This document was written as accompanying material to a presentation at the 3rd International Linux Audio Conference 2005 in Karlsruhe, Germany.

Music Synthesis Under Linux

Tim Janik
University of Hamburg, Germany
timj@gtk.org

ABSTRACT

While there is lots of desktop software emerging for Linux which is being used productively by many end-users, this is not the case as far as music software is concerned. Most commercial and non-commercial music is produced either without software or by using proprietary software products. With BEAST, an attempt is made to improve the situation for music synthesis. Since most everything that is nowadays possible with hardware synthesizers can also be processed by stock PC hardware, it's merely a matter of a suitable implementation to enable professional music production based on free software. As a result, the development of BEAST focuses on multiple design goals. High quality demands are made on the mathematical characteristics of the synthesis, signals are processed on a 32-bit-basis throughout the program and execution of the synthesis core is fully real-time capable. Furthermore, the synthesis architecture allows scalability across multiple processors to process synthesis networks. Other major design goals are interoperability, so the synthesis core can be used by third-party applications, and language flexibility, so all core functionality can be controlled from script languages like scheme. In addition, the design of all components accounts for an intense focus on the graphical user interface to allow simple and if possible intuitive operation of the program.

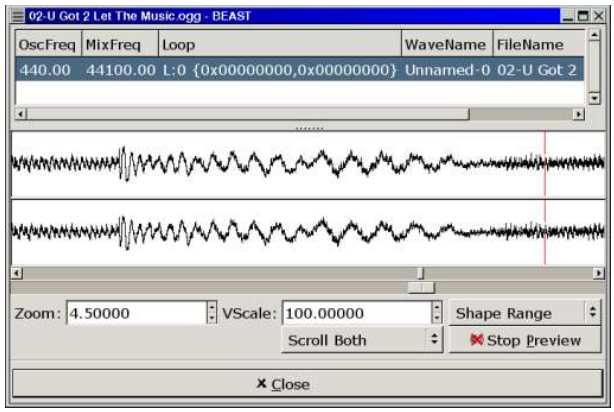
Keywords

Modular Synthesis, MIDI Sequencer, Asynchronous Parallel Processing, Pattern Editor.

1 BEAST/BSE - An Overview

BEAST is a graphical front-end to BSE which is a synthesis and sequencing engine in a separate shared library. Both are being released under the GPL and are being developed as free software for the best part of a decade. Since the first public release, some parts have been rolled out and reintegrated into other Projects, for instance the BSE Object system which became GObject in Glib. The programming interface of BSE is wrapped up by a glue layer, which allows for various language bindings. Currently a C binding exists which is used by BEAST. A C++ binding exists which is used to implement plugins for BSE and there also is a Scheme binding which is used for application scripting in BEAST or scripting BSE through the scheme shell bsesh.

BEAST allows for flexible sound synthesis and song composition based on utilization of synthesis instruments and audio samples. To store songs and synthesis settings, a special BSE specific hybrid text/binary file format is used which allows for seamless integration of audio samples, synthesis instruments and sequencing information.



Wave View Dialog

Since the 0.5 development branch, BEAST offers a zoomable time domain display of audio samples with preview abilities. Several audio file formats are supported, in particular MP3, WAV, AIFF, Ogg/Vorbis and BseWave which is a hybrid text/binary file format used to store multi samples with loop and other accompanying information. A utility for creation, compression and editing of BseWave files is released with version 0.6.5 of BEAST. Portions of audio files are loaded into memory on demand and are decoded on the fly

even for intense compression formats like Ogg/Vorbis or MP3. This allows for processing of very large audio files like 80 megabytes of MP3 data which roughly relates to 600 megabytes of decoded wave data or one hour of audio material. To save decoding processing power, especially for looped samples, decoded audio data is cached up to a couple of megabytes, employing a sensible caching algorithm that prefers trashing of easily decoded sample data (AIFF or WAV) over trashing processing intense data (Ogg/Vorbis).

The synthesis core runs asynchronously and performs audio calculations in 32-bit floating point arithmetic. The architecture is designed to support distribution of synthesis module calculations across multiple processors, in case multiple processors are available and the operating system supports process binding. In principle the sampling rate is freely adjustable, but it is in practice limited by operating system IO capabilities. The generated audio output can be recorded into a separate wave file.

The graphical user interface of BEAST sports concurrent editing of multiple audio projects, and unlimited undo/redo functionality for all editing functions. To easily give audio setups a try and for interactive alterations of synthesis setups, real-time MIDI events are processed. This allows utilization of BEAST as a ordinary MIDI synthesizer.

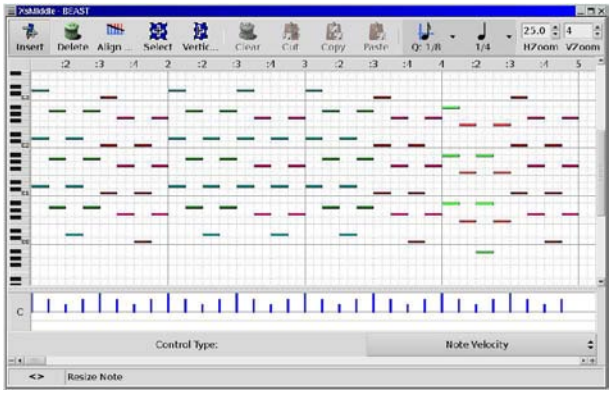
Since the complete programming interface of the synthesis core is available through a scheme shell, BEAST allows registration of scheme scripts at startup to extend its functionality and to automate complex editing tasks.

2 Song Composition

Songs consist of individual tracks with instruments assigned to them, and each track may contain multiple parts. A part defines the notes that are to be played for a specific time period.

The type of instrument assigned to a track is either a synthesis instrument, or an audio sample. Synthesis instruments are separate entities within the song's audio project and as such need to be constructed or loaded before use. In current versions, to enable sound compression or echo effects, post processing of audio data generated by a track or song is supported by assigning designated post processing synthesis meshes to them which simply act as ordinary audio filters, modifying the input signal before output.

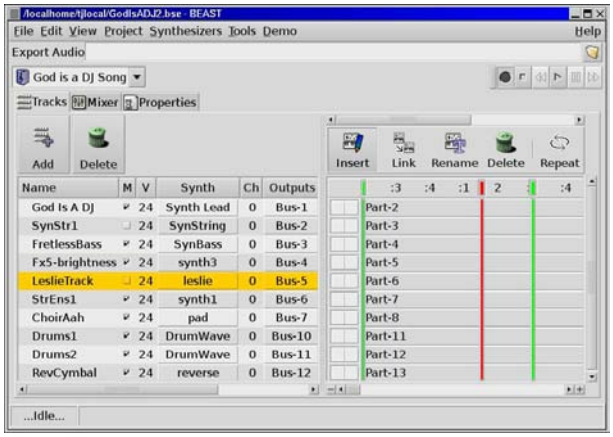
The post processing mechanism is currently being reworked, to integrate with the audio mixer framework that started shipping in recent versions of the 0.6 development branch. In the new audio mixer, audio busses can freely be created and connected, so volume metering or adjustment and effects processing is possible for arbitrary combinations of tracks and channels. Other standard features like muting or solo playback of busses are supported as well.



Piano Roll and MIDI Event Dialog

To allow editing of parts, a zoomable piano roll editor is supplied. Notes can be positioned by means of drag-and-drop in a two dimensional piano key versus time grid arrangement. This enables variation of note lengths and pitch through modification of note placement and graphical length. The piano keys also allow preview of specific notes by clicking on or dragging about. Also many other standard editing features are available via context menu or the toolbar, for instance note and event selection, cutting, pasting, insertion, quantization and script extensions. MIDI events other than notes, such as velocity or volume events can also be edited in an event editor region next to the piano roll editor. Newer versions of BEAST even sport an experimental pattern editor mode, which resembles well-known sound tracker interfaces. The exact integration of pattern mode editing with MIDI parts is still being worked out though.

Similar to notes within parts, the individual parts are arranged within tracks via drag-and-drop in the zoomable track view. Tracks also allow links to parts so a part can be reused multiple times within multiple tracks or a single track. The track view also offers editing abilities to select individual tracks to be processed by the sequencer, specification of the number of synthesis voices to be reserved and adding comments.



3 Synthesis Characteristics

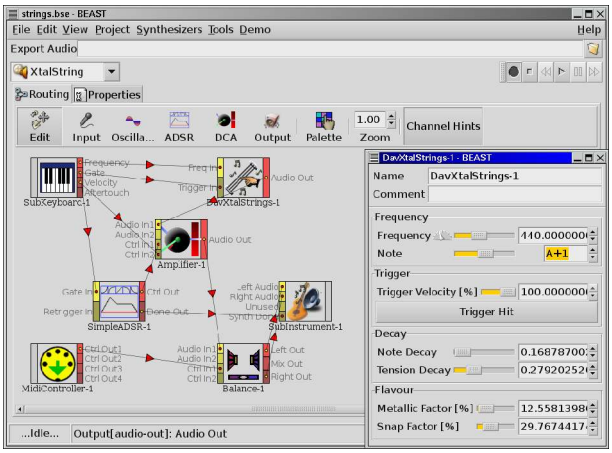
The synthesis facilities of the standard 0.6 development branch of the BEAST distribution, roughly equates the facilities of a simple modular synthesizer. However the quality and number of the supplied synthesis modules is constantly being improved.

Various synthesis modules are available. Amongst the audio sources are an Audio Oscillator, a Wave Oscillator, Noise, Organ and a Guitar Strings module. Routing functionality is implemented by modules like Mixer, Amplifier, ADSR-Envelope, Adder, Summation, Multiplier and Mini Sequencer. Various effect modules are also supplied, many based on recursive filters, i.e. Distortion, Reverb, Resonance, Chorus, Echos and the list goes on. Finally, a set of connection or IO modules is supplied for instrument input and output, MIDI input or synthesis mesh interconnection.

Apart from the synthesis modules shipped with the standard distribution, BSE also supports execution of LADSPA modules. Unfortunately, limitations in the LADSPA parameter system hinder seamless integration of LADSPA modules into the graphical user interface.

In general, the modules are implemented aliasing free, and highly speed optimized to allow real-time applicability. Per module, multiple properties (phase in an oscillator, resonance frequency of filters, etc...) are exported and can be edited through the user interface to alter synthesis functionality. A large part of mutable module parameters is exported through separate input or output channels, to allow for maximum flexibility in the construction of synthesis meshes.

BEAST generally does not differentiate between audio and control signals. Rather, the control or audio character of a signal is determined by the way of utilization through the user.



The graphical user interface provides for simple access to the construction and editing functionality of synthesis networks. Modules can be selected from a palette or context menu, and are freely placeable on a zoomable canvas. They are then connected at input and output ports via click-and-drag of connection lines. For each module, an information dialog is available and separate dialogs are available to edit module specific properties. Both dialogs are listed in the module context menu. Properties are grouped by functional similarities within editing dialogs, and many input fields support multiple editing metaphors, like fader bars and numeric text fields. All property and connection editing functions integrate with the project hosted undo/redo mechanism, so no editing mistakes committed can be finally destructive.

3.1 Voice-Allocation

The maximum number of voices for the playback of songs and for MIDI controlled synthesis can be specified through the graphical user interface. Increasing this number does not necessarily result in an increase in processor load, it just sets an upper limit within which polyphonic synthesis is carried out. To reduce processor load most effectively, the actual voice allocation is adjusted dynamically during playback time. This is made possible by running the synthesis core asynchronously to the rest of the application, and by preparing a processing plan which allows for concurrent processing of voice synthesis modules. This plan takes module dependencies into account which allows distribution of synthesis module processing tasks across multiple processors. Execution of individual processing branches of this plan can be controlled with sample granularity. This allows suspension of module

branches from inactive voices. The fine grained control of processing activation which avoids block quantization of note onsets allows for precise realization of timing specifications provided by songs.

4 User experience and documentation

Like with most audio and synthesis applications, BEAST comes with a certain learning curve for the user to overcome. However, prior use of similar sequencing or synthesis applications may significantly contribute to reduction of this initial effort. The ever growing number of language translations can also be of significant help here, especially for novice users. BEAST does not currently come with a comprehensive manual, but it does provide a "Quick Start" guide which illustrates the elementary editing functions, and the user interface is equipped with tooltips and other informative elements explaining or exemplifying the respective functionality. Beyond that, development documentation for the programming interfaces, design documents, an FAQ, Unix manual pages and an online "Help Desk" for individual user problems are provided, accessible through the "Help" menu.

5 Future Plans

Although BEAST already provides solid functionality to compose songs and work with audio projects, there is still a long list of todo items for future development.

Like with any free software project with an open development process, we appreciate contributions and constructive criticism, so some of the todo highlights are outlined here:

- Extend the set of standard instruments provided.
- Implement more advanced effect and distortion modules.
- Adding a simple GUI editor for synthesis mesh skins.
- Implementing new sound drivers, e.g. interfacing with Jack.
- New instrument types are currently being worked on such as GUS Patches.
- Support for internal resampling is currently in planning stage.
- Extending language bindings and interoperability.

6 Acknowledgements

Our thanks go to the long list of people who have contributed to the BEAST project over the years.

7 Internet Addresses

BEAST home page:
http://beast.gtk.org
Contacts, mailing list links, IRC channel:
http://beast.gtk.org/contact.html
Open project discussion forums:
http://beast.gtk.org/wiki:BeastBse

8 Abbreviations and References

ADSR – *Attack-Decay-Sustain-Release*, envelope phases for volume shaping.
BEAST - *Bedevilled Audio System*, http://beast.gtk.org.
BSE - *Bedevilled Sound Engine*.
C++, C - Programming languages, http://www.research.att.com/~bs/C++.html.
FAQ – *Frequently Asked Questions*.
GLib - *Library of useful routines for C programming*, http://www.gtk.org.
GObject - GLib object system library.
GPL - *GNU General Public License*, http://www.gnu.org/licenses/gpl.html.
GUI – *Graphical User Interface*.
GUS Patch – *Gravis Ultrasound Patch* audio file format.
IRC – *Internet Relay Chat*.
Jack - *Jack Audio Connection Kit*, http://jackit.sourceforge.net.
LADSPA - *Linux Audio Developer's Simple Plugin API*, http://www.ladspa.org.
MIDI - *Musical Instruments Digital Interface*, http://www.midi.org/about-midi/specshome.shtml.
MP3, WAV, AIFF - sound file formats, http://beast.gtk.org/links-related.html.
Ogg/Vorbis - open audio codec, http://www.xiph.org/ogg/vorbis.