SKILL UP!

将来にわたって使えるスキルを身につけるために

オブジェクト指向ではじめるプログラミング

日向 俊二
HYUGA, Shunji

第11回 C#でMIDIを扱う（その2）

Technology Tools

- ❑ Visual Basic .NET
- ☑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ☑ Other:
  .NET Framework SDK

Level

★ ★ ★ ★ ★

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、付録CD-ROMの¥DOTNET¥CSHARPディレクトリに収録しています。

¥MIDIKBRD
キーボードプログラム

※この記事では.NET Framework SDKを利用して解説していますが、C#Builder、SharpDevelopなどでも利用できます（C#コンパイラならすべて利用できます）。

前回のおさらい

前回は、コンピュータの画面上に表示して、マウスで操作する、単純な「MIDIキーボード」アプリケーションを作りました。そのために必要なこととして、MIDIの概要や、.NET Frameworkを使ったC#のプログラムから従来のWindowsのDLLの中にあるWin32 APIの関数を呼び出す方法について説明しました。さらに、ユーザーコントロールの作成についても概説しました。

前回のサンプルでは、ひとつのアプリケーションを構成するプログラムコードを、3つのクラスに分けて記述し、それぞれのクラスの内容を別のファイルに保存しました。

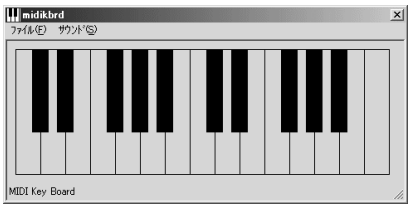
今回は、前回のサンプルにコードを追加して機能を増やし、より興味深いプログラムにすると共に、プログラム全体の構成や細部について検討してみましょう。

サンプルmidikbrd

今回のサンプルmidikbrdは、前回のサンプルMidiKBにコードを追加して作成します。そのため、外観は前回のサンプルMidiKBにそっくりです（図1）。ただし、［ファイル］メニューの隣に［サウンド］メニューを作成して新たな機能を追加しています。

midikbrdのメニューの構成を表1に示します。表1のメニュー項目の中には、見慣れない言葉があるかもしれません。これらについてはコラムを参照してください。音楽にあまり興味を持っていない人は詳細を知る必要はありませんが、知っておくとあとの説明やプログラムコードの理解に役立ちます。

図1：サンプルmidikbrd



★1　C#では、ひとつのファイルに複数のクラスを記述することもできます。しかし、まったく種類の異なる機能を持つクラスは、通常、別のファイルに記述するほうがよいでしょう。

プログラムの構造を見る

さて、メニューができて、追加するべき内容が決まったら、続いてコードを追加すればよさそうです。しかし、ここですぐにコードを追加せずに、ちょっと考えてみることが大切です。前回はプログラムを整理したり再検討したりすることができなかったので、ここでもとのプログラム全体を振り返ってみましょう。そのうえで、必要なコードを追加することにします。

さて、MidiKBのC#のソースファイルは表2に示す3つでした。そして、さらに、MakefileとMidiKB.icoを使い、「MidiKB.exe」を生成しました。

クラスの働きとWindowsや.NET Frameworkとの関係は、概ね図2に示すようになります。MidiKBでファイルを3つに分けたのは、偶然ではなく、図2の①～⑤に示すように、使用する機能（.NET FrameworkのベースクラスやWin32 API）とクラスに含める機能の違いを、ファイルという形で物理的に分離したためであることがわかるでしょう★1。

今回のサンプルmidikbrdでも、この構造を生かしたまま、コードを追加し

表1：midikbrdのメニュー

メニュー	サブメニュー	名前
ファイル(&F)		mnuFile
	終了(&X)	mnuFileExit
サウンド(&S)		mnuSound
	単音(&S)	mnuSoundSingle
	和音(&H)	mnuSoundTriad
	-	mnuSpacer1
	オールオフ(&O)	mnuSoundOff
	-	mnuSpacer2
	密集配置	mnuSoundClosed
	分散配置	mnuSoundOpen

※-はスペーサー。

表2：MidiKBのC#のソースファイル

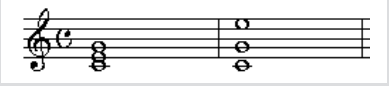
ファイル名	内容
frmMain.cs	メインフォームのクラス
KBCntrl.cs	キーボードコントロールのクラス
Win32MIDI.cs	Win32 APIのMIDI関数を呼び出すためのクラス

COLUMN　サンプルmidikbrdのための音楽マメ知識

今回のサンプルmidikbrdのメニューにある「単音」は、前回のサンプルMidiKBで実現したように、一度にひとつの音を鳴らすことを意味します。一方、一度に複数の音を鳴らすと、「和音」になります。今回のサンプルでは、長3和音（一番下の音がドのときはド,ミ,ソの和音）だけを使いますが、短3和音（ド,♭ミ,ソ）や、セブンス（7の和音、ド,ミ,ソ,♭シ）、不協和音まで含めて、さまざまな種類の和音があります。今回のサンプルは興味があればそれらを追加できるように考えてあります。詳細は付録CD-ROMのソースファイルのコメントを参照してください。

「オールオフ」はMIDIのオールサウンドオフメッセージに対応します。これは使っているMIDIデバイスで鳴らしている音をすべて止めます。普通は必要なさそうな機能ですが、MIDIの仕様上、タイミングや何らかのエラーの影響などで音が鳴りっぱなしになる可能性は否定できません。そのため、主要なMIDIアプリケーションの多くは、必要に応じてユーザーが手動でオールサウンドオフまたはオールノートオフをデバイスに送って鳴っている音を即座に止められるように作られています。

「密集配置」と「分散配置」は和声（ハーモニー）に関する用語です。密集配置（Closed voicing）は、図A左（1小節目）に示すように一番下の音（根音）から和音の構成音を3度間隔で順に上に重ねたものを指します。分散配置は、解離配置（Open voicing）ともいい、図A右（2小節目）に示すように和音の構成音を分散させて重ねたものを指します。

図A：「密集配置」と「分散配置」

同じ和音であっても密集配置にするか、分散配置にするかで、耳に聞こえる響きが異なります（この場合は、同じドミソからなる和音ですが、分散配置にしたほうが高い音まで含む豊かな響きに聞こえることでしょう）。興味があれば、さらにさまざまな配置（一番下の音が根音以外の構成音となる、和音の転回形など）も鳴らせるように追加して、響きの違いを楽しんでみてください。

なお、音楽における「度」は簡単にいえば音階上の2音間の距離で、3度というのはド-ミの間隔、5度はド-ソの間隔です。