtk_button_new_with_label("Broadcasting");
350     studio_entry = gtk_entry_new();
351     gtk_entry_set_text(GTK_ENTRY(studio_entry), g_strconcat("
            file://", gtk_entry_get_text(GTK_ENTRY(computer_entry)),
            "/", NULL));
352     g_signal_connect(G_OBJECT(studio_label), "clicked",
353             G_CALLBACK(gb_assistant_apply),
354                 gtk_entry_get_text(GTK_ENTRY(studio_entry)));
355     g_signal_connect(G_OBJECT(studio_entry), "clicked",
356             G_CALLBACK(gb_assistant_apply),
357             gtk_entry_get_text(GTK_ENTRY(studio_entry)));
358     page[8].widget = gtk_entry_new();
359     gtk_entry_set_text(GTK_ENTRY(page[8].widget),
            gtk_entry_get_text(GTK_ENTRY(studio_entry)));
360     gtk_box_pack_start(GTK_BOX(page[8].widget), GTK_WIDGET(
            studio_label),
361                     FALSE, FALSE, 5);
362     gtk_box_pack_start(GTK_BOX(page[8].widget), GTK_WIDGET(
            studio_entry),
363                     FALSE, FALSE, 5);
364     album_label = gtk_label_new("Album");
365     album_entry = gtk_entry_new();
```

```
366     g_signal_connect(G_OBJECT(album_label), "clicked",
367              G_CALLBACK(gb_assistant_apply),
368              gtk_entry_get_text(GTK_ENTRY(album_entry)));
369     album = g_strconcat(g_get_user_special_dir (
            G_USER_DIRECTORY_MUSIC), "/", gtk_entry_get_text(
            GTK_ENTRY(label_entry)), NULL);
370     gtk_entry_set_text(GTK_ENTRY(album_entry), (gchar *)album);
371     page[9].widget = gtk_entry_new();
372     gtk_entry_set_text(GTK_ENTRY(page[9].widget), album);
373     g_signal_connect (GTK_BUTTON(album_entry), "clicked",
            G_CALLBACK(gb_assistant_apply), GTK_ENTRY(album_entry));
374     g_signal_connect (GTK_BOX(page[9].widget), "clicked",
            G_CALLBACK(gb_assistant_apply), GTK_ENTRY(album_entry));
375     g_signal_connect(G_OBJECT(album_label), "clicked",
376              G_CALLBACK(gb_assistant_apply),
377              album_entry);
378     gtk_box_pack_start(GTK_BOX(page[9].widget), GTK_WIDGET(
            album_label),
379                FALSE, FALSE, 5);
380     gtk_box_pack_start(GTK_BOX(page[9].widget), GTK_WIDGET(
            album_entry),
381                FALSE, FALSE, 5);
382     stream_label = gtk_button_new_with_label("Protocol");
383     stream_entry = gtk_entry_new();
384     gtk_entry_set_text(GTK_ENTRY(stream_entry), "Torrent");
385     g_signal_connect(G_OBJECT(stream_entry), "clicked",
386              G_CALLBACK(gb_assistant_apply),
387                  gtk_entry_get_text(GTK_ENTRY(stream_entry)));
388     page[10].widget = gtk_label_new(_("Click␣to␣launch␣HTTP␣
            streaming␣server␣client"));
389     gtk_entry_set_text(GTK_ENTRY(page[10].widget), "Click␣Apply"
            );
390     g_signal_connect (GTK_BUTTON(stream_entry), "clicked",
            G_CALLBACK(gb_assistant_apply), gtk_entry_get_text(
            GTK_ENTRY(studio_entry)));
391     g_signal_connect(G_OBJECT(stream_label), "clicked",
392              G_CALLBACK(gb_assistant_apply),
393                  gtk_entry_get_text(GTK_ENTRY(stream_entry)));
394     gtk_box_pack_start(GTK_BOX(page[10].widget), GTK_WIDGET(
            stream_label),
395                FALSE, FALSE, 5);
396     gtk_box_pack_start(GTK_BOX(page[10].widget), GTK_WIDGET(
            stream_entry),
397                FALSE, FALSE, 5);
398     for (i = 0; i < 11; i++) {
399             page[i].index = gtk_assistant_append_page(
                    GTK_ASSISTANT(introduction),
400                                GTK_WIDGET(page[i].widget));
401         gtk_assistant_set_page_title(GTK_ASSISTANT(introduction)
                ,
402                               GTK_WIDGET(page[i].widget),
403                               page[i].title);
404         gtk_assistant_set_page_type(GTK_ASSISTANT(introduction),
405                              GTK_WIDGET(page[i].widget),
```

```
406                         page[i].type);
407         gtk_assistant_set_page_complete(GTK_ASSISTANT(
                introduction),
408                         GTK_WIDGET(page[i].widget),
409                         page[i].complete);
410     }
411     g_signal_connect(G_OBJECT(entry), "changed",
412              G_CALLBACK(gb_assistant_entry_changed), pipeline);
413     g_signal_connect(G_OBJECT(introduction), "cancel",
414              G_CALLBACK(gb_assistant_cancel), main_loops);
415     g_signal_connect(G_OBJECT(introduction), "close",
416              G_CALLBACK(gb_assistant_close), pipeline);
417     g_signal_connect(G_OBJECT(introduction), "apply",
418              G_CALLBACK(gb_assistant_close), pipeline);
419 /* musicianpage = gtk_entry_new (); */
420     /* real_time = g_get_real_time(); */
421     /* gtk_assistant_insert_page (introduction, */
422     /*                            musicianpage, */
423     /*                            0); */
424     /* gtk_assistant_set_page_title (introduction, */
425     /*                               musicianpage, */
426     /*                               "Musician Setup"); */
427     /* gtk_assistant_set_page_type (introduction, */
428     /*                              musicianpage, */
429     /*                              GTK_ASSISTANT_PAGE_INTRO); */
430     /* songpage = gtk_entry_new (); */
431     /* gtk_entry_set_text (songpage, g_strconcat(g_get_home_dir
            (), _("/Music/"), g_get_real_name(), " - Song.gingerblue
            ", NULL)); */
432     /* real_time = g_get_real_time(); */
433     /* gtk_assistant_insert_page (introduction, */
434     /*                            songpage, */
435     /*                            1); */
436     /* gtk_assistant_set_page_title (introduction, */
437     /*                               songpage, */
438     /*                               "Song Setup"); */
439     /* gtk_assistant_set_page_type (introduction, */
440     /*                              songpage, */
441     /*                              GTK_ASSISTANT_PAGE_CONTENT); */
442     /* gtk_assistant_next_page(introduction); */
443     /* instrumentpage = gtk_entry_new (); */
444     /* gtk_entry_set_text (instrumentpage, "Guitar"); */
445     /* gtk_assistant_set_page_type (introduction, */
446     /*                              instrumentpage, */
447     /*                              GTK_ASSISTANT_PAGE_CONTENT); */
448     /* gtk_assistant_insert_page (introduction, */
449     /*                            instrumentpage, */
450     /*                            2); */
451     /* gtk_assistant_set_page_title (introduction, */
452     /*                               instrumentpage, */
453     /*                               "Instrument Setup"); */
454     /* recordpage = gtk_entry_new (); */
455     /* gtk_entry_set_text (recordpage, "Microphone Line"); */
456     /* gtk_assistant_set_page_type (introduction, */
```

```
457     /*                              recordpage, */
458     /*                              GTK_ASSISTANT_PAGE_SUMMARY); */
459     /* gtk_assistant_insert_page (introduction, */
460     /*                            recordpage, */
461     /*                            3); */
462     /* gtk_assistant_set_page_title (introduction, */
463     /*                               recordpage, */
464     /*                               "Recording Setup"); */
465     /* gtk_assistant_set_page_complete (introduction, recordpage
            , 1); */
466     /* gtk_assistant_set_forward_page_func (introduction, */
467     /*                                      gb_assistant_cb, */
468     /*                                      NULL, */
469     /*                                      NULL); */
470     /* gtk_assistant_commit (introduction); */
471     gtk_widget_show_all(GTK_WIDGET(introduction));
472     /* FIXME Fix core dump
473        main_window = gingerblue_main_loop (data);
474        gtk_widget_show_all (main_window);
475      */
476     gst_init(&argc, &argv);
477     gst_init(NULL, NULL);
478
479     pipeline = gst_pipeline_new("record_pipe");
480
481     src = gst_element_factory_make("autoaudiosrc", "auto_source"
            );
482     conv = gst_element_factory_make("audioconvert", "convert");
483     enc = gst_element_factory_make("vorbisenc", "vorbis_enc");
484     muxer = gst_element_factory_make("oggmux", "oggmux");
485     sink = gst_element_factory_make("filesink", "sink");
486     filename = g_strconcat(g_get_user_special_dir(
            G_USER_DIRECTORY_MUSIC), "/",
487                 gtk_entry_get_text(GTK_ENTRY(musician_entry))
                        , "_-_",
488                 gtk_entry_get_text(GTK_ENTRY(song_entry)),"_[
                        ",
489                 g_date_time_format_iso8601 (datestamp), "]",
490                 ".ogg", NULL);
491     g_object_set(G_OBJECT(sink), "location",
492              g_strconcat(g_get_user_special_dir(
                    G_USER_DIRECTORY_MUSIC),
493                  "/", gtk_entry_get_text(GTK_ENTRY(
                        musician_entry)), "_-_",
494                  gtk_entry_get_text(GTK_ENTRY(song_entry)),
495                  ".ogg", NULL), NULL);
496     gst_bin_add_many(GST_BIN(pipeline), src, conv, enc, muxer,
            sink, NULL);
497     gst_element_link_many(src, conv, enc, muxer, sink, NULL);
498
499     gst_element_set_state(pipeline, GST_STATE_PLAYING);
500
501     main_loops = g_main_loop_new(NULL, TRUE);
502     g_main_loop_run(main_loops);
```

```
503
504     gst_element_set_state(pipeline, GST_STATE_NULL);
505     g_main_loop_unref(main_loops);
506     gst_object_unref(GST_OBJECT(pipeline));
507
508     /* player = play_new ("http://stream.radionorwegian.com/56.
            ogg", gingerblue_data->volume); */
509     /* input_volume_value = gb_window_set_volume(
            GTK_VOLUME_BUTTON (input_volume), 0.00);   *\/ */
510     /* g_signal_connect (GTK_BUTTON (input_record), "clicked",
            G_CALLBACK (gb_window_new_record), gingerblue_data->
            volume); */
511     /* g_signal_connect (GTK_BUTTON (input_pause), "clicked",
            G_CALLBACK (gb_window_pause_record), gingerblue_data->
            volume); */
512     /* g_signal_connect (GTK_BUTTON (input_break), "clicked",
            G_CALLBACK (gb_window_break_record), gingerblue_data->
            volume); */
513     /* g_signal_connect (GTK_VOLUME_BUTTON (input_volume), "
            value-changed", G_CALLBACK (gb_window_pause_record),
            gingerblue_data->volume); */
514     /* g_signal_connect (GTK_VOLUME_BUTTON (input_volume), "
            value-changed", G_CALLBACK (gb_window_store_volume),
            gingerblue_data->volume);   */
515     g_signal_connect(GTK_WINDOW(introduction), "destroy",
516             G_CALLBACK(gtk_main_quit), NULL);
517     g_signal_connect(GTK_WINDOW(introduction), "destroy",
518             G_CALLBACK(gtk_main_quit), NULL);
519
520     /* g_free (gingerblue_data); */
521
522     g_date_time_unref (datestamp);
523
524     xspf = fopen(g_strconcat(g_get_user_special_dir(
            G_USER_DIRECTORY_MUSIC), "/", gtk_entry_get_text(
            GTK_ENTRY(label_entry)), ".xspf", NULL), "w+");
525     fprintf(xspf, "<?xml␣version=\"1.0\"␣encoding=\"UTF-8\"?>\n"
            );
526     fprintf(xspf, "<playlist␣version=\"1\"␣xmlns=\"http://xspf.
            org/ns/0/\">\n");
527     fprintf(xspf, "<trackList>\n");
528     fprintf(xspf, "<track>\n");
529     fprintf(xspf, "%s", g_strconcat("<title>",
            gtk_entry_get_text(GTK_ENTRY(song_entry)), "</title>\n",
            NULL));
530     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));
531     fprintf(xspf, "</track>\n");
532     fprintf(xspf, "</trackList>\n");
533     fprintf(xspf, "</playlist>\n");
534     fclose(xspf);
535
```

```
536     gtk_main();
537     return (0);
538 }
```

7.1.20 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
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 }
```

7.1.21 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
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 }
163
164 int gb_record_cb (char *path, gpointer data)
165 {
166     return 0;
167 }
```

7.1.22 gingerblue-2.0.1/src/gingerblue-song.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: 2.0.1 (2021-10-25)
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
16 GtkWidget *gb_song_new (gchar *title) {
17     GtkWidget *window;
18     window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
19     gtk_window_set_title (GTK_WINDOW (window), title);
20     return (window);
21 }
22 GtkWidget *gb_song_quit (gchar *title) {
23     GtkWidget *window;
24     window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
25     gtk_window_set_title (GTK_WINDOW (window), title);
26     return (window);
27 }
```

7.1.23 gingerblue-2.0.1/src/gingerblue-studio-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: 2.0.1 (2021-10-25)
7     Website: http://www.gingerblue.org/
8
9   */
10
11 #include <gtk/gtk.h>
12 #include <gst/gst.h>
13 #include "gingerblue.h"
14
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 }
```

7.1.24 gingerblue-2.0.1/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: 2.0.1 (2021-10-25)
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 8

Specification

GNOME Gingerblue 2.0.1 is specified with the Gingerblue XML meta data.

Chapter 9

Multiple-Location Audio Recording 1.0

9.1 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.

9.1.1 Example

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

<gingerblue version='2.0.1'>
  <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>
```

9.2 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/

9.2.1 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>
```

9.3 Gingerblue HTML 1.0 Document

9.3.1 Example

```
<html>
  <head>
    <title>Wilber - Song</title>
  </head>
  <body>
    <h1>Wilber</h1>
    <h2><a href='http://wiki.gnome.org/Apps/Gingerblue'>Gingerblue 2.0.1</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 2.0.1 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/
- Install binary package from macports.org

```
sudo port install gingerblue

gingerblue
```

- Install dependencies from macports.org

```
sudo port install git desktop-file-utils geoclue2 geocode-glib
sudo port install glib2 gstreamer1 libxml2 pango
sudo port install gstreamer1-gst-plugins-base gtk3
sudo port install gstreamer1-gst-plugins-bad
sudo port install gstreamer1-gst-plugins-good
sudo port install gstreamer1-gst-plugins-ugly zlib xz
sudo port install adwaita-icon-theme libchamplain
sudo port install autoconf automake clang-9.0 geoclue2
sudo port install geocode-glib gnome-common gtk-doc
sudo port install intltool itstool
sudo port install p5.28-xml-sax-expat pkgconfig yelp-tools
```

- Install latest source from gitlab.gnome.org

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

cd gingerblue/

./configure --prefix=/usr/local

make

sudo make install

[ENTER PASSWORD]

/usr/local/bin/gingerblue
```

Chapter 10

Results

The formal proof is the audio file that was recorded running GNOME Gingerblue 2.0.1 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 11

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”, 1978.

[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*

Application Letter to University of Copenhagen

To whom it may concern,

I have studied Computer Science (Object-oriented programming) and Mathematics (Linear Algebra) at University of Oslo since 1997 in my home city Oslo.

I have been building a network, maintaining network connectivity at Fjellbirkeland at University of Oslo since 1998-1999 and worked at Norwegian Computer Center (NR -- www.nr.no) in 2001-2004.

I have worked on building a commercial 400.000 domain network in Norway since 2003 (Domainnameshop -- www.domainnameshop.com) with Ståle Schumacher, Dag Fredrik Øien, and Jan Ingvoldstad.

My plans for advanced studies at University of Copenhagen is to complete my Bachelor of Science degree at University of Oslo in 2024 with 60 study points of Mathematics-Economics, Mathematics or Computer Science at a top Danish university in Copenhagen, where Hans Christian Ørsted worked on electricity in 1820.

I hope to further perfect my work at the Gingerblue project (www.gingerblue.org) before June 24, 2024.

Ole Kristian Aamot
www.gnomeradio.org
www.gingerblue.org
www.gnomevoice.org

GNOME Radio: http://www.oleaamot.no/omu/bachelor/Aamot,2020.pdf
Gingerblue: http://www.oleaamot.no/uio/bachelor/Aamot,2022.pdf
GNOME Voice: http://www.oleaamot.no/ntnu/bachelor/Aamot,2024.pdf