SVFIG Presentation
21 November 2003
Cogswell Polytechnical College

Forth Music Project

Implement a set of platform-independent tools in Forth for custom audio and MIDI production

Windows Platform

Multimedia API / DirectX (DirectSound & DirectMusic)

Multimedia API

1. MCI audio
2. Low-level PCM audio
3. Low-level MIDI
4. Interrupt-driven timer services
5. Joystick capture

MIDI (Musical Instrument Digital Interface – MIDI 1.0 Specifiction,1984)

Protocol for remote control of digital musical instruments

Controller–>slave model

Messages

Channel mode – device modes: mono/poly/omni

Channel voice – notes, controllers, pgm. change, etc.

System exclusive – device configuration data

System real time – timing clock pulses

System common – MTC (MIDI time code)

Channel voice message

Status Data Data

Data byte range 0 - 7F

Status byte ranges

Messages sent over 16 channels

80 - 8F	Note off
90 - 9F	Note on
A0 - AF	Polyphonic Key Pressure (Aftertouch)
B0 - BF	Control Change

C0 - CF	Program change
D0 - DF	Channel pressure
E0 - EF	Pitch bend

System Requirements

- MIDI hardware interface or internal MIDI configuration in software
- MIDI drivers
- MIDI applications or programming capabilities

API architecture for Low-level MIDI

- MIDI input functions
- MIDI output functions

MIDI output set

Poll number of MIDI out devices	midiOutGetNumDevs()
Get device capabilities	midiOutGetDevCaps()
Open MIDI out device	midiOutOpen()
Close MIDI out device	midiOutClose()
Send MIDI short message (Status-Data-Data)	midiOutShortMessage()
Send system exclusive message	midiOutLongMessage()

```
typedef struct midioutcaps_tag {
    UINT        wMid;
    UINT        wPid;
    VERSION     vDriverVersion;
    char        szPname[MAXPNAMELEN];          // 32 bytes
    UINT        wTechnology;
    UINT        wVoices;
    UINT        wNotes;
    UINT        wChannelMask;
    DWORD       dwSupport;
} MIDIOUTCAPS;
```

Implementation in Swift Forth

```
/* Byte packing function from MS MMAPI documentation */
UINT FAR PASCAL sendMIDIEvent( HMIDIOUT hMidiOut, BYTE bStatus,
                                                    BYTE bData1, BYTE bData2 )
{
        union {
                DWORD           dwData;
                BYTE            bData[4];
        } u;

        u.bData[0] = bStatus;
        u.bData[1] = bData1;
        u.bData[2] = bData2;
        u.bData[3] = 0;

        return midiOutShortMsg( hMidiOut, u.dwData );
}
```

144 60 100 (note on note # velocity)

(144 * 1) + (60 * 256) + (100 * 65536) = 6569104

128 60 0 (note off note# velocity)

(128 * 1) + (60 * 256) +) = 15488

```
: pack-bytes 65536 * swap 256 * + + ; ( n n n -- n )
```