Software Development Kit Overview

SDK Version 1.0

Contents

Code Examples

Figures

Tables

1 Goals of the NINTENDO GAMECUBE SDK

During development of the NINTENDO GAMECUBE (GCN) console, we noted that game developers had many difficult tasks to accomplish in a short period of time in order to development and release a game. Therefore, we designed the NINTENDO GAMECUBE SDK to provide useful, flexible software components and numerous examples, our goal being to help game developers familiarize themselves with the machine as quickly as possible.

We know that game developers must code and select all of the software necessary to make their games, so the GCN SDK is really just a list from which developers can choose the components and features that they find useful. Simple software components, designed specifically for video games, give game developers flexibility in deciding how to use various system resources.

2 Development hardware

2.1 Programmer development system

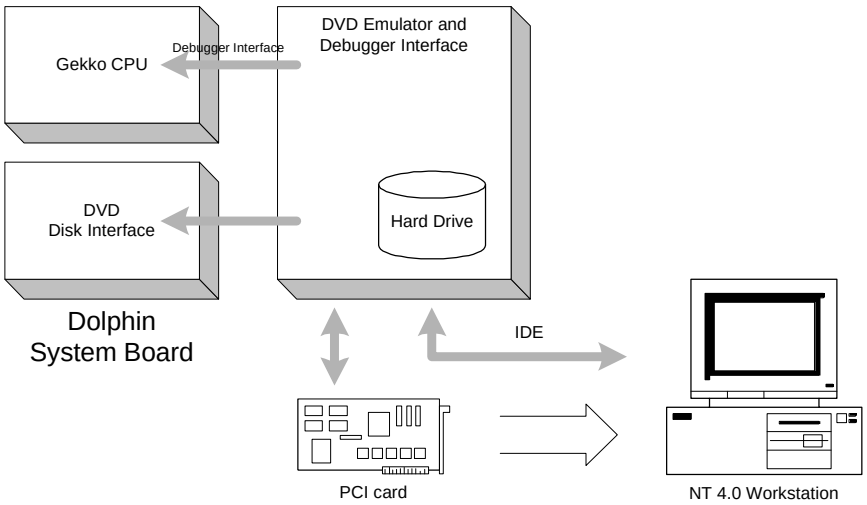

Figure 1 Dolphin Development Hardware (DDH) system blocks

2.1.1 Goals

The goals of the Dolphin Development Hardware (DDH) are:

- Accurate emulation of console hardware.
- Rapid program change/debug sessions.
- Extra memory to enhance capabilities of development tools.
- Reasonable cost so that each programmer can have a dedicated station.

2.1.2 Key features

Key features of the DDH include:

- Parity of all functional blocks between the development system and the final console.
- DVD drive performance emulation and file system emulation.
- Double main memory size (48MB).

Note that some of these features may not be available on initial versions of the DDH.

2.1.3 Disk emulation

The disk emulation system provides the following features:

- The hard drive connects directly to the PC, which enables high speed updates of data files and program files to the game's data DVD disk data sets.

- Emulation of correct seek and transfer timing, using disk geography information supplied by the developer to define file location on the optical disk.
- Translation of Windows FAT symbolic file system to NINTENDO GAMECUBE DVD file system by generating a symbol table describing the directory and file hierarchy.
- Error emulation (e.g., “scratched disk” messages).

Note that some of these features may not be available on the initial versions of the DDH.

2.2 DDH/PC host communication interface

The DDH can facilitate game development through its ability to communicate directly with the PC. We provide a software library to access the hardware interface linking the DDH’s EXI0 or EXI1 interface with the PC’s USB interface.

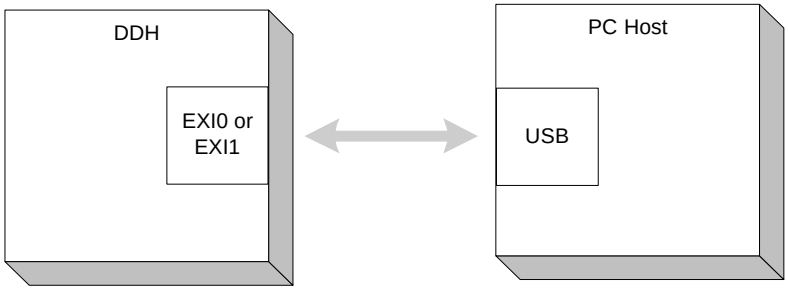

Figure 2 DDH/PC host communication interface

2.3 Software bug testing system – TBD

To be described in a future release.

3 SDK components

Component Type	Components
Compiler Suite	Metrowerks CodeWarrior compiler/debugger suite SN System compiler/debugger suite
Operating System	GCN operating system DVD file system Game controller pad library
Graphics	3D graphics library Matrix library 2D graphics library Video display library Demonstration library Texture conversion tools Geometry conversion tools Articulated character animation libraries and tool
Audio	Wavetable construction tool Sound effects design tool Wavetable and sound effects synthesis library DVD streaming library

Table 1 SDK components

4 Compiler and debugger suites

The GCN has two compiler/debugger suites from which to choose. Both suites include the following components:

- C/C++ language compiler.
- PPC assembler, including the new Gekko CPU instruction set.
- Standard C libraries.
- Debugger.
- Command line compiler for the Makefile build environment.
- IDE build environment.

Currently, only the Metrowerks CodeWarrior compiler and debugger suite is available. The SN Systems tools will be provided shortly.

5 Build environment

The GCN SDK ships with many demonstration programs to show how to use different library components. These demos all use the Makefile build environment from the Cygnus Cygwin UNIX-style command shell and make environment. We supply the freeware Cygwin package on the SDK CD-ROM. You can also download the latest Cygwin distribution from http://sourceware.cygnus.com/cygwin/.

For more details, please read “Build System” in the *NINTENDO GAMECUBE Programmer’s Guide*.

6 Operating system

6.1 Memory address map

During the execution of a game, we need to switch quickly between cached and uncached access. The operating system accesses most code and data through cached addresses for speed, but it must access hardware registers as well. For this reason, we use a memory address map similar to the Nintendo 64 (N64) MIPS memory map. There are separate memory address ranges for cached access and uncached access.

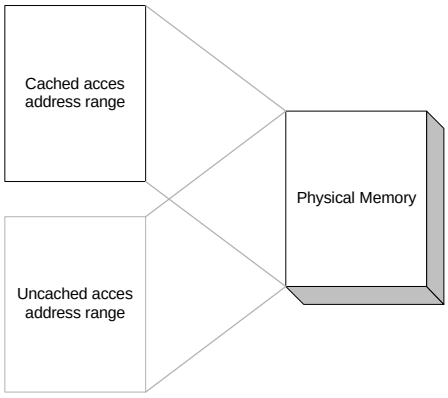

Figure 3 Cached and uncached memory address map

6.2 Execution model

The GCN supports threads and interrupt event callback handlers. Game developers can choose the most suitable and familiar execution models for specific games.

Threads have message queues and conditional variable synchronization functions. Using threads can cause critical sections in reentrant code. The GCN OS provides mutual execution functions to protect these critical code sections.

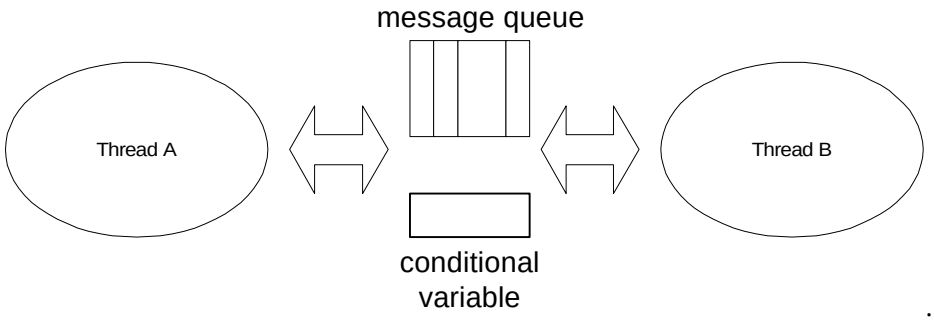

Figure 4 Thread communication

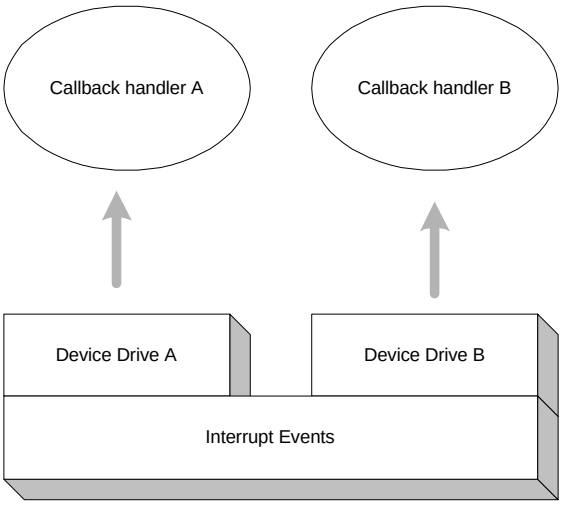

Figure 5 Interrupt event callback handler

6.3 Utility functions

We provide the following utility functions for game development:

Utility Library	Purpose
Memory allocation	Basic multiple heap memory allocation library.
Standard C libraries	Provided by compiler suite. Handle math, string, buffer file IO, …
Stopwatch, Alarms	Stopwatch handles elapsed-time capture, accurate to 50Mhz resolution. Alarms provide countdown timer functionality.
Dynamic linking	TBD

Table 2 Utility libraries

6.4 DVD file system

The SDK provides file system access for the large-capacity DVD drive. This file system includes the following feature set:

- Symbolic character-string names for files, directories, and paths.
- Variably-sized files.
- Non-blocking and blocking access methods (i.e., asynchronous and synchronous access, respectively).

6.4.1 Random access comparison of DVD drive to mask ROM

On the N64, many game developers randomly accessed the mask ROM to load sound and animation data into the main memory. The latency in this case was negligible, making it possible on the N64 to load tens of thousands of files per second. The GCN's DVD optical drive will have a random-access latency of over 100 milliseconds (i.e., over six 60Hz frames). This means it will be impossible to load more than 10 non-sequential files per second on The GCN.

Therefore, we suggest that developers used to N64 programming take time to consider the critical issue of disk access time. In order to minimize random-access disk seeks, it is very important that you plan out how your game will access data. In particular, merging data into large blocks is essential.

The GCN DVD disk emulation system provides tools to specify disk file placement; however, these tools cannot solve every problem. If a large number of files need to be loaded—and the files are not contiguous on the disk—the system will take an unacceptably long time to load the files. Therefore, the developer should plan early in the game design process to utilize the massive capacity of the optical disk while minimizing the effect of the longer seek time.

Feature	Optical Disk	Mask ROM
Random access	100 milliseconds $1/10^{th}$ of a second	Few microseconds
Capacity	Very Large	8MB

Table 3 Optical disk and mask ROM comparison

7 Graphics

7.1 Graphics library (GX)

The GCN Graphics library (GX) has functions to render geometry with many attributes. The GX library's main purpose is to provide the logical API that game engines need to perform rendering, and it is implemented as a thin layer of code above the hardware to ensure highest performance. If you are familiar with OpenGL, you will find that the GX library is similar to it in many respects.

7.1.1 Drawing geometry

The GX library has two main methods for drawing geometry. The first method is very much like OpenGL immediate mode, with functions that look like this:

```
GXBeginTriangle();
GXPosition(x, y, z);
GXColor(r, g, b, a);
…
```

Code 1 GX library functions

This immediate mode API is ideal when the CPU must synthesize geometry data from a higher-level description (e.g., a height field or Bezier patch). These calls are inline functions and the compiler performs excellent optimization.

The second method sends geometry directly to the Graphics Processor (GP) by way of a memory-resident display list format. This method offers superior performance for non-animated data.

The immediate mode API and the Display List (DL) format both support configurable vertex representations, which in turn support:

- Direct or indexed vertex components. Vertex components (position, normal, color, and each texture coordinate) may all be indexed from arrays independently, or placed in the memory stream directly.

- Each vertex component can have a different-sized representation and precision. The available direct types are: **8-bit signed and unsigned integer**, **16-bit signed and unsigned integer**, and **32-bit floating point**. A scale is available to position the decimal point for the integer types. The indirect types **8-bit index** or **16-bit index** can be used to index into an array of any of the direct types.

This flexible representation allows the game developer to organize vertex data in a way that is appropriate for specific games. The ability to index each component separately eliminates a great deal of data duplication.

7.1.2 Geometry processing control

For geometry processing, the GX library contains functions to define and/or set the following:

- Local lighting.
- Modelview matrices and the projection matrix.
- Backface and view-frustum clipping.
- Viewport.
- Texture coordinate projection mappings.

- Reflection mapping.
- Bump maps.

7.1.3 Texture application

For the application of textures, the GX library contains functions for these operations:

- Define texture objects (bitmap location, wrap and mirror parameters, filter type).
- Load a Color Look-Up Table (CLUT).
- Configure Graphics Processor (GP) TMEM texture caches.
- Set multiple texture combine operators (TEV).

7.1.4 Other pixel operations

The GX library supports many pixel operations, including:

- Antialiasing.
- Z buffer control.
- Blending.
- Fog.

7.1.5 Miscellaneous functions

The GX library also contains functions to provide:

- CPU-to-GP FIFO control.
- Performance counters.

7.2 Matrix-Vector library (MTX)

The GCN SDK provides a Matrix-Vector library (MTX) to perform common matrix operations. It can:

- Construct matrices by common parameters (translation, rotation, scale, quaternion).
- Perform typical 3D vector operations (transformation, dot product, cross product, normalization).
- Build forward and inverse matrix stacks.

The GCN SDK includes the source code to this library.

7.3 Demonstration library (DEMO)

The Demonstration library (DEMO) performs simple configuration of system resources, which sometimes requires the use of functions from several different libraries. For example, one of the main functions of the DEMO library is to set video display resolution. To set a simple video resolution configuration, we must:

- Allocate the correct size of frame buffer to match the video resolution.
- Set graphics viewports to the correct size.
- Set video mode correctly.

The GCN SDK includes the source code to this library.

7.4 2D Graphics library (G2D)

The 2D Graphics library (G2D) supports:

- Multiple image layers.
- Image formatting with tiles (like the SNES-type of 2D consoles).
- 2D rotation of images.
- Bitmap tile sorting for fast display (i.e., efficient use of the texture cache).
- Viewport clipping for fast display.

The GCN SDK includes the source code to this library.

7.5 Character Pipeline (articulated animation set)

The GCN SDK provides a set of example libraries and tools to perform articulated animation. These libraries and tools are intended as a guide for developers who wish to learn the basic feature set of the hardware. By no means do they represent a complete solution for game development. These tools and libraries perform the following functions:

- Extract texture for the graphics hardware (using `TexConv`/TC).
- Extract geometry for the graphics hardware (use `MaxConv`/C3).
- Extract hierarchy and animation tracks for character animation (use `MaxConv`/C3).
- Display geometry and texture using the GX library.
- Animate characters.

For easy reference, we call this suite of tools and libraries the "Character Pipeline" (see the *NINTENDO GAMECUBE Character Pipeline & CG Tools Guide* for detailed information).

7.5.1 Data extraction libraries and tools

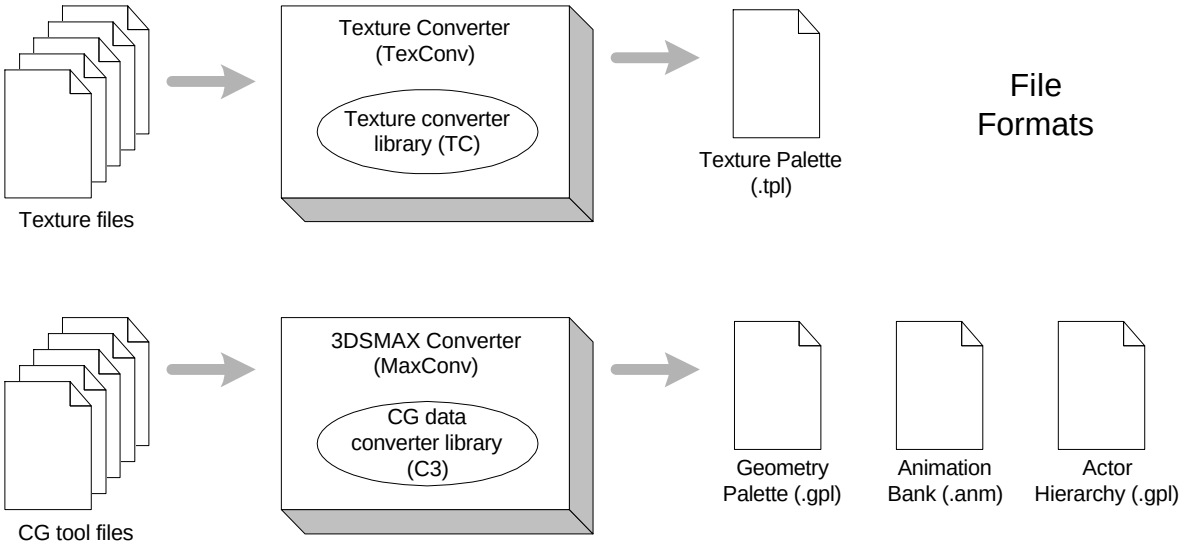

Figure 6 Character Pipeline data extraction path

During development of the NINTENDO GAMECUBE system, we observed that game developers use many different CG tools to make games. Because computer graphics tool vendors release new suites fairly frequently, we were faced with the daunting proposition of trying to provide a ready-made converter for every CG tool available. Of course this

was impracticable, so we resolved instead to provide game developers with an easy method for making their own converters. As a result, the NINTENDO GAMECUBE SDK provides both converter tools *and* a converter library:

- **TC** – Texture extraction and conversion library.
- **TexConv** – Texture converter which uses the TC library for texture conversion.
- **C3** – 3D geometry and animation data extraction and conversion library.
- **MaxConv** – 3D Studio MAX converter which uses the C3 library for conversion.

Both of the libraries are very easy to use and have simple APIs to load data. Developers can use a CG tool's data extraction API and then insert this data into these libraries to generate a GCN output format (GPL, TPL, ACT, ANI). Here is an example of the C3 API:

```
C3BeginPolyPrimitive();
C3BeginVertex();
C3SetPosition(x, y, z);
C3SetColor(r, g, b, a);
```

Code 2 3D geometry conversion API

7.5.1.1 Texture conversion tool (TexConv/TC)

The Graphics Processor can support:

- Trilinearly-filtered mipmaps.
- S3TC-compressed textures.

The GP requires that the texture image be organized in a tiled format with correct alignment and padding of image data. The GCN SDK provides a texture converter tool and library to convert textures to Graphics Processor formats. The texture converter tool is called `TexConv`, the texture converter library is called TC. `TexConv` and TC both use the TGA (`.tga`) file format. The TGA format can support RGBA, monochrome intensity, color-indexed, and CLUT data, so it is very useful for games.

The GCN SDK includes the source code to `TexConv` and the TC library.

7.5.1.1.1 TexConv and TC library data optimization

`TexConv` and TC performs the following operations on textures:

- Tiling and padding required to prepare data for graphics hardware.
- S3TC texture format encoding.
- Mipmap generation.

7.5.1.2 Geometry conversion tool (MaxConv/C3)

The GCN SDK provides `MaxConv`, a 3D Studio MAX converter plug-in tool, to convert 3D Studio MAX data sets to the GCN geometry format. The SDK also provides the C3 converter library which can be used to build other CG tool converters.

The GCN SDK includes the source code to both the `MaxConv` plug-in and the C3 library.

7.5.1.2.1 MaxConv and C3 library data optimization

The C3 library performs many data optimization features. Here is a short list of speed and memory optimizations:

- Triangle-strip creation.
- Elimination of common vertex component data using vertex component indexing.
- Quantization of floating point to fixed-point numbers to reduce storage requirements; e.g., quantization on position, color, normal, and texture coordinate data.
- Minimized matrix loads for stitched character skin animation.

7.5.2 Runtime libraries

The Character Pipeline includes many runtime libraries to support animated character display. These libraries load the character animation file formats and handle the hierarchical animation of a game character.

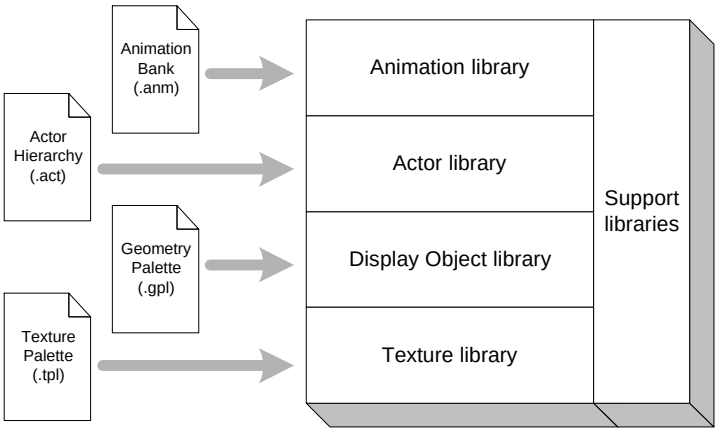

Figure 7 Character Pipeline runtime libraries

7.5.2.1 Display Object library (DO)

The DO library handles the loading, 3D-space manipulation, and display of a static graphical object. The goal of this library is to enable artists to construct any object they would like to display. The DO library supports:

- Different textures for different groups of faces.
- Different shading (flat, Gouraud) for different groups of faces.
- Different polygon display optimizations (triangle strips, quads).

7.5.2.2 Actor library (ACT)

The ACT library handles the loading and display of a hierarchical 3D actor. (3D hierarchy is sometimes referred to as the skeleton.)

7.5.2.3 Animation library (ANM)

The Animation library handles the loading and sequencing of animation for an actor. The ANM library will support:

- Multiple sequences (e.g., “walk,” “run,” “jump”).
- Keyframing and a variety of interpolation methods, including: Tension/Continuity/Bias (TCB), Bezier (smooth), Linear, and Quaternion SLERP.

8 Audio

8.1 Audio and graphics game framework

The GCN has a dedicated DSP for audio synthesis. Developers used to N64 programming may be glad to learn that they no longer have to write the DSP scheduler to manage time-sharing of the DSP processor between graphics and audio.

Note, however, that the GCN CPU is shared between graphics and audio. The CPU is interrupted every five milliseconds so that the audio library can generate commands for the DSP.

Audio processing is not synchronized to graphics processing, so there should be no difference in sound between NTSC and PAL formats.

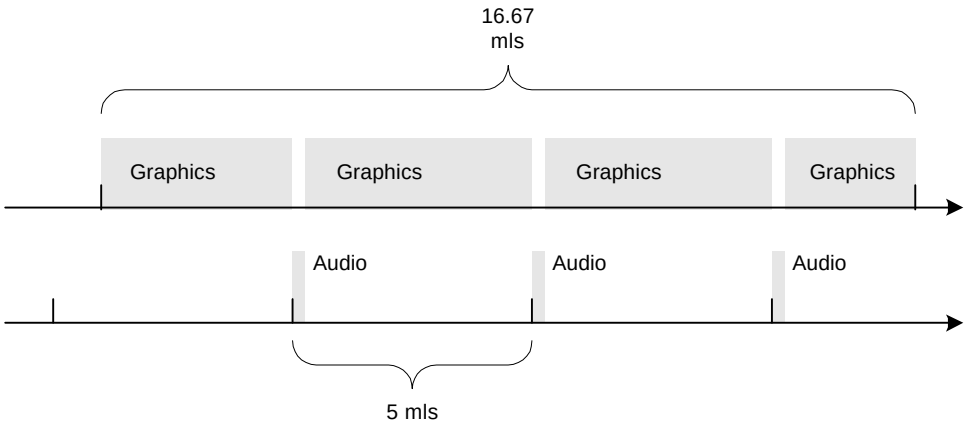

Figure 8 Independent scheduling for CPU processing of audio and graphics

8.2 Factor5 MusyX sound system

The GCN uses Factor 5's MusyX sound system with specific enhancements for this console. You can learn more about MusyX on Factor 5's web site: http://www.factor5.com/musyx.htm.

The MusyX sound system has these features:

- Graphical user interface tool to construct wavetables and preview MIDI scores.
- Runtime API library for interactive playback.
- Wavetable synthesis with ADPCM-compressed wavetables (i.e., no ADPCM-frame restrictions on loops.)
- 3D audio.
- Dolby Surround Sound.
- Reverberation.
- Macro language enabling the musician to generate sounds procedurally.
- DLS wavetable input format which allows use of the DLS standard to provide data exchange between many wavetable construction/editing tools.

The synthesizer has the following quality and performance characteristics:

- Wavetable synthesis parameters that update every 1 millisecond.
- 64 channels.

- Multiple auxiliary busses (effects channels).
- Output sample rate at 32kHz Dolby surround encoded.

8.3 Sound sets

MusyX will ship two sounds sets:

- General MIDI wavetable from Roland's Sound Canvas product.
- Factor5's wavetable.

8.3.1 Roland wavetable

The Roland General MIDI wavetable in DLS format can be imported into MusyX tools. This wavetable set features samples at 22.050kHz, 226 instruments, and eight drum sets. Both 8-bit and 16-bit wavetables are supplied with MusyX.

