TMidiGen

——Properties Methods Installation alan.warriner@bigfoot.com www.alan-warriner.co.uk

TMidiGen is a MIDI component by Alan Warriner which allows the simple creation of sound effects and note sequences within an application without the need for external files or resources. TMidiGen achieves this by generating the MIDI data in memory.

TMidiGen provides:

- 175 instruments from any General MIDI capable sound card.
- Advanced Pitch Bend, Modulation, Sustain, Chorus and Reverb effects.
- Easy Volume and Pan adjustment.
- Simple method of playing individual notes.
- Note sequences can be assembled in strings to provide chords and tunes.
- Sound duration variable in 10mS steps
- Easy creation of sound loops.

TMidiGen Properties

——TMidiGen
In TMidiGen

Instrument
Note
Octave
Duration
PitchBend
PBDelay
PBDuration
Sustain
Loops
Modulation
Chorus
Reverb
Volume
Pan
Channel
IsPlaying

Installation

TMidiGen

Delphi 4

Unzip the Delphi archive.

Copy the MidiGen.dcu and TMidiGen.dcp files to the ..Delphi\Lib folder.

Copy TMidiGen.bpl to ..Delphi\Bin folder

Start Delphi and select:

Component -> Install Packages

Click on *Add*

Browse to the location of TMidiGen.bpl (..Delphi\Bin)

Select TMidiGen.bpl

OK all the way back to Delphi

TMidiGen should now be installed on the Samples Palette.

Other Versions Of Delphi

Extract the Delphi Component Source.zip files (MidiGen.pas, TMidiGen.res & TMidiGen.dpk) into a folder.

Start Delphi

Select the *File -> Open Project* menu option.

Select files of type Delphi Package (*.dpk) from the File Open dialog.

Browse to the folder where the files were un-zipped.

Select TMidiGen.dpk

From the small dialog which appears select *Compile*.

The compiler will complain that it needs VCLx.y or something similar, select *OK*.

When compilation is complete select *Install*.

C++Builder 4

Unzip the C++Builder archive to a folder of your choice.

Copy TMidiGen.bpl to a folder on the IDE search path (e.g. ...CBuilder\Bin).

Copy MidiGen.hpp to ..CBuilder\Include or equivalent.

Copy TMidiGen.bpi and TMidiGen.lib to ...CBuilder\Lib or equivalent.

Start C++Builder and select:

Component -> Install Packages.

Click on *Add*

Browse to the folder where TMidiGen.bpl was copied.

Select the file TMidiGen.bpl and click *OK*.

OK all the way back to C++Builder.

TMidiGen should now be installed on the Samples Palette.

Other Versions Of C++Builder

Extract the C++Builder Component Source.zip files (TMidiGen.cpp, MidiGen.hpp, MidiGen.pas, TMidiGen.bpk & TMidiGen.res) into a folder.

Start C++Builder.

Select the *File -> Open Project* menu option.

Browse to the folder where the files were un-zipped.

Select TMidiGen.bpk

From the small dialog which appears select *Compile*.

The compiler should complain that it needs VCLx.y or something similar, select *OK*.

When compilation is complete select *Install*.

Conversion Between Delphi And C++Builder When Using This Help File

Delphi Example	*C++Builder Example*
property AProperty: Integer;	__property int AProperty;
procedure SomeProc;	void __fastcall SomeProc();
procedure SomeProc(SomeVal: int);	void __fastcall SomeProc(int SomeVal);
function SomeFunc(): Integer;	int __fastcall SomeFunc();
function SomeFunc(SomeVal: int): Integer	int __fastcall SomeFunc(int SomeVal);
SomeComponent.SomeProperty;	SomeComponent->SomeProperty;

'a string constant';	"a string constant";
Assignment :=	Assignment =
Comparison =	Comparison ==

TMidiGen Methods

TMidiGen

In TMidiGen

Play

PlayString

PlayNote

Stop

MidiAvailable

SetDevice

TMidiGen.Instrument

TMidiGen Example

Determines which instrument will be used.

Type TMGInstrument = (mgAcousticGrandPiano,mgBrightAcousticPiano, ..., mgOpenTriangle);

property Instrument: TMGInstrument;

Description

Selects the instrument which will be used to generate the sound when Play and PlayNote methods are called, and the default instrument for PlayString. Values above *mgGunshot* (127) are percussion instruments.

Full TMGInstrument List

0 - mgAcousticGrandPiano 1 - mgBrightAcousticPiano 2 - mgElectricGrandPiano,
3 - mgHonkyTonkPiano 4 - mgElectricPiano1 5 - mgElectricPiano2
6 - mgHarpsichord 7 - mgClavinet 8 - mgCelesta
9 - mgGlockenspiel 10 - mgMusicBox 11 - mgVibraphone
12 - mgMarimba 13 - mgXylophone 14 - mgTubularBells
15 - mgDulcimer 16 - mgDrawbarOrgan 17 - mgPercussiveOrgan
18 - mgRockOrgan 19 - mgChurchOrgan 20 - mgReedOrgan
21 - mgAccordion 22 - mgHarmonica 23 - mgTangoAccordion
24 - mgAcousticNylonGuitar 25 - mgAcousticSteelGuitar 26 - mgJazzElectricGuitar
27 - mgCleanElectricGuitar 28 - mgMutedElectricGuitar 29 - mgOverdrivenGuitar
30 - mgDistortionGuitar 31 - mgGuitarHarmonics 32 - mgAcousticBass
33 - mgFingeredElectricBass 34 - mgPickedElectricBass 35 - mgFretlessBass
36 - mgSlapBass1 37 - mgSlapBass2 38 - mgSynthBass1
39 - mgSynthBass2 40 - mgViolin 41 - mgViola
42 - mgCello 43 - mgContrabass 44 - mgTremoloStrings
45 - mgPizzicatoStrings 46 - mgOrchestralHarp 47 - mgTimpani
48 - mgStringEnsemble1 49 - mgStringEnsemble2 50 - mgSynthStrings1
51 - mgSynthStrings2 52 - mgChoirAahs 53 - mgVoiceOohs
54 - mgSynthVoice 55 - mgOrchestraHit 56 -

mgTrumpet
57 - mgTrombone 58 - mgTuba 59 - mgMutedTrumpet
60 - mgFrenchHorn 61 - mgBrassSection 62 - mgSynthBrass1
63 - mgSynthBrass2 64 - mgSopranoSax 65 - mgAltoSax
66 - mgTenorSax 67 - mgBaritoneSax 68 - mgOboe
69 - mgEnglishHorn 70 - mgBassoon 71 - mgClarinet
72 - mgPiccolo 73 - mgFlute 74 - mgRecorder
75 - mgPanFlute 76 - mgBlownBottle 77 - mgShakuhachi
78 - mgWhistle 79 - mgOcarina 80 - mgSquareLead
81 - mgSawtoothLead 82 - mgCalliopeLead 83 - mgChiffLead
84 - mgCharangLead 85 - mgVoiceLead 86 - mgFifthsLead
87 - mgBassandLead 88 - mgNewAgePad 89 - mgWarmPad
90 - mgPolySynthPad 91 - mgChoirPad 92 - mgBowedPad
93 - mgMetallicPad 94 - mgHaloPad 95 - mgSweepPad
96 - mgSynthFXRain 97 - mgSynthFXSoundtrack 98 - mgSynthFXCrystal
99 - mgSynthFXAtmosphere 100 - mgSynthFXBrightness 101 - mgSynthFXGoblins
102 - mgSynthFXEchoes 103 - mgSynthFXSciFi 104 - mgSitar
105 - mgBanjo 106 - mgShamisen 107 - mgKoto
108 - mgKalimba 109 - mgBagpipe 110 - mgFiddle
111 - mgShanai 112 - mgTinkleBell 113 - mgAgogo
114 - mgSteelDrums 115 - mgWoodblock 116 - mgTaikoDrum
117 - mgMelodicTom 118 - mgSynthDrum 119 - mgReverseCymbal
120 - mgGuitarFretNoise 121 - mgBreathNoise 122 - mgSeashore
123 - mgBirdTweet 124 - mgTelephoneRing 125 - mgHelicopter
126 - mgApplause 127 - mgGunshot

Percussion Instruments

128 - mgAcousticBassDrum 129 - mgBassDrum1 130 - mgSideStick
131 - mgAcousticSnare 132 - mgHandClap 133 - mgElectricSnare
134 - mgLowFloorTom 135 - mgClosedHiHat 136 - mgHighFloorTom
137 - mgPedalHiHat 138 - mgLowTom 139 - mgOpenHiHat
140 - mgLowMidTom 141 - mgHiMidTom 142 - mgCrashCymbal1

143 - mgHighTom
144 - mgRideCymbal1
145 - mgChineseCymbal
146 - mgRideBell
147 - mgTambourine
148 - mgSplashCymbal
149 - mgCowbell
150 - mgCrashCymbal2
151 - mgVibraslap
152 -mgRideCymbal2
153 -mgHiBongo
154 - mgLowBongo
155 -mgMuteHiConga
156 -mgOpenHiConga
157 - mgLowConga
158 -mgHighTimbale
159 -mgLowTimbale
160 - mgHighAgogo
161 -mgLowAgogo
162 -mgCabasa
163 - mgMaracas
164 -mgShortWhistle
165 -mgLongWhistle
166 - mgShortGuiro
167 -mgLongGuiro
168 -mgClaves
169 - mgHiWoodBlock
170 -mgLowWoodBlock
171 -mgMuteCuica
172 - mgOpenCuica
173 -mgMuteTriangle
174 -mgOpenTriangle

Example

//This example assumes the current form contains a TMidiGen component named MidiGen1.

```
//set note to C# in octave 3
MidiGen1.Note:=mgCsharp;
MidiGen1.Octave:=3;

//0.5 seconds
MidiGen1.Duration:=50;

//modulation at 50%
MidiGen1.Modulation:=50;

//use a Violin
MidiGen1.Instrument:=mgViolin;

//play the note
MidiGen1.Play;

//snare drum hit, note property doesn't matter for percussion
MidiGen1.Instrument:=mgAcousticSnare;
MidiGen1.Play;
```

TMidiGen.Note

TMidiGen Example

Selects the desired note to play.

Type TMGNote
(mgC,mgCsharp,mgD,mgDsharp,mgE,mgF,mgFsharp,mgG,mgGsharp,mgA,mgAsharp,mgB);

property Note: TMGNote;

Description

This property determines which note will be played when the Play method is called. Works in conjunction with the Octave property.

Valid range is mgC in Octave 0 to mgG in Octave 10.

NB: If a percussion instrument is selected in the Instrument property then the note property has no effect. Percussion instruments are not 'tuned' ans so all note values will sound the same.

If Channel 10 is selected in the Channel property then the Note value will select which percussion instrument plays, in keeping with the General MIDI standard.

TMidiGen.Channel

TMidiGen

Selects which MIDI channel will be used to play sounds.

Type TMGChannel (mgCh01,mgCh02, ... ,mgCh16);

property Channel: TMGChannel;

Description

This property determines which channel the MIDI data will be played on.

NB If a percussion instrument is selected in the Instrument property then Channel 10 (mgCh10) will automatically be selected to play that instrument.

TMidiGen.Duration

TMidiGen Example

Duration determines the length of time for which the selected tone will sound.

property Duration: Cardinal;

Description

This is the duration, in seconds x 0.01, that the selected sound will be played. Minimum value is 0.01 seconds, any entry less than this will be rounded up to the minimum setting.

TMidiGen.Octave

TMidiGen Example

This property determines which octave the selected note is in.

property Octave: Integer;

Description

Used in conjunction with the Note property to determine the octave for the current note. Is also the default initial octave for the PlayString method.

Valid range is 0 to 10.

TMidiGen.PitchBend

TMidiGen Example

Determines the level of the pitch bend effect.

property PitchBend: Integer;

Description

This property controls the level of the pitch bend effect over a range of ±12 semi-tones in steps of a 0.1 semi-tones.

Valid range is -120 to +120.

See also PBDelay and PBDuration.

Example

//This example assumes the current form contains a TMidiGen component named MidiGen1.

//set note to D in octave 5
MidiGen1.Note:=mgD;
MidiGen1.Octave:=5;

// 4 seconds
MidiGen1.Duration:=400;

//use a Trumpet
MidiGen1.Instrument:=mgTrumpet;

//bend up 2 semi-tones
MidiGen1.PitchBend:=20;

//delay for a second (25% of Duration) before pitch bend starts)
MidiGen1.PBDelay:=25;

//pitch bend will be complete after 3 seconds (66% of remaining time after PBDelay)
MidiGen1.PBDuration:=66;

//play the note
MidiGen1.Play;

//bend down an octave
MidiGen1.PitchBend:= -120;

//no delay before pitch bend starts
MidiGen1.PBDelay:=0;

//pitch bend will be spread over full duration
MidiGen1.PBDuration:=100;

//play E in the third octave
MidiGen1.PlayString('E3');

TMidiGen.PBDelay

TMidiGen Example

This property determines the delay before the pitch bend effect is applied.

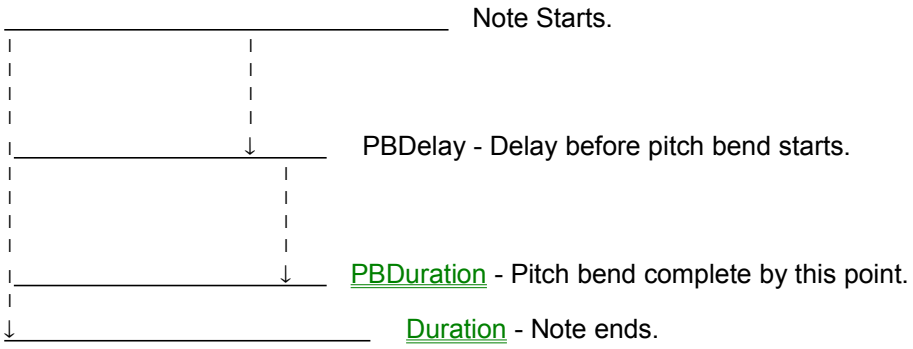
property PBDelay: Integer;

Description

PBDelay is the percentage of the Duration property that the component waits before applying the pitch bend effect.

Valid range is 0 to 100.

See also PitchBend and PBDuration.



TMidiGen.PBDuration

TMidiGen Example

Determines the time span over which the pitch bend effect is applied.

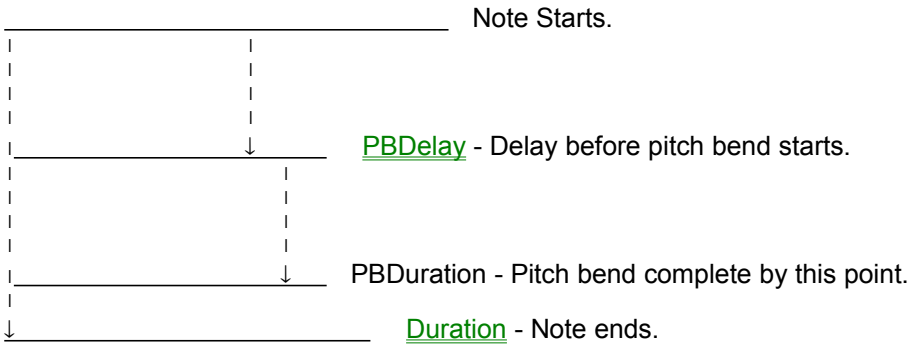
property PBDuration: Integer;

Description

PBDuration is the percentage of the Duration time remaining, after the PBDelay period, over which the pitch bend effect is applied.

Valid range is 0 to 100.

See also PitchBend and PBDelay.



TMidiGen.Sustain

TMidiGen

Determines whether sustain is applied to notes.

property Sustain: bool;

Description

The sustain effect prolongs the amount of time a note is audible.

TMidiGen.Loops

TMidiGen

The Loops property allows a sound to play a number of times or continuously.

property Loops: Cardinal;

Description

The selected note or sequence will repeat the number of times specified in this property. A value of 0 will cause the sound to loop continuously until another sound is started or the Stop method is called.

TMidiGen.Modulation

TMidiGen Example

Sets the level for the modulation effect.

```
property Modulation: Integer;
```

Description

Sets the modulation effect level as a percentage of its maximum.

Valid range is 0 to 100.

TMidiGen.Chorus

TMidiGen

Sets the level for the chorus effect.

property Chorus: Integer;

Description

Sets the chorus effect level as a percentage of its maximum.

Valid range is 0 to 100.

TMidiGen.Reverb

TMidiGen

Sets the level for the reverb effect.

property Reverb: Integer;

Description

Sets the reverb effect level as a percentage of its maximum.

Valid range is 0 to 100.

TMidiGen.Volume

TMidiGen

Sets the volume level.

```
property Volume: Integer;
```

Description

Sets the volume as a percentage of its maximum.

Valid range is 0 to 100.

TMidiGen.Pan

TMidiGen Example

This property determines the stereo position of the selected sound.

property Pan: Integer;

Description

Sound position can be set from full left (-100) full right (+100).

Valid range is -100 to +100.

Example

//This example assumes the current form contains a TMidiGen component named MidiGen1.

```
//set note to A# in octave 6
MidiGen1.Note:= mgAsharp;
MidiGen1.Octave:= 6;

//play out of left speaker only
MidiGen1.Pan:= -100;

//play the note
MidiGen1.Play;

//now play out of right speaker only
MidiGen1.Pan:= 100;

//play again
MidiGen1.Play;
```

TMidiGen.Play

TMidiGen Example

The Play method will sound the selected note.

```
function Play: bool;
```

Description

Plays the currently selected Note and applies any effects selected.

The note will play for the length of time specified in Duration parameter, and will repeat the number of times specified in the Loops property.

Return value is true if the note could be played or false if an error occurred.

TMidiGen.PlayString

TMidiGen Example

The PlayString method will play the notes specified in the string parameter.

function PlayString(Notes: **String**): bool;

Description

PlayString allows a series of notes and instrument changes to be specified in a string, and played.

The Notes string can contain codes to; play a note, alter a note's delay and duration, and select an instrument.

Entries in the Notes string are separated by commas.

Return value is true if note could be played or false if an error occurred.

NB PitchBend will only have an effect if all events within the string have a delay section of zero and are all of the same duration.

String Format

The full format for an entry in the string is:

Delay ; Event ; Duration

The Delay and Duration sections are optional. If omitted the Delay value will default to the most recent value specified in the string, or zero if none has been specified yet. The default Duration is either the most recently specified, or the value in the Duration property.

The event section can either be a note specifier or an instrument change selection. Notes are selected using a note and octave combination or an absolute note number.

Note and Octave Combination Example

C#4 - C sharp in the 4th octave. *PlayString('C#4');*

D- - D flat in the most recently specified octave or the Octave property if no octave has been specified yet. *PlayString('D-');*

Absolute Note Example

N123 - Note 123 which is the equivalent of the parameter passed in the PlayNote method. *PlayString('N123');*

Instrument Selection

I69 - Selects instrument 69 (EnglishHorn). *PlayString('I69');*

Example

//This example assumes the current form contains a TMidiGen component named MidiGen1.

//play a 3 note chord
MidiGen1.PlayString('C4, E, G');

//same chord, different method
MidiGen1.PlayString('N48, N52, N55');

//play the same 3 notes 0.5 seconds apart
MidiGen1.PlayString('C, 50;E, G');

//play 2 notes on the Harpsichord followed by one on the Tuba
MidiGen1.PlayString('I6, C, 50;E, I58, 0;G');

//play a 3 note chord in octave 2 followed by a three note chord in octave 5 one second later
MidiGen1.PlayString('C2, E, G, 100;C5, 0;E, G');

//play a 4 second note on the Drawbar Organ overlaid with 2 notes delayed by 1 second of 0.5 second duration on the Tubular Bells
MidiGen1.PlayString('I16, C;400, 100;I14;50, E, G');

//play 3 notes of 0.3 second duration separated by 0.5 seconds, followed by the same notes separated by 1 second
MidiGen1.PlayString('C;30, 50;E, G, 100;C, E, G');

TMidiGen.Stop

TMidiGen
Halts the playing of the sound.

procedure Stop;

Description

Stop will immediately silence any note or note sequence which is playing.

The Stop method is used to terminate a note which is looping when the Loops property is set to a value other than 1.

TMidiGen.IsPlaying

TMidiGen Example

A read-only property which indicates if the TMidiGen component is busy.

property IsPlaying: bool;

Description

If IsPlaying is true then component is still playing MIDI data. A false result indicates that the sound has terminated.

Example

//This example assumes the current form contains a TMidiGen component named MidiGen1.

//play a long duration note
MidiGen1.Duration:=500;
MidiGen1.Play;

//wait for it to stop before returning
while MidiGen1.IsPlaying Do;

TMidiGen.MidiAvailable

TMidiGen Example

Indicates if the current MIDI device is available.

```
function MidiAvailable: bool;
```

Description

If MidiAvailable returns true then the current MIDI device is available for use.

A return value of false indicates that another program has locked the MIDI device or no MIDI hardware is installed.

Example

//This example assumes the current form contains a TMidiGen component named MidiGen1.

```
//MIDI available?
if MidiGen1.MidiAvailable then
      begin
       //make a noise
      MidiGen1.Play;
      end
else
      begin
      //what else can you do?
      Beep;
      end;
```

TMidiGen.PlayNote

TMidiGen

The PlayNote method will sound the note specified by the NoteValue parameter.

```
function PlayNote(NoteValue: Integer) bool;
```

Description

Plays the note passed to the function in the NoteValue parameter. Note values range from 0 to 127.

A NoteValue of 0 is equivalent to C in octave 0, and 127 to G in octave 10.

The note will play for the length of time specified in Duration parameter, and will repeat the number of times specified in the Loops property.

Return value is true if the note could be played or false if an error occurred.

Example

//This example assumes the current form contains a TMidiGen component named MidiGen1.

//play D# in the 5th octave
MidiGen1.Note:=mgDsharp;
MidiGen1.Octave:=5;
MidiGen1.Play;

//this code has exactly the same effect
MidiGen1.PlayNote(63);

TMidiGen.SetDevice

TMidiGen

Allows the MIDI device to be changed.

```
function SetDevice(Device: Integer): bool;
```

Description

The default MIDI device is the MIDI Mapper (Device= -1).

SetDevice allows the MIDI device to be changed to the device number specified in the parameter passed to the function.

A return value of true indicates that the device number was valid. False indicates an illegal device number.