

Midi Player Tool Kit for Unity

Thierry Bachmann
Version 2.85
Thu Oct 22 2020

Table of Contents

Namespace Index	1
Hierarchical Index	1
Class Index	1
MidiPlayerTK.....	2
Class Documentation.....	8
MidiExternalPlayer.....	8
MidiFileLoader.....	10
MidiFilePlayer	16
MidiFileWriter2.....	27
MidiInReader.....	34
MidiKeyboard	36
MidiListPlayer.....	39
MidiListPlayer.MPTK_MidiPlayItem.....	46
MidiLoad	48
MidiPlayerGlobal	54
MidiSpatializer	60
MidiStreamPlayer.....	61
MidiSynth.....	64
MPTKChordBuilder	84
MPTKChordLib	87
MPTKEvent.....	89
MPTKRangeLib	94
Index.....	96

Namespace Index

Packages

Here are the packages with brief descriptions (if available):

MidiPlayerTK	2
------------------------------------	---

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

MidiFileLoader.....	10
MidiFileWriter2	27
MidiKeyboard	36
MidiListPlayer.....	39
MidiListPlayer.MPTK_MidiPlayItem.....	46
MidiLoad.....	48
MidiPlayerGlobal	54
MidiSynth.....	64
MidiFilePlayer	16
MidiExternalPlayer	8
MidiSpatializer.....	60
MidiInReader	34
MidiStreamPlayer	61
MPTKChordBuilder.....	84
MPTKChordLib	87
MPTKEvent	89
MPTKRangeLib.....	94

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

MidiExternalPlayer ([MPTK PRO] - Script associated to the prefab MidiExternalPlayer .. Play a midi file from a path on the local desktop or from a web site. There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector)	8
MidiFileLoader (Script associated to the prefab MidiFileLoader . No sequencer, no synthetizer, no music playing capabilities. Usefull to load all or part of the Midi events from	

a Midi and process, transform, write them to what you want. List of Midi file must be defined with Midi Player Setup (see Unity menu MPTK))	10
MidiFilePlayer (Script associated to the prefab MidiFilePlayer . Simply, play a Midi file. Midi files must be defined from the Unity menu MPTK in the Unity editor. There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector)	16
MidiFileWriter2 ([MPTK PRO] - Write a midi file from differents sources based on NAudio framework. See full example TestMidiWriter.cs with a light sequencer. This class replaced MidiFileWriter with these changes: channel start at 0, new specific event, better control. More information here: https://paxstellar.fr/class-midifilewriter2/)	27
MidiInReader ([MPTK PRO] - Script associated to the prefab MidiInReader . Read Midi events from a Midi keyboard connected your device (Windows 10 or MacOS). See example of use in TestMidiInputScripting.cs There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector)	34
MidiKeyboard (Send and receive Midi Message from a Midi keyboard connected to the device https://paxstellar.fr/class-midkeyboard/)	36
MidiListPlayer ([MPTK PRO] - Script for the prefab MidiListPlayer . Play a list of pre-selected midi file from the dedicated inspector. List of Midi files must exists in MidiDB. See Midi Player Setup (Unity menu MPTK))	39
MidiListPlayer.MPTK MidiPlayItem (Define a midi to be added in the list)	46
MidiLoad (Class for loading a Midi file. No sequencer, no synthetizer, so music playing capabilities. Usefull to load all the Midi events from a Midi and process, transform, write them to what you want.	48
)	
MidiPlayerGlobal (Singleton class to manage all global features of MPTK. More information here: https://paxstellar.fr/midiplayerglobal/)	54
MidiSpatializer ([MPTK PRO] - Script associated to the prefab MidiSpatializer . It's quite light because the major job is done with MidiSynth There is no specific API for this prefab. Scripting is necessary to defined position of channel or instrument in your 3D env)	60
MidiStreamPlayer (Play generated notes. Any Midi file is necessary rather create music from your own algorithm with MPTK_PlayEvent() . Duration can be set in the MPTKEvent , but a note can also be stopped with MPTK_StopEvent())	61
MidiSynth ()	64
MPTKChordBuilder ([MPTK PRO] Chord builder class for MPTK. Usefull to generate Midi Music with MidiStreamPlayer - V2.82 See example in TestMidiStream.cs and ExtStreamPlayerPro.cs)	84
MPTKChordLib ([MPTK PRO] - Load library of chord from ChordLib.csv in folder Resources/GeneratorTemplate.csv - V2.82 new)	87
MPTKEvent (Midi Event class for MPTK. Use this class to generate Midi Music with MidiStreamPlayer or to read midi events from a Midi file with MidiLoad or to receive midi events from MidiFilePlayer OnEventNotesMidi. With this class, you can: play and stop a note, change instrument (preset, patch, ...), change some control as modulation)	89
MPTKRangeLib ([MPTK PRO] - Load library of scale from GammeDefinition.csv in folder Resources/GeneratorTemplate.csv - V2.82 new)	94

Namespace Documentation

MidiPlayerTK Namespace Reference

Classes

- class [MidiExternalPlayer](#)

[MPTK PRO] - Script associated to the prefab [MidiExternalPlayer](#). Play a midi file from a path on the local desktop or from a web site. There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.

- class [MidiFileLoader](#)
Script associated to the prefab [MidiFileLoader](#). No sequencer, no synthesizer, no music playing capabilities. Usefull to load all or part of the Midi events from a Midi and process, transform, write them to what you want. List of Midi file must be defined with Midi Player Setup (see Unity menu MPTK).
- class [MidiFilePlayer](#)
Script associated to the prefab [MidiFilePlayer](#). Simply, play a Midi file. Midi files must be defined from the Unity menu MPTK in the Unity editor. There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.
- class [MidiFileWriter2](#)
[MPTK PRO] - Write a midi file from different sources based on NAudio framework. See full example [TestMidiWriter.cs](#) with a light sequencer. This class replaced [MidiFileWriter](#) with these changes: channel start at 0, new specific event, better control. More information here: <https://paxstellar.fr/class-midifilewriter2/>
- class [MidiInReader](#)
[MPTK PRO] - Script associated to the prefab [MidiInReader](#). Read Midi events from a Midi keyboard connected your device (Windows 10 or MacOS). See example of use in [TestMidiInputScripting.cs](#) There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.
- class [MidiKeyboard](#)
Send and receive Midi Message from a Midi keyboard connected to the device <https://paxstellar.fr/class-midikeyboard/>
- class [MidiListPlayer](#)
[MPTK PRO] - Script for the prefab [MidiListPlayer](#). Play a list of pre-selected midi file from the dedicated inspector. List of Midi files must exists in MidiDB. See Midi Player Setup (Unity menu MPTK).
- class [MidiLoad](#)
Class for loading a Midi file. No sequencer, no synthesizer, so music playing capabilities. Usefull to load all the Midi events from a Midi and process, transform, write them to what you want.
- class [MidiPlayerGlobal](#)
Singleton class to manage all global features of MPTK. More information here: <https://paxstellar.fr/midiplayerglobal/>
- class [MidiSpatializer](#)
[MPTK PRO] - Script associated to the prefab [MidiSpatializer](#). It's quite light because the major job is done with [MidiSynth](#) There is no specific API for this prefab. Scripting is necessary to defined position of channel or instrument in your 3D env.
- class [MidiStreamPlayer](#)
Play generated notes. Any Midi file is necessary rather create music from your own algorithm with [MPTK_PlayEvent\(\)](#). Duration can be set in the [MPTKEvent](#), but a note can also be stopped with [MPTK_StopEvent\(\)](#).
- class [MidiSynth](#)
- class [MPTKChordBuilder](#)
[MPTK PRO] Chord builder class for MPTK. Usefull to generate Midi Music with [MidiStreamPlayer](#) - V2.82 See example in [TestMidiStream.cs](#) and [ExtStreamPlayerPro.cs](#)
- class [MPTKChordLib](#)
[MPTK PRO] - Load library of chord from [ChordLib.csv](#) in folder [Resources/GeneratorTemplate.csv](#) - V2.82 new
- class [MPTKEvent](#)

Midi Event class for MPTK. Use this class to generate Midi Music with [MidiStreamPlayer](#) or to read midi events from a Midi file with [MidiLoad](#) or to receive midi events from [MidiFilePlayer](#) [OnEventNotesMidi](#). With this class, you can: play and stop a note, change instrument (preset, patch, ...), change some control as modulation

- class [MPTKRangeLib](#)
[MPTK PRO] - Load library of scale from GammeDefinition.csv in folder Resources/GeneratorTemplate.csv - V2.82 new

Enumerations

- enum [MPTKCommand](#) : byte { [NoteOff](#) = 0x80, [NoteOn](#) = 0x90, [KeyAfterTouch](#) = 0xA0, [ControlChange](#) = 0xB0, [PatchChange](#) = 0xC0, [ChannelAfterTouch](#) = 0xD0, [PitchWheelChange](#) = 0xE0, [Sysex](#) = 0xF0, [Eox](#) = 0xF7, [TimingClock](#) = 0xF8, [StartSequence](#) = 0xFA, [ContinueSequence](#) = 0xFB, [StopSequence](#) = 0xFC, [AutoSensing](#) = 0xFE, [MetaEvent](#) = 0xFF }
MIDI command codes. Defined the action to be done with the message: note on/off, change instrument, ... Depending of the command selected, others properties must be set; Value, Channel,
- enum [MPTKController](#) : byte { [BankSelect](#) = 0, [Modulation](#) = 1, [BreathController](#) = 2, [FootController](#) = 4, [MainVolume](#) = 7, [Pan](#) = 10, [Expression](#) = 11, [BankSelectLsb](#) = 32, [Sustain](#) = 64, [Portamento](#) = 65, [Sostenuto](#) = 66, [SoftPedal](#) = 67, [LegatoFootswitch](#) = 68, [ResetAllControllers](#) = 121, [AllNotesOff](#) = 123, [AllSoundOff](#) = 120 }
MidiController enumeration <http://www.midi.org/techspecs/midimessages.php#3>
- enum [MPTKMeta](#) : byte { [TrackSequenceNumber](#) = 0x00, [TextEvent](#) = 0x01, [Copyright](#) = 0x02, [SequenceTrackName](#) = 0x03, [TrackInstrumentName](#) = 0x04, [Lyric](#) = 0x05, [Marker](#) = 0x06, [CuePoint](#) = 0x07, [ProgramName](#) = 0x08, [DeviceName](#) = 0x09, [MidiChannel](#) = 0x20, [MidiPort](#) = 0x21, [EndTrack](#) = 0x2F, [SetTempo](#) = 0x51, [SmpteOffset](#) = 0x54, [TimeSignature](#) = 0x58, [TimeSignature](#) = 0x58, [KeySignature](#) = 0x59, [SequencerSpecific](#) = 0x7F }
MIDI MetaEvent Type

Enumeration Type Documentation

enum [MPTKCommand](#) : byte [strong]

MIDI command codes. Defined the action to be done with the message: note on/off, change instrument, ... Depending of the command selected, others properties must be set; Value, Channel,

Enumerator:

NoteOff	Note Off
NoteOn	Note On. Value contains the note to play between 0 and 127.
KeyAfterTouch	Key After-touch

ControlChange	Control change. Controller contains identify the controller to change (Modulation, Pan, Bank Select ...). Value will contains the value of the controller between 0 and 127.
PatchChange	Patch change. Value contains patch/preset/instrument value between 0 and 127.
ChannelAfterTouch	Channel after-touch
PitchWheelChange	Pitch wheel change
Sysex	Sysex message
Eox	Eox (comes at end of a sysex message)
TimingClock	Timing clock (used when synchronization is required)
StartSequence	Start sequence
ContinueSequence	Continue sequence
StopSequence	Stop sequence
AutoSensing	Auto-Sensing
MetaEvent	Meta-event

enum [MPTKController](#) : byte[strong]

MidiController enumeration <http://www.midi.org/techspecs/midimessages.php#3>

Enumerator:

BankSelect	Bank Select (MSB)
Modulation	Modulation (MSB)
BreathController	Breath Controller
FootController	Foot controller (MSB)
MainVolume	Main volume
Pan	Pan
Expression	Expression
BankSelectLsb	Bank Select LSB ** not implemented **
Sustain	Sustain
Portamento	Portamento On/Off
Sostenuto	Sostenuto On/Off
SoftPedal	Soft Pedal On/Off
LegatoFootswitch	Legato Footswitch
ResetAllControllers	Reset all controllers
AllNotesOff	All notes off

AllSoundOff	All sound off

enum [MPTKMeta](#) : byte [strong]

MIDI MetaEvent Type

Enumerator:

TrackSequenceNumber	Track sequence number
TextEvent	Text event
Copyright	Copyright
SequenceTrackName	Sequence track name
TrackInstrumentName	Track instrument name
Lyric	Lyric
Marker	Marker
CuePoint	Cue point
ProgramName	Program (patch) name
DeviceName	Device (port) name
MidiChannel	MIDI Channel (not official?)
MidiPort	MIDI Port (not official?)

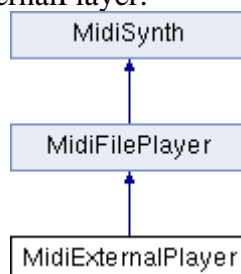
EndTrack	End track
SetTempo	Set tempo
SmpteOffset	SMPTE offset
TimeSignmature	Time signature (typo error, deprecated!)
TimeSignature	Time signature
KeySignature	Key signature
SequencerSpecific	Sequencer specific

Class Documentation

MidiExternalPlayer

[MPTK PRO] - Script associated to the prefab [MidiExternalPlayer](#).. Play a midi file from a path on the local deskop or from a web site. There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.

Inheritance diagram for MidiExternalPlayer:



Public Member Functions

- new [MidiLoad](#) [MPTK_Load](#) ()
Not applicable for external

- new void [MPTK_Next \(\)](#)
Not applicable for external
- override void [MPTK_Play \(\)](#)
Play the midi file defined in MPTK_MidiName
- new void [MPTK_Previous \(\)](#)
Not applicable for external

Properties

- new int [MPTK_MidiIndex](#) [get, set]
Not applicable for external
- new string [MPTK_MidiName](#) [get, set]
*Full path to Midi file or URL to play. Must start with [file://](#) or [http://](#) or [https://](#). Example:
MPTK_MidiName="http://www.midiworld.com/midis/other/c2/bolero.mid";*

Additional Inherited Members

Detailed Description

[MPTK PRO] - Script associated to the prefab [MidiExternalPlayer](#).. Play a midi file from a path on the local deskop or from a web site. There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.

```
// Example of script. See TestMidiExternalPlayer.cs for a more detailed usage.
// Need for a reference to the Prefab (to be set in the hierarchy or can be done by
script)
MidiExternalPlayer midiExternalPlayer;

if (midiExternalPlayer==null)
    Debug.LogError("TestMidiExternalPlayer: there is no MidiExternalPlayer Prefab set
in Inspector.");

midiExternalPlayer.MPTK_MidiName =
"http://www.midiworld.com/midis/other/c2/bolero.mid";
midiExternalPlayer.MPTK_Play();
!
```

Member Function Documentation

new [MidiLoad](#) MPTK_Load ()

Not applicable for external

new void MPTK_Next ()

Not applicable for external

override void MPTK_Play () [virtual]

Play the midi file defined in MPTK_MidiName

Reimplemented from [MidiFilePlayer](#).

new void MPTK_Previous ()

Not applicable for external

Property Documentation

new int MPTK_MidiIndex [get], [set]

Not applicable for external

new string MPTK_MidiName [get], [set]

Full path to Midi file or URL to play. Must start with [file://](#) or [http://](#) or [https://](#). Example:
MPTK_MidiName="http://www.midiworld.com/midis/other/c2/bolero.mid";

MidiFileLoader

Script associated to the prefab [MidiFileLoader](#). No sequencer, no synthetizer, no music playing capabilities. Usefull to load all or part of the Midi events from a Midi and process, transform, write them to what you want. List of Midi file must be defined with Midi Player Setup (see Unity menu MPTK).

Inherits MonoBehaviour.

Public Member Functions

- void [MPTK_Load](#) (byte[] midiBytesToLoad=null)
Load the midi file defined with MPTK_MidiName or MPTK_MidiIndex or from a array of bytes
- void [MPTK_Next](#) ()
Read next Midi from the list of midi defined in MPTK (see Unity menu Midi)
- [MPTKEvent.EnumLength](#) [MPTK_NoteLength](#) ([MPTKEvent](#) note)
Return note length as https://en.wikipedia.org/wiki/Note_value

- void [MPTK_Previous](#) ()
Read previous Midi from the list of midi defined in MPTK (see Unity menu Midi)
- List< [MPTKEvent](#) > [MPTK_ReadMidiEvents](#) (long fromTicks=0, long toTicks=long.MaxValue)
Read the list of midi events available in the Midi from a ticks position to an end position.

Public Attributes

- int [MPTK_DeltaTicksPerQuarterNote](#)
From Midi Header: Delta Ticks Per Quarter Note. Represent the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.
- TimeSpan [MPTK_Duration](#)
Duration of the midi. This duration is not constant depending of midi event change tempo inside the midi file.
- bool [MPTK_EnableChangeTempo](#)
Should accept change tempo from Midi Events ?
- double [MPTK_InitialTempo](#)
Initial tempo found in the Midi
- bool [MPTK_KeepNoteOff](#)
Should keep note off event Events ?
- bool [MPTK_LogEvents](#)
Log midi events
- int [MPTK_MicrosecondsPerQuarterNote](#)
From the SetTempo event: The tempo is given in micro seconds per quarter beat. To convert this to BPM we needs to use the following equation: $BPM = 60,000,000/[tt \ tt]$ Warning: this value can change during the playing when a change tempo event is find. <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_No32ndNotesInQuarterNote](#)
From TimeSignature event: This value specifies the number of 1/32nds of a note happen every MIDI quarter note. It is usually 8 which means that a quarter note happens every quarter note. <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_NumberBeatsMeasure](#)
From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_NumberQuarterBeat](#)
From TimeSignature event: number of quarter notes in a beat. Equal 2 Power TimeSigDenominator. <https://paxstellar.fr/2020/09/11/midi-timing/>

- TimeSpan [MPTK_RealDuration](#)
Real Duration of the midi calculated with the midi change tempo events find inside the midi file.
- long [MPTK_TickLast](#)
Last tick position in Midi: Time of the last midi event in sequence expressed in number of "ticks". $MPTK_TickLast / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.
- int [MPTK_TicksInMetronomeClick](#)
From TimeSignature event: The standard MIDI clock ticks every 24 times every quarter note (crotchet) so a [cc] value of 24 would mean that the metronome clicks once every quarter note. A [cc] value of 6 would mean that the metronome clicks once every 1/8th of a note (quaver). <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_TimeSigDenominator](#)
From TimeSignature event: The denominator specifies the number of quarter notes in a beat. 2 represents a quarter-note, 3 represents an eighth-note, etc. . <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_TimeSigNumerator](#)
From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. In MIDI the denominator value is stored in a special format. i.e. the real denominator = $2^{[dd]}$ <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_TrackCount](#)
Count of track read in the Midi file

Properties

- int [MPTK_MidiIndex](#) [get, set]
Index Midi. Find the Index of Midi file from the popup in [MidiFileLoader](#) inspector. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the resource folder and open Midi File Setup to automatically integrate Midi in MPTK. return -1 if not found
- string [MPTK_MidiName](#) [get, set]
Midi name to load. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the resource folder and open Midi File Setup to automatically integrate Midi in MPTK.

Detailed Description

Script associated to the prefab [MidiFileLoader](#). No sequencer, no synthetizer, no music playing capabilities. Usefull to load all or part of the Midi events from a Midi and process, transform, write them to what you want. List of Midi file must be defined with Midi Player Setup (see Unity menu MPTK).

```
// Example of script. See TestMidiFileLoad.cs for a more detailed usage.
// Need of a reference to the Prefab (to be set in the hierarchy)
MidiFileLoader MidiLoader;
```

```

if (MidiLoader==null)
    Debug.LogError("TestMidiFileLoad: there is no MidiFileLoader Prefab set in
Inspector.");

// Defined index (from the Midi list defined in MPTK)
MidiLoader.MPTK_MidiIndex = midiindex;

// Load Midi event from the Midi file
MidiLoader.MPTK_Load();

// Get the list of events from start to end (in ticks)
List<MPTKEvent> events = MidiLoader.MPTK_ReadMidiEvents(StartTicks, EndTicks);
!

```

Member Function Documentation

void MPTK_Load (byte[] *midiBytesToLoad* = null)

Load the midi file defined with MPTK_MidiName or MPTK_MidiIndex or from a array of bytes

Parameters

<i>midiBytesToLoad</i>	
------------------------	--

void MPTK_Next ()

Read next Midi from the list of midi defined in MPTK (see Unity menu Midi)

MPTKEvent.EnumLength MPTK_NoteLength (MPTKEvent *note*)

Return note length as https://en.wikipedia.org/wiki/Note_value

Parameters

<i>note</i>	
-------------	--

Returns

[MPTKEvent.EnumLength](#)

void MPTK_Previous ()

Read previous Midi from the list of midi defined in MPTK (see Unity menu Midi)

List<MPTKEvent> MPTK_ReadMidiEvents (long *fromTicks* = 0, long *toTicks* = long.MaxValue)

Read the list of midi events available in the Midi from a ticks position to an end position.

Parameters

<i>fromTicks</i>	ticks start
------------------	-------------

<i>toTicks</i>	ticks end
----------------	-----------

Returns

Member Data Documentation

int MPTK_DeltaTicksPerQuarterNote

From Midi Header: Delta Ticks Per Quarter Note. Represent the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.

TimeSpan MPTK_Duration

Duration of the midi. This duration is not constant depending of midi event change tempo inside the midi file.

bool MPTK_EnableChangeTempo

Should accept change tempo from Midi Events ?

double MPTK_InitialTempo

Initial tempo found in the Midi

bool MPTK_KeepNoteOff

Should keep note off event Events ?

bool MPTK_LogEvents

Log midi events

int MPTK_MicrosecondsPerQuarterNote

From the SetTempo event: The tempo is given in micro seconds per quarter beat. To convert this to BPM we needs to use the following equation: $BPM = 60,000,000 / [tt \ tt]$ Warning: this value can change during the playing when a change tempo event is find. <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_No32ndNotesInQuarterNote

From TimeSignature event: This value specifies the number of 1/32nds of a note happen every MIDI quarter note. It is usually 8 which means that a quarter note happens every quarter note. <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_NumberBeatsMeasure

From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_NumberQuarterBeat

From TimeSignature event: number of quarter notes in a beat. Equal 2 Power TimeSigDenominator. <https://paxstellar.fr/2020/09/11/midi-timing/>

TimeSpan MPTK_RealDuration

Real Duration of the midi calculated with the midi change tempo events find inside the midi file.

long MPTK_TickLast

Last tick position in Midi: Time of the last midi event in sequence expressed in number of "ticks". $MPTK_TickLast / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.

int MPTK_TicksInMetronomeClick

From TimeSignature event: The standard MIDI clock ticks every 24 times every quarter note (crotchet) so a [cc] value of 24 would mean that the metronome clicks once every quarter note. A [cc] value of 6 would mean that the metronome clicks once every 1/8th of a note (quaver). <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_TimeSigDenominator

From TimeSignature event: The denominator specifies the number of quarter notes in a beat. 2 represents a quarter-note, 3 represents an eighth-note, etc. . <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_TimeSigNumerator

From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. In MIDI the denominator value is stored in a special format. i.e. the real denominator = $2^{[dd]}$ <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_TrackCount

Count of track read in the Midi file

Property Documentation

int MPTK_MidiIndex [get], [set]

Index Midi. Find the Index of Midi file from the popup in [MidiFileLoader](#) inspector. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the resource folder and open Midi File Setup to automatically integrate Midi in MPTK. return -1 if not found

```
midiFileLoader.MPTK MidiIndex = 1;  
!
```

Parameters

<i>index</i>	
--------------	--

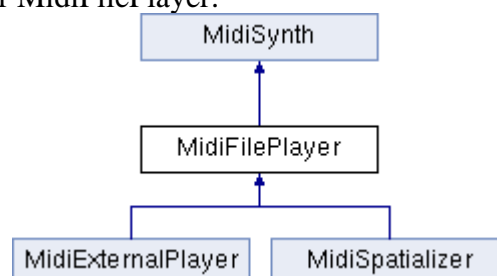
string MPTK_MidiName [get], [set]

Midi name to load. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the resource folder and open Midi File Setup to automatically integrate Midi in MPTK.

MidiFilePlayer

Script associated to the prefab [MidiFilePlayer](#). Simply, play a Midi file. Midi files must be defined from the Unity menu MPTK in the Unity editor. There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.

Inheritance diagram for MidiFilePlayer:



Public Member Functions

- [MidiLoad](#) [MPTK_Load](#) ()
Load the midi file defined with MPTK_MidiName or MPTK_MidiIndex. It's an optional action before playing a midi file with [MPTK_Play\(\)](#)
Use this method to get all Midi events before start playing.
- void [MPTK_Next](#) ()
Play next Midi from the list of midi defined in MPTK (see Unity menu Midi)
- [MPTKEvent.EnumLength](#) [MPTK_NoteLength](#) ([MPTKEvent](#) note)
Return note length as https://en.wikipedia.org/wiki/Note_value
- void [MPTK_Pause](#) (float timeToPauseMS=-1f)
Pause the current playing
- virtual void [MPTK_Play](#) ()
Play the midi file defined with MPTK_MidiName or MPTK_MidiIndex
- virtual void [MPTK_Play](#) (float delayMillisecond)
[MPTK PRO] Play the midi file defined with MPTK_MidiName or MPTK_MidiIndex with ramp-up to the volume defined with MPTK_Volume (0.5 by default)
- void [MPTK_PlayNextOrPrevious](#) (int offset)
[MPTK PRO] - Play next or previous Midi from the MidiDB list.
- void [MPTK_Previous](#) ()
Play previous Midi from the list of midi defined in MPTK (see Unity menu Midi)
- void [MPTK_RePlay](#) ()
Restart playing of the current midi file
- void [MPTK_SearchMidiToPlay](#) (string name)
[MPTK PRO] - Find a Midi in the Unity resources folder MidiDB which contains the name (case sensitive) Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.
- void [MPTK_Stop](#) ()
Stop playing
- virtual void [MPTK_Stop](#) (float delayMillisecond)
[MPTK PRO] Stop playing within a delay. The volume decrease until the playing is stopped
- void [MPTK_SwitchMidiWithDelay](#) (int index, string name, float volume, float delayToStopMillisecond, float delayToStartMillisecond)
Switch playing between two Midis with ramp-up. This method is usefull for an integration with Bolt: main Midi parameters are defined in one call.
- void [MPTK_UnPause](#) ()

UnPause the current playing

Public Attributes

- bool [MPTK_PauseOnFocusLoss](#)
Should the Midi playing must be paused when the application lost the focus?
- bool [MPTK_StartPlayAtFirstNote](#)
Should the midi start playing at the first note found ?
- LoadingStatusMidiEnum [MPTK_StatusLastMidiLoaded](#)
Status of the last midi loaded. The status is updated in a coroutine, so the status can change at each frame.
- EventEndMidiClass [OnEventEndPlayMidi](#)
Define unity event to trigger at end of playing the midi.
- EventNotesMidiClass [OnEventNotesMidi](#)
Define unity event to trigger when notes available from the Midi file.
- EventStartMidiClass [OnEventStartPlayMidi](#)
Define unity event to trigger at start of playing the Midi.

Properties

- int [MPTK_DeltaTicksPerQuarterNote](#) [get]
Delta Ticks Per Quarter Note. Indicate the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.
- TimeSpan [MPTK_Duration](#) [get]
Duration (TimeSpan) of the midi.
- float [MPTK_DurationMS](#) [get]
Duration (milliseconds) of the midi.
- bool [MPTK_IsPaused](#) [get]
Is Midi file playing is paused ?
- bool [MPTK_IsPlaying](#) [get]
Is Midi file is playing ?
- bool [MPTK_KeepNoteOff](#) [get, set]
Should keep note off event Events from the Midi file ?
- bool [MPTK_LogEvents](#) [get, set]
Log midi events

- bool [MPTK_Loop](#) [get, set]
Should automatically restart playing when Midi reaches the end ? The midi doesn't need to be reload.
- List<TrackMidiEvent >? [MPTK_MidiEvents](#) [get]
[DEPRECATED] Get all the raw midi events available in the midi file. Use rather the class [MidiLoad](#).
- int [MPTK_MidiIndex](#) [get, set]
*Index Midi. Find the Index of Midi file (same values ad from the popup in [MidiFilePlayer](#) inspector).
Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK. return -1 if not found*
- [MidiLoad](#) [MPTK_MidiLoaded](#) [get]
*Get detailed information about the midi playing. This readonly properties is available only when a Midi is playing.
Rather use the method [MPTK_Load\(\)](#) to get information about a Midi before playing. V2.82.*
- virtual string [MPTK_MidiName](#) [get, set]
Midi name to play. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.
- bool [MPTK_PlayOnStart](#) [get, set]
Should the Midi start playing when application starts ?
- TimeSpan [MPTK_PlayTime](#) [get]
Time from the start of playing the current midi
- double? [MPTK_Position](#) [get, set]
Set or Get midi position of midi playing (in millisecond). If the Midi contains change of tempo, the position could not reflect the real time since the beginning. Use [MPTK_TickCurrent](#) to change the position in tick which is independent of the tempo and the speed.
- double [MPTK_PulseLenght](#) [get]
Lenght in millisecond of a quarter
- int [MPTK_Quantization](#) [get, set]
Level of quantization :
- float [MPTK_Speed](#) [get, set]
*Speed of playing. Between 0.1 (10%) to 10 (1000%).
Set to 1 for normal speed.*
- double [MPTK_Tempo](#) [get]
Get the current tempo from the Midi file (independent from [MPTK_Speed](#)). Return QuarterPerMinuteValue similar to BPM (Beat Per Measure)

- long? [MPTK_TickCurrent](#) [get, set]
Set or get the current tick position in Midi which is independent of the tempo and the speed. Use [MPTK_Position](#) to change the position in milliseconds.
- long? [MPTK_TickLast](#) [get]
Last tick position in Midi: Value of the tick for the last midi event in sequence expressed in number of "ticks". $MPTK_TickLast / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.

Additional Inherited Members

Detailed Description

Script associated to the prefab [MidiFilePlayer](#). Simply, play a Midi file. Midi files must be defined from the Unity menu MPTK in the Unity editor. There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.

```
// Example of script. See TestMidiFilePlayerScripting.cs for a more detailed usage.
// Need of a reference to the Prefab (to be set in the hierarchy or from the script)
MidiFilePlayer midiFilePlayer;

if (midiExternalPlayer==null)
    midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
if (midiExternalPlayer==null)
    Debug.LogError("TestMidiExternalPlayer: there is no MidiFilePlayer Prefab set in
Inspector.");

// Random select for the Midi
int index = UnityEngine.Random.Range(0, MidiPlayerGlobal.MPTK_ListMidi.Count);
midiFilePlayer.MPTK_MidiIndex = index;

// Play!
midiFilePlayer.MPTK_Play();

!
```

Member Function Documentation

[MidiLoad](#) [MPTK_Load \(\)](#)

Load the midi file defined with [MPTK_MidiName](#) or [MPTK_MidiIndex](#). It's an optional action before playing a midi file with [MPTK_Play\(\)](#)

Use this method to get all Midi events before start playing.

```
private void GetMidiInfo()
{
    MidiLoad midiloading = midiFilePlayer.MPTK_Load();
    if (midiloading != null)
    {
        infoMidi = "Duration: " + midiloading.MPTK_Duration.TotalSeconds + "
seconds\n";
        infoMidi += "Tempo: " + midiloading.MPTK_InitialTempo + "\n";
        List<MPTKEvent> listEvents = midiloading.MPTK_ReadMidiEvents();
        infoMidi += "Count Midi Events: " + listEvents.Count + "\n";
        Debug.Log(infoMidi);
    }
}
```

```
}  
!
```

Returns

[MidiLoad](#) to access all the properties of the midi loaded

void MPTK_Next ()

Play next Midi from the list of midi defined in MPTK (see Unity menu Midi)

[MPTKEvent.EnumLength](#) MPTK_NoteLength ([MPTKEvent](#) *note*)

Return note length as https://en.wikipedia.org/wiki/Note_value

Parameters

<i>note</i>	
-------------	--

Returns

[MPTKEvent.EnumLength](#)

void MPTK_Pause (float *timeToPauseMS* = -1f)

Pause the current playing

Parameters

<i>timeToPauseMS</i>	time to pause in milliseconds. default or < 0 : indefinitely
----------------------	--

virtual void MPTK_Play () [virtual]

Play the midi file defined with MPTK_MidiName or MPTK_MidiIndex

Reimplemented in [MidiExternalPlayer](#).

virtual void MPTK_Play (float *delayMillisecond*) [virtual]

[MPTK PRO] Play the midi file defined with MPTK_MidiName or MPTK_MidiIndex with ramp-up to the volume defined with MPTK_Volume (0.5 by default)

Parameters

<i>delayMillisecond</i>	ramp-up delay in milliseconds to get the default volume
-------------------------	---

void MPTK_PlayNextOrPrevious (int *offset*)

[MPTK PRO] - Play next or previous Midi from the MidiDB list.

Parameters

<i>offset</i>	Forward or backward count in the list. 1:the next, -1:the previous
---------------	--

void MPTK_Previous ()

Play previous Midi from the list of midi defined in MPTK (see Unity menu Midi)

void MPTK_RePlay ()

Restart playing of the current midi file

void MPTK_SearchMidiToPlay (string *name*)

[MPTK PRO] - Find a Midi in the Unity resources folder MidiDB which contains the name (case sensitive) Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the resource folder and open Midi File Setup to automatically integrate Midi in MPTK.

```
midiFilePlayer.MPTK_SearchMidiToPlay("Adagio");  
midiFilePlayer.MPTK_Play();  
!
```

void MPTK_Stop ()

Stop playing

virtual void MPTK_Stop (float *delayMillisecond*) [virtual]

[MPTK PRO] Stop playing within a delay. The volume decrease until the playing is stopped

Parameters

<i>delayMillisecond</i>	decrease time in milliseconds
-------------------------	-------------------------------

void MPTK_SwitchMidiWithDelay (int *index*, string *name*, float *volume*, float *delayToStopMillisecond*, float *delayToStartMillisecond*)

Switch playing between two Midis with ramp-up. This method is usefull for an integration with Bolt: main Midi parameters are defined in one call.

Parameters

<i>index</i>	Index of the Midi to play. Index is used only if no name is defined.
<i>name</i>	Name of the Midi to play. Can be part of the Midi Name. If set, this parameter has priority over index parameter.
<i>volume</i>	Volume of the Midi. -1 to not change the default volume

<i>delayToStopMillis econd</i>	Delay to stop the current midi playing (with volume decrease) or delay before playing the Midi if not Misi is playing
<i>delayToStartMillis econd</i>	Delay to get the Midi at full volume (ramp-up volume)

void MPTK_UnPause ()

UnPause the current playing

Member Data Documentation

bool MPTK_PauseOnFocusLoss

Should the Midi playing must be paused when the application lost the focus?

bool MPTK_StartPlayAtFirstNote

Should the midi start playing at the first note found ?

LoadingStatusMidiEnum MPTK_StatusLastMidiLoaded

Status of the last midi loaded. The status is updated in a coroutine, so the status can change at each frame.

EventEndMidiClass OnEventEndPlayMidi

Define unity event to trigger at end of playing the midi.

```

MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
...
if (!midiFilePlayer.OnEventEndPlayMidi.HasEvent())
{
    // No listener defined, set now by script. EndPlay will be called.
    midiFilePlayer.OnEventEndPlayMidi.AddListener(EndPlay);
}
...
public void EndPlay(string midiname, EventEndMidiEnum reason)
{
    Debug.LogFormat("End playing midi {0} reason:{1}", midiname, reason);
}
!

```

EventNotesMidiClass OnEventNotesMidi

Define unity event to trigger when notes available from the Midi file.

```

MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
    ...
if (!midiFilePlayer.OnEventNotesMidi.HasEvent())
{
    // No listener defined, set now by script. NotesToPlay will be called for each
    new notes read from Midi file
    midiFilePlayer.OnEventNotesMidi.AddListener(NotesToPlay);
}
    ...
public void NotesToPlay(List<MPTKEvent> notes)
{
    Debug.Log(notes.Count);
    foreach (MPTKEvent midievent in notes)
    {
        ...
    }
}
!

```

EventStartMidiClass OnEventStartPlayMidi

Define unity event to trigger at start of playing the Midi.

```

! MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
    ...
if (!midiFilePlayer.OnEventStartPlayMidi.HasEvent())
{
    // No listener defined, set now by script. StartPlay will be called.
    midiFilePlayer.OnEventStartPlayMidi.AddListener(StartPlay);
}
    ...
public void StartPlay(string midiname)
{
    Debug.LogFormat("Start playing midi {0}", midiname);
}
!

```

Property Documentation

int MPTK_DeltaTicksPerQuarterNote [get]

Delta Ticks Per Quarter Note. Indicate the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.

TimeSpan MPTK_Duration [get]

Duration (TimeSpan) of the midi.

float MPTK_DurationMS [get]

Duration (milliseconds) of the midi.

bool MPTK_IsPaused [get]

Is Midi file playing is paused ?

bool MPTK_IsPlaying [get]

Is Midi file is playing ?

bool MPTK_KeepNoteOff [get], [set]

Should keep note off event Events from the Midi file ?

bool MPTK_LogEvents [get], [set]

Log midi events

bool MPTK_Loop [get], [set]

Should automatically restart playing when Midi reaches the end ? The midi doesn't need to be reload.

List<TrackMidiEvent>? MPTK_MidiEvents [get]

[DEPRECATED] Get all the raw midi events available in the midi file. Use rather the class [MidiLoad](#).

```
MidiLoad MidiLoaded = new MidiLoad();
MidiLoaded.MPTK_Load(midiindex);
List<MPTKEvent> events = MidiLoaded.MPTK_ReadMidiEvents();
!
```

int MPTK_MidiIndex [get], [set]

Index Midi. Find the Index of Midi file (same values ad from the popup in [MidiFilePlayer](#) inspector).
Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK. return -1 if not found

```
midiFilePlayer.MPTK_MidiIndex = 33;
midiFilePlayer.MPTK_Play();
!
```

///

Parameters

<i>index</i>	
--------------	--

MidiLoad MPTK_MidiLoaded [get]

Get detailed information about the midi playing. This readonly properties is available only when a Midi is playing.

Rather use the method [MPTK_Load\(\)](#) to get information about a Midi before playing. V2.82.

virtual string MPTK_MidiName [get], [set]

Midi name to play. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.

```
midiFilePlayer.MPTK_MidiName = "Albinoni - Adagio";
midiFilePlayer.MPTK_Play();
!
```

bool MPTK_PlayOnStart [get], [set]

Should the Midi start playing when application starts ?

TimeSpan MPTK_PlayTime [get]

Time from the start of playing the current midi

double? MPTK_Position [get], [set]

Set or Get midi position of midi playing (in millisecond). If the Midi contains change of tempo, the position could not reflect the real time since the beginning. Use [MPTK_TickCurrent](#) to change the position in tick which is independent of the tempo and the speed.

```
double currentPosition = Math.Round(midiFilePlayer.MPTK_Position / 1000d, 2);
double newPosition =
Math.Round(GUILayout.HorizontalSlider((float)currentPosition, 0f,
(float)midiFilePlayer.MPTK_Duration.TotalSeconds, GUILayout.Width(buttonWidth)),
2);
if (newPosition != currentPosition)
{
    Debug.Log("New position " + currentPosition + "--> " + newPosition);
    midiFilePlayer.MPTK_Position = newPosition * 1000d;
}
!
```

double MPTK_PulseLenght [get]

Lenght in millisecond of a quarter

int MPTK_Quantization [get], [set]

Level of quantization :

- 0 = None
- 1 = Quarter Note
- 2 = Eighth Note
- 3 = 16th Note
- 4 = 32th Note
- 5 = 64th Note

float MPTK_Speed [get], [set]

Speed of playing. Between 0.1 (10%) to 10 (1000%).

Set to 1 for normal speed.

double MPTK_Tempo [get]

Get the current tempo from the Midi file (independent from MPTK_Speed). Return QuarterPerMinuteValue similar to BPM (Beat Per Measure)

long? MPTK_TickCurrent [get], [set]

Set or get the current tick position in Midi which is independent of the tempo and the speed. Use MPTK_Position to change the position in milliseconds.

long? MPTK_TickLast [get]

Last tick position in Midi: Value of the tick for the last midi event in sequence expressed in number of "ticks". $MPTK_TickLast / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.

MidiFileWriter2

[MPTK PRO] - Write a midi file from differents sources based on NAudio framework. See full example TestMidiWriter.cs with a light sequencer. This class replaced MidiFileWriter with these changes: channel start at 0, new specfic event, better control. More information here:

<https://paxstellar.fr/class-midifilewriter2/>

Public Member Functions

- [MidiFileWriter2](#) ()

Create an empty [MidiFileWriter2](#) type=1 and Delta Ticks Per Quarter Note=120

- [MidiFileWriter2](#) (int deltaTicksPerQuarterNote, int midiFileType)
Create a [MidiFileWriter2](#) with an empty Midi Event list
- void [MPTK_AddBPMChange](#) (int track, long absoluteTime, int bpm)
Creates a tempo change. No track nor channel as tempo change applied to the whole midi
- void [MPTK_AddChangePreset](#) (int track, long absoluteTime, int channel, int preset)
Add a change preset
- void [MPTK_AddChannelAfterTouch](#) (int track, long absoluteTime, int channel, int afterTouchPressure)
Add a Channel After-Touch Event
- void [MPTK_AddControlChange](#) (int track, long absoluteTime, int channel, [MPTKController](#) controller, int controllerValue)
Creates a control change event
- void [MPTK_AddNote](#) (int track, long absoluteTime, int channel, int note, int velocity, int duration)
Add a note event. the corresponding Noteoff is automatically created.
- void [MPTK_AddPitchWheelChange](#) (int track, long absoluteTime, int channel, float pitchWheel)
Creates a control change event
- void [MPTK_AddTempoChange](#) (int track, long absoluteTime, int microsecondsPerQuarterNote)
Creates a tempo change. No track nor channel as tempo change applied to the whole midi.
- void [MPTK_AddText](#) (int track, long absoluteTime, [MPTKMeta](#) typeMeta, string text)
Create a new TimeSignatureEvent. This event is optionnal. Midi sequencer assumes the default value is 4,4,24,32. No track nor channel as tempo change applied to the whole midi.
- void [MPTK_AddTimeSignature](#) (int track, long absoluteTime, int numerator, int denominator, int ticksInMetronomeClick=24, int no32ndNotesInQuarterNote=32)
Create a new TimeSignatureEvent. This event is optionnal. Midi sequencer assumes the default value is 4,4,24,32. No track nor channel as tempo change applied to the whole midi.
- void [MPTK_CreateTrack](#) (int count)
Create tracks
- void [MPTK_EndTrack](#) (int trackNumber)
Close the track (mandatory for a well formed midi file)
- bool [MPTK_LoadFromFile](#) (string filename)
Load a Midi file from OS system file (could be dependant of the OS)
- bool [MPTK_LoadFromMidiDB](#) (int indexMidiDb)

Create a [MidiFileWriter2](#) from a Midi found in MPTK MidiDB

- bool [MPTK_LoadFromMPTK](#) (List< TrackMidiEvent > MidiSorted)
Create a [MidiFileWriter2](#) from a MPTK list of midi events. A midi file must be loaded before from a [MidiFilePlayer](#) gameobject (as in example) or from a call to [MidiFileWriter2.MPTK_LoadFromFile\(filename\)](#).
- bool [MPTK_WriteToFile](#) (string filename)
Write Midi file to an OS folder
- bool [MPTK_WriteToMidiDB](#) (string filename)
Write Midi file to MidiDB. To be used only in edit mode not in a standalone application.

Static Public Member Functions

- static int [MPTK_GetMicrosecondsPerQuarterNote](#) (int bpm)
Convert BPM to duration or a quarter in microsecond

Properties

- int? [MPTK_DeltaTicksPerQuarterNote](#) [get]
Get the DeltaTicksPerQuarterNote of the loaded midi
- int? [MPTK_MidiFileType](#) [get]
Get the midi file type of the loaded midi (0,1,2)
- int? [MPTK_TrackCount](#) [get]
Get the track count of the loaded midi

Detailed Description

[MPTK PRO] - Write a midi file from differents sources based on NAudio framework. See full example TestMidiWriter.cs with a light sequencer. This class replaced MidiFileWriter with these changes: channel start at 0, new specific event, better control. More information here: <https://paxstellar.fr/class-midifilewriter2/>

Constructor & Destructor Documentation

[MidiFileWriter2](#) ()

Create an empty [MidiFileWriter2](#) type=1 and Delta Ticks Per Quarter Note=120

MidiFileWriter2 (int *deltaTicksPerQuarterNote*, int *midiFileType*)

Create a [MidiFileWriter2](#) with an empty Midi Event list

Parameters

<i>deltaTicksPerQuarterNote</i>	Delta Ticks Per Quarter Note
<i>midiFileType</i>	type of Midi format. Must be 0 or 1 (better)

Member Function Documentation

void MPTK_AddBPMChange (int *track*, long *absoluteTime*, int *bpm*)

Creates a tempo change. No track nor channel as tempo change applied to the whole midi

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Tick time for this event
<i>bpm</i>	quarter per minute

void MPTK_AddChangePreset (int *track*, long *absoluteTime*, int *channel*, int *preset*)

Add a change preset

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Tick time for this event
<i>channel</i>	Channel must be in the range 0-15
<i>preset</i>	Preset (program/patch) must be in the range 0-127

void MPTK_AddChannelAfterTouch (int *track*, long *absoluteTime*, int *channel*, int *afterTouchPressure*)

Add a Channel After-Touch Event

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Tick time for this event
<i>channel</i>	Channel must be in the range 0-15
<i>afterTouchPressure</i>	After-touch pressure from 0 to 127

void MPTK_AddControlChange (int *track*, long *absoluteTime*, int *channel*, [MPTKController](#) *controller*, int *controllerValue*)

Creates a control change event

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Tick time for this event
<i>channel</i>	Channel must be in the range 0-15
<i>controller</i>	The MIDI Controller
<i>controllerValue</i>	Controller value

void MPTK_AddNote (int *track*, long *absoluteTime*, int *channel*, int *note*, int *velocity*, int *duration*)

Add a note event. the corresponding Noteoff is automatically created.

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Tick time for this event
<i>channel</i>	Channel must be in the range 0-15
<i>note</i>	Note must be in the range 0-127
<i>velocity</i>	Velocity must be in the range 0-127.
<i>duration</i>	Tick duration

void MPTK_AddPitchWheelChange (int *track*, long *absoluteTime*, int *channel*, float *pitchWheel*)

Creates a control change event

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Tick time for this event
<i>channel</i>	Channel must be in the range 0-15
<i>pitchWheel</i>	Pitch Wheel Value. 1:normal value, 0:pitch mini, 2:pitch max

void MPTK_AddTempoChange (int *track*, long *absoluteTime*, int *microsecondsPerQuarterNote*)

Creates a tempo change. No track nor channel as tempo change applied to the whole midi.

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Tick time for this event
<i>microsecondsPerQuarterNote</i>	Microseconds per quarter note

void MPTK_AddText (int *track*, long *absoluteTime*, [MPTKMeta](#) *typeMeta*, string *text*)

Create a new TimeSignatureEvent. This event is optional. Midi sequencer assumes the default value is 4,4,24,32. No track nor channel as tempo change applied to the whole midi.

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Absolute time of this event

<i>typeMeta</i> MetaEvent type (must be one that is MetaEvent type (must be one that is Parameters <i>text</i>	The text in this type
associated with text data)	

void MPTK_AddTimeSignature (int *track*, long *absoluteTime*, int *numerator*, int *denominator*, int *ticksInMetronomeClick* = 24, int *no32ndNotesInQuarterNote* = 32)

Create a new TimeSignatureEvent. This event is optional. Midi sequencer assumes the default value is 4,4,24,32. No track nor channel as tempo change applied to the whole midi.

Parameters

<i>track</i>	Track for this event
<i>absoluteTime</i>	Time at which to create this event
<i>numerator</i>	Numerator
<i>denominator</i>	Denominator
<i>ticksInMetronomeClick</i>	Ticks in Metronome Click. Set to 24 for a standard value.
<i>no32ndNotesInQuarterNote</i>	No of 32nd Notes in Quarter Click. Set to 32 for a standard value.

void MPTK_CreateTrack (int *count*)

Create tracks

Parameters

<i>count</i>	number of tracks to create
--------------	----------------------------

void MPTK_EndTrack (int *trackNumber*)

Close the track (mandatory for a well formed midi file)

Parameters

<i>trackNumber</i>	Track number to close
--------------------	-----------------------

static int MPTK_GetMicrosecondsPerQuarterNote (int *bpm*)[static]

Convert BPM to duration of a quarter in microsecond

Parameters

<i>bpm</i>	m
------------	---

Returns

bool MPTK_LoadFromFile (string *filename*)

Load a Midi file from OS system file (could be dependant of the OS)

Parameters

<i>filename</i>	
-----------------	--

Returns

bool MPTK_LoadFromMidiDB (int *indexMidiDb*)

Create a [MidiFileWriter2](#) from a Midi found in MPTK MidiDB

Parameters

<i>indexMidiDb</i>	
--------------------	--

bool MPTK_LoadFromMPTK (List< TrackMidiEvent > *MidiSorted*)

Create a [MidiFileWriter2](#) from a MPTK list of midi events. A midi file must be loaded before from a [MidiFilePlayer](#) gameobject (as in example) or from a call to `MidiFileWriter2.MPTK_LoadFromFile(filename)`.

Parameters

<i>MidiSorted</i>	
-------------------	--

bool MPTK_WriteToFile (string *filename*)

Write Midi file to an OS folder

Parameters

<i>filename</i>	filename of the midi file
-----------------	---------------------------

Returns

bool MPTK_WriteToMidiDB (string *filename*)

Write Midi file to MidiDB. To be used only in edit mode not in a standalone application.

Parameters

<i>filename</i>	filename of the midi file without any folder and any extension
-----------------	--

Returns

Property Documentation

int? MPTK_DeltaTicksPerQuarterNote [get]

Get the DeltaTicksPerQuarterNote of the loaded midi

int? MPTK_MidiFileType [get]

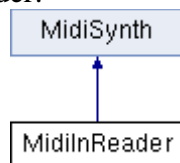
Get the midi file type of the loaded midi (0,1,2)

int? MPTK_TrackCount [get]

Get the track count of the loaded midi

MidiInReader

[MPTK PRO] - Script associated to the prefab [MidiInReader](#). Read Midi events from a Midi keyboard connected your device (Windows 10 or MacOS). See example of use in TestMidiInputScripting.cs
There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.
Inheritance diagram for MidiInReader:



Public Attributes

- bool [MPTK_LogEvents](#)
Log midi events
- bool [MPTK_ReadMidiInput](#)
Read Midi input
- EventMidiClass [OnEventInputMidi](#)
Define unity event to trigger when note available from the Midi file.

Additional Inherited Members

Detailed Description

[MPTK PRO] - Script associated to the prefab [MidiInReader](#). Read Midi events from a Midi keyboard connected your device (Windows 10 or MacOS). See example of use in TestMidiInputScripting.cs There is no need to writing a script. For a simple usage, all the job can be done in the prefab inspector.

```
// Example of script. See TestMidiInputScripting.cs for a more detailed usage.
// Need for a reference to the Prefab (can also be set from the hierarchy)
MidiInReader midiIn = FindObjectOfType<MidiInReader>();

if (midiIn == null)
    Debug.Log("Can't find a MidiInReader Prefab in the Hierarchy. No events will be read");

// There is two methods to trigger event: in inspector from the Unity editor or by script
midiIn.OnEventInputMidi.AddListener((MPTKEvent evt) =>
{
    // your processing here
    Debug.Log(evt.ToString());
});
!
```

Member Data Documentation

bool MPTK_LogEvents

Log midi events

bool MPTK_ReadMidiInput

Read Midi input

EventMidiClass OnEventInputMidi

Define unity event to trigger when note available from the Midi file.

```
MidiInReader midiFilePlayer = FindObjectOfType<MidiInReader>();
...
if (!midiFilePlayer.OnEventInputMidi.HasEvent())
{
    // No listener defined, set now by script. NotesToPlay will be called for each
    new notes read from Midi file
    midiFilePlayer.OnEventInputMidi.AddListener(NotesToPlay);
}
...
public void NotesToPlay(MPTKEvent notes)
{
    Debug.Log(notes.Value);
    foreach (MPTKEvent midievent in notes)
    {
        ...
    }
}
!
```

MidiKeyboard

Send and receive Midi Message from a Midi keyboard connected to the device
<https://paxstellar.fr/class-midikeyboard/>

Public Types

- enum [PluginError](#)
General error return values

Public Member Functions

- static void [MPTK_CloseAllInp](#) ()
Close all input device for receiving Midi message
- static void [MPTK_CloseOut](#) (int index)
Close device for output
- static int [MPTK_CountInp](#) ()
Count of output device detected
- static int [MPTK_CountOut](#) ()
Count of input device detected
- static void [MPTK_OpenAllInp](#) ()
Open or refresh all input device for receiving Midi message
- static void [MPTK_OpenOut](#) (int index)
Open device for output

Static Public Member Functions

- static void [MPTK_ExcludeSystemMessage](#) (bool exclude)
Exclude system message
- static string [MPTK_GetInpName](#) (int index)
Name of the device
- static string [MPTK_GetOutName](#) (int index)
Name of the device
- static void [MPTK_Init](#) ()
Plugins Init
- static void [MPTK_PlayEvent](#) ([MPTKEvent](#) evnt, int device)

Play one midi event with a thread so the call return immediately.

- static [MPTKEvent MPTK_Read](#) ()
Read a Midi message from all devices input connected
- static string [MPTK_Version](#) ()
Get current version of the plugins

Properties

- static [PluginError MPTK_LastStatus](#) [get]
Last status, value resetted after the call

Detailed Description

Send and receive Midi Message from a Midi keyboard connected to the device <https://paxstellar.fr/class-midikeyboard/>

Member Enumeration Documentation

enum [PluginError](#) [strong]

General error return values

Member Function Documentation

static void MPTK_CloseAllInp ()

Close all input device for receiving Midi message

static void MPTK_CloseOut (int *index*)

Close device for output

Parameters

<i>index</i>	
--------------	--

static int MPTK_CountInp ()

Count of output device detected

Returns

static int MPTK_CountOut ()

Count of input device detected

Returns

static void MPTK_ExcludeSystemMessage (bool *exclude*) [static]

Exclude system message

Parameters

<i>exclude</i>	If true exclude all messages with status/command >= 0xF0. Default: true
----------------	---

static string MPTK_GetInpName (int *index*) [static]

Name of the device

Parameters

<i>index</i>	Id of the device
--------------	------------------

Returns

static string MPTK_GetOutName (int *index*) [static]

Name of the device

Parameters

<i>index</i>	Id of the device
--------------	------------------

Returns

static void MPTK_Init () [static]

Plugins Init

static void MPTK_OpenAllInp ()

Open or refresh all input device for receiving Midi message

static void MPTK_OpenOut (int *index*)

Open device for output

Parameters

<i>index</i>	
--------------	--

static void MPTK_PlayEvent ([MPTKEvent](#) *evnt*, int *device*)[static]

Play one midi event with a thread so the call return immediately.

Parameters

<i>evnt</i>	
<i>device</i>	

static [MPTKEvent](#) MPTK_Read () [static]

Read a Midi message from all devices input connected

Returns

static string MPTK_Version () [static]

Get current version of the plugins

Returns

Property Documentation

[PluginError](#) **MPTK_LastStatus** [static], [get]

Last status, value resetted after the call

MidiListPlayer

[MPTK PRO] - Script for the prefab [MidiListPlayer](#). Play a list of pre-selected midi file from the dedicated inspector. List of Midi files must exists in MidiDB. See Midi Player Setup (Unity menu MPTK).

Inherits MonoBehaviour.

Classes

- class [MPTK_MidiPlayItem](#)
Define a midi to be added in the list

Public Member Functions

- void [MPTK_AddMidi](#) (string name, float start=0, float end=0)
Add a Midi name to the list. Use the exact name defined in Unity resources (folder MidiDB) without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the resource folder and open Midi File Setup to automatically integrate Midi in MPTK.
- [MPTK_MidiPlayItem](#) [MPTK_GetAt](#) (int index)
Get description of a play item at position.
- void [MPTK_NewList](#) ()
Create an empty list
- void [MPTK_Next](#) ()
Play next Midi in list
- void [MPTK_Pause](#) ()
Pause the current playing
- void [MPTK_Play](#) ()
Play the midi in list at MPTK_PlayIndex position
- void [MPTK_Previous](#) ()
Play previous Midi in list
- void [MPTK_ReIndexMidi](#) ()
Recalculate the index of the midi from the list.
- void [MPTK_RemoveMidi](#) (string name)
Remove a Midi name from the list. Use the exact name defined in Unity resources folder MidiDB without any path or extension.
- void [MPTK_RemoveMidiAt](#) (int index)
Remove a Midi at position from the list..
- void [MPTK_RePlay](#) ()
Restart playing the current midi file
- void [MPTK_Stop](#) ()
Stop playing
- void [MPTK_UnPause](#) ()
Pause the current playing

Public Attributes

- MidiListPlayerStatus [MPTK_MidiFilePlayer_1](#)
First [MidiFilePlayer](#) to play the Midi
- MidiListPlayerStatus [MPTK_MidiFilePlayer_2](#)
Second [MidiFilePlayer](#) to play the Midi
- float [MPTK_OverlayTimeMS](#)
Duration of overlay between playing two midi
- List< [MPTK_MidiPlayItem](#) > [MPTK_PlayList](#)
Play list
- EventEndMidiClass [OnEventEndPlayMidi](#)
Define unity event to trigger at end
- EventStartMidiClass [OnEventStartPlayMidi](#)
Define unity event to trigger at start

Properties

- TimeSpan [MPTK_Duration](#) [get]
Duration of the midi. This duration can change during the playing when Change Tempo Event are processed.
- bool [MPTK_IsPaused](#) [get]
Is Midi file playing is paused ?
- bool [MPTK_IsPlaying](#) [get]
Is Midi file is playing ?
- bool [MPTK_Loop](#) [get, set]
Should automatically restart when Midi reach the end ?
- int? [MPTK_PlayIndex](#) [get, set]
Play a specific Midi in the list.
- bool [MPTK_PlayOnStart](#) [get, set]
Should the Midi start playing when application start ?
- double [MPTK_Position](#) [get, set]
Set or Get midi position time from 0 to lenght time of midi playing (in millisecond). No effect if the Midi is not playing.
- long [MPTK_TickCurrent](#) [get, set]

Current tick position in Midi: Time of the current midi event expressed in number of "ticks". $MPTK_TickCurrent / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.

- long [MPTK_TickLast](#) [get]
Last tick position in Midi: Value of the tick for the last midi event in sequence expressed in number of "ticks". $MPTK_TickLast / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.
- float [MPTK_Volume](#) [get, set]
Volume of midi playing. Must be ≥ 0 and ≤ 1

Detailed Description

[MPTK PRO] - Script for the prefab [MidiListPlayer](#). Play a list of pre-selected midi file from the dedicated inspector. List of Midi files must exists in MidiDB. See Midi Player Setup (Unity menu MPTK).

Member Function Documentation

void MPTK_AddMidi (string *name*, float *start* = 0, float *end* = 0)

Add a Midi name to the list. Use the exact name defined in Unity resources (folder MidiDB) without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.

```
midiListPlayer.MPTK_AddMidi("Albinoni - Adagio");  
midiListPlayer.MPTK_AddMidi("Conan The Barbarian", 10000, 20000);  
!
```

Parameters

<i>name</i>	midi filename as defined in resources
<i>start</i>	starting time of playing (ms). Default: start of the midi
<i>end</i>	ending time of playing (ms). Default: end of midi

[MPTK_MidiPlayItem](#) MPTK_GetAt (int *index*)

Get description of a play item at position.

```
// GET the second midi in list (start at 0)  
midiListPlayer.MPTK_GetAt(1);  
!
```

void MPTK_NewList ()

Create an empty list

void MPTK_Next ()

Play next Midi in list

void MPTK_Pause ()

Pause the current playing

void MPTK_Play ()

Play the midi in list at MPTK_PlayIndex position

void MPTK_Previous ()

Play previous Midi in list

void MPTK_ReIndexMidi ()

Recalculate the index of the midi from the list.

void MPTK_RemoveMidi (string *name*)

Remove a Midi name from the list. Use the exact name defined in Unity resources folder MidiDB without any path or extension.

```
midiListPlayer.MPTK_RemoveMidi("Albinoni - Adagio");  
!
```

void MPTK_RemoveMidiAt (int *index*)

Remove a Midi at position from the list..

```
// Removes the second midi in list (start at 0)  
midiListPlayer.MPTK_RemoveMidiAt(1);  
!
```

void MPTK_RePlay ()

Restart playing the current midi file

void MPTK_Stop ()

Stop playing

void MPTK_UnPause ()

Pause the current playing

Member Data Documentation

MidiListPlayerStatus MPTK_MidiFilePlayer_1

First [MidiFilePlayer](#) to play the Midi

MidiListPlayerStatus MPTK_MidiFilePlayer_2

Second [MidiFilePlayer](#) to play the Midi

float MPTK_OverlayTimeMS

Duration of overlay between playing two midi

List<[MPTK_MidiPlayItem](#)> MPTK_PlayList

Play list

EventEndMidiClass OnEventEndPlayMidi

Define unity event to trigger at end

EventStartMidiClass OnEventStartPlayMidi

Define unity event to trigger at start

Property Documentation

TimeSpan MPTK_Duration [get]

Duration of the midi. This duration can change during the playing when Change Tempo Event are processed.

bool MPTK_IsPaused [get]

Is Midi file playing is paused ?

bool MPTK_IsPlaying [get]

Is Midi file is playing ?

bool MPTK_Loop [get], [set]

Should automatically restart when Midi reach the end ?

int? MPTK_PlayIndex [get], [set]

Play a specific Midi in the list.

bool MPTK_PlayOnStart [get], [set]

Should the Midi start playing when application start ?

double MPTK_Position [get], [set]

Set or Get midi position time from 0 to lenght time of midi playing (in millisecond). No effect if the Midi is not playing.

```
// Be carefull when modifying position on fly from GUI.
// Each change generates 0.2s of pause, avoid little and frequent position
change.
// Below change is applied only above 2 decimals.
double currentPosition = Math.Round(midiFilePlayer.MPTK_Position / 1000d, 2);
double newPosition =
Math.Round(GUILayout.HorizontalSlider((float)currentPosition, 0f,
(float)midiFilePlayer.MPTK_RealDuration.TotalSeconds,
GUILayout.Width(buttonWidth)), 2);
if (newPosition != currentPosition)
{
    Debug.Log("New position " + currentPosition + "--> " + newPosition);
    midiFilePlayer.MPTK_Position = newPosition * 1000d;
```

```
}  
!
```

long MPTK_TickCurrent [get], [set]

Current tick position in Midi: Time of the current midi event expressed in number of "ticks".
 $MPTK_TickCurrent / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.

long MPTK_TickLast [get]

Last tick position in Midi: Value of the tick for the last midi event in sequence expressed in number of "ticks". $MPTK_TickLast / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.

float MPTK_Volume [get], [set]

Volume of midi playing. Must be ≥ 0 and ≤ 1

MidiListPlayer.MPTK_MidiPlayItem

Define a midi to be added in the list

Public Attributes

- float [EndFrom](#)
Time (ms) position where to end playing the midi file
- int [Index](#)
Position of the Midi in the list. Use method [MPTK_ReIndexMidi\(\)](#) recalculate the index.
- string [MidiName](#)
Midi Name. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the resource folder and open Midi File Setup to automatically integrate Midi in MPTK.
- bool [Selected](#)
Select or unselect this Midi to be played in the list ...)
- float [StartFrom](#)
Time (ms) position where to start playing the midi file
- bool [UIAction](#)

Select or unselect this Midi in the Inspector to apply actions (reorder, delete, ...) NO MORE USED

Detailed Description

Define a midi to be added in the list

Member Data Documentation

float EndFrom

Time (ms) position where to end playing the midi file

int Index

Position of the Midi in the list. Use method [MPTK_ReIndexMidi\(\)](#) recalculate the index.

string MidiName

Midi Name. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.

bool Selected

Select or unselect this Midi to be played in the list ...)

float StartFrom

Time (ms) position where to start playing the midi file

bool UIAction

Select or unselect this Midi in the Inspector to apply actions (reorder, delete, ...) NO MORE USED

MidiLoad

Class for loading a Midi file. No sequencer, no synthesizer, so music playing capabilities. Useful to load all the Midi events from a Midi and process, transform, write them to what you want.

Public Member Functions

- double [MPTK_ConvertTickToTime](#) (long tick)
Convert the tick duration to a real time duration in millisecond regarding the current tempo.
- long [MPTK_ConvertTimeToTick](#) (double time)
Convert a real time duration in millisecond to a number of tick regarding the current tempo.
- bool [MPTK_Load](#) (byte[] datamidi, bool strict=false)
Load Midi from an array of bytes
- bool [MPTK_Load](#) (int index, bool strict=false)
Load Midi from midi MPTK referential (Unity resource). The index of the Midi file can be found in the window "Midi File Setup". Display with menu MPTK / Midi File Setup
- bool [MPTK_Load](#) (string midiname, bool strict=false)
Load Midi from a Midi file from Unity resources. The Midi file must be present in Unity MidiDB resource folder.
- bool [MPTK_LoadFile](#) (string filename, bool strict=false)
Load Midi from a local file
- List< [MPTKEvent](#) > [MPTK_ReadMidiEvents](#) (long fromTicks=0, long toTicks=long.MaxValue)
Read the list of midi events available in the Midi from a ticks position to an end position.

Public Attributes

- int [MPTK_DeltaTicksPerQuarterNote](#)
Read from Midi Header: Delta Ticks Per Quarter Note. Represent the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48 ticks. Also named Division.
- TimeSpan [MPTK_Duration](#)
Duration (TimeSpan) of the midi.
- float [MPTK_DurationMS](#)
Duration (milliseconds) of the midi.
- double [MPTK_InitialTempo](#)
Initial tempo found in the Midi
- int [MPTK_MicrosecondsPerQuarterNote](#)

Read from the SetTempo event: The tempo is given in micro seconds per quarter beat. To convert this to BPM we need to use the following equation: $BPM = 60,000,000 / [tt \cdot tt \cdot tt]$ Warning: this value can change during the playing when a change tempo event is found.
<https://paxstellar.fr/2020/09/11/midi-timing/>

- int [MPTK_No32ndNotesInQuarterNote](#)
 From TimeSignature event: This value specifies the number of 1/32nds of a note happen every MIDI quarter note. It is usually 8 which means that a quarter note happens every quarter note.
<https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_NumberBeatsMeasure](#)
 From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_NumberQuarterBeat](#)
 From TimeSignature event: number of quarter notes in a beat. Equal 2 Power TimeSigDenominator.
<https://paxstellar.fr/2020/09/11/midi-timing/>
- long [MPTK_TickCurrent](#)
 Current tick position in Midi: Time of the current midi event expressed in number of "ticks". $MPTK_TickCurrent / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.
- long [MPTK_TickFirstNote](#)
 Tick for the first note found
- long [MPTK_TickLast](#)
 Last tick position in Midi: Time of the last midi event in sequence expressed in number of "ticks". $MPTK_TickLast / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.
- int [MPTK_TicksInMetronomeClick](#)
 From TimeSignature event: The standard MIDI clock ticks every 24 times every quarter note (crotchet) so a [cc] value of 24 would mean that the metronome clicks once every quarter note. A [cc] value of 6 would mean that the metronome clicks once every 1/8th of a note (quaver).
<https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_TimeSigDenominator](#)
 From TimeSignature event: The denominator specifies the number of quarter notes in a beat. 2 represents a quarter-note, 3 represents an eighth-note, etc. . <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_TimeSigNumerator](#)
 From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. In MIDI the denominator value is stored in a special format. i.e. the real denominator = $2^{[dd]}$ <https://paxstellar.fr/2020/09/11/midi-timing/>
- int [MPTK_TrackCount](#)
 Count of track read in the Midi file

Properties

- double [MPTK_CurrentTempo](#) [get]
Initial tempo found in the Midi

Detailed Description

Class for loading a Midi file. No sequencer, no synthesizer, so music playing capabilities. Useful to load all the Midi events from a Midi and process, transform, write them to what you want.

Member Function Documentation

double MPTK_ConvertTickToTime (long *tick*)

Convert the tick duration to a real time duration in millisecond regarding the current tempo.

Parameters

<i>tick</i>	duration in ticks
-------------	-------------------

Returns

duration in milliseconds

long MPTK_ConvertTimeToTick (double *time*)

Convert a real time duration in millisecond to a number of tick regarding the current tempo.

Parameters

<i>time</i>	duration in milliseconds
-------------	--------------------------

Returns

duration in ticks

bool MPTK_Load (byte[] *datamidi*, bool *strict = false*)

Load Midi from an array of bytes

Parameters

<i>datamidi</i>	byte array midi
<i>strict</i>	If true will error on non-paired note events, default:false

Returns

true if loaded

bool MPTK_Load (int *index*, bool *strict* = false)

Load Midi from midi MPTK referential (Unity resource). The index of the Midi file can be found in the window "Midi File Setup". Display with menu MPTK / Midi File Setup

```
public MidiLoad MidiLoaded;
// .....
MidiLoaded = new MidiLoad();
MidiLoaded.MPTK_Load(14) // index for "Beattles - Michelle"
Debug.Log("Duration:" + MidiLoaded.MPTK_Duration);
!
```

Parameters

<i>index</i>	
<i>strict</i>	If true will error on non-paired note events, default:false

Returns

true if loaded

bool MPTK_Load (string *midiname*, bool *strict* = false)

Load Midi from a Midi file from Unity resources. The Midi file must be present in Unity MidiDB resource folder.

```
public MidiLoad MidiLoaded;
// .....
MidiLoaded = new MidiLoad();
MidiLoaded.MPTK_Load("Beattles - Michelle")
Debug.Log("Duration:" + MidiLoaded.MPTK_Duration);
!
```

Parameters

<i>midiname</i>	Midi file name without path and extension
<i>strict</i>	if true, check strict compliance with the Midi norm

Returns

true if loaded

bool MPTK_LoadFile (string *filename*, bool *strict* = false)

Load Midi from a local file

Parameters

<i>filename</i>	Midi path and filename to load
<i>strict</i>	if true strict respect of the midi norm is checked

Returns

List<[MPTKEvent](#)> MPTK_ReadMidiEvents (long *fromTicks* = 0, long *toTicks* = long.MaxValue)

Read the list of midi events available in the Midi from a ticks position to an end position.

Parameters

<i>fromTicks</i>	ticks start
<i>toTicks</i>	ticks end

Returns

Member Data Documentation

int MPTK_DeltaTicksPerQuarterNote

Read from Midi Header: Delta Ticks Per Quarter Note. Represent the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48 ticks. Also named Division.

TimeSpan MPTK_Duration

Duration (TimeSpan) of the midi.

float MPTK_DurationMS

Duration (milliseconds) of the midi.

double MPTK_InitialTempo

Initial tempo found in the Midi

int MPTK_MicrosecondsPerQuarterNote

Read from the SetTempo event: The tempo is given in micro seconds per quarter beat. To convert this to BPM we need to use the following equation: $BPM = 60,000,000 / [t \cdot t \cdot t]$ Warning: this value can change during the playing when a change tempo event is found.
<https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_No32ndNotesInQuarterNote

From TimeSignature event: This value specifies the number of 1/32nds of a note happen every MIDI quarter note. It is usually 8 which means that a quarter note happens every quarter note.
<https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_NumberBeatsMeasure

From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_NumberQuarterBeat

From TimeSignature event: number of quarter notes in a beat. Equal 2 Power TimeSigDenominator. <https://paxstellar.fr/2020/09/11/midi-timing/>

long MPTK_TickCurrent

Current tick position in Midi: Time of the current midi event expressed in number of "ticks". $MPTK_TickCurrent / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.

long MPTK_TickFirstNote

Tick for the first note found

long MPTK_TickLast

Last tick position in Midi: Time of the last midi event in sequence expressed in number of "ticks". $MPTK_TickLast / MPTK_DeltaTicksPerQuarterNote$ equal the duration time of a quarter-note regardless the defined tempo.

int MPTK_TicksInMetronomeClick

From TimeSignature event: The standard MIDI clock ticks every 24 times every quarter note (crotchet) so a [cc] value of 24 would mean that the metronome clicks once every quarter note. A [cc] value of 6 would mean that the metronome clicks once every 1/8th of a note (quaver). <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_TimeSigDenominator

From TimeSignature event: The denominator specifies the number of quarter notes in a beat. 2 represents a quarter-note, 3 represents an eighth-note, etc. . <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_TimeSigNumerator

From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually

the first beat of each bar has extra emphasis. In MIDI the denominator value is stored in a special format. i.e. the real denominator = $2^{[dd]}$ <https://paxstellar.fr/2020/09/11/midi-timing/>

int MPTK_TrackCount

Count of track read in the Midi file

Property Documentation

double MPTK_CurrentTempo [get]

Initial tempo found in the Midi

MidiPlayerGlobal

Singleton class to manage all global features of MPTK. More information here:

<https://paxstellar.fr/midiplayerglobal/>

Inherits MonoBehaviour.

Static Public Member Functions

- static float [MPTK_DistanceToListener](#) (Transform trf)
Calculate distance with the AudioListener.
- static int [MPTK_FindMidi](#) (string name)
Find index of a Midi by name. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.
- static bool [MPTK_IsReady](#) (float delay=0.5f)
Check if SoudFont is loaded. Add a default wait time because Unity AudioSource need a delay to be really ready to play. Hummm, like a diesel motor ?
- static bool [MPTK_LoadLiveSF](#) (string pPathSF=null, int defaultBank=-1, int drumBank=-1, bool restartPlayer=true)
[MPTK PRO] - Load a SoundFont on the fly when application is running. SoundFont is loaded from a local file or from the web. If some Midis are playing they are restarted. Loading is done in background (coroutine), so method return immediately
- static void [MPTK_Quit](#) ()
Stop all Midi Synthesizer dans Midi Sequencer and exit application
- static void [MPTK_SelectBankDrum](#) (int nbank)

Change current bank on fly

- static void [MPTK_SelectBankInstrument](#) (int nbank)
Change default current bank on fly
- static void [MPTK_SelectSoundFont](#) (string name, bool restartPlayer=true)
[MPTK PRO] - Changing the current Soundfont on fly. If some Midis are playing they are restarted.
- static void [MPTK_Stop](#) ()
Stop all Midi Synthesizer dans Midi Sequencer

Public Attributes

- string [MPTK_LiveSoundFont](#)
[MPTK PRO] - Full path to SoundFont file (.sf2) or URL to load. Defined in the [MidiPlayerGlobal](#) editor inspector. Must start with [file://](#) or [http://](#) or [https://](#).

Static Public Attributes

- static int [MPTK_CountWaveLoaded](#)
Count of wave loaded
- static List< MPTKListItem > [MPTK_ListBank](#)
Get the list of banks available
- static List< MPTKListItem > [MPTK_ListDrum](#)
Get the list of presets available
- static List< MPTKListItem > [MPTK_ListMidi](#)
List of midi(s) available
- static List< MPTKListItem > [MPTK_ListPreset](#)
Get the list of presets available for instruments for the selected bank
- static List< MPTKListItem > [MPTK_ListPresetDrum](#)
Get the list of presets available for instrument
- static bool [MPTK_SoundFontLoaded](#) = false
True if soundfont is loaded

Properties

- static int [MPTK_CountPresetLoaded](#) [get]
Count of preset loaded
- static List< string > [MPTK_ListSoundFont](#) [get]
List of Soundfont(s) available

- static bool? [MPTK_LoadSoundFontAtStartup](#) [get, set]
If true load soundfont when startup
- static bool? [MPTK_LoadWaveAtStartup](#) [get, set]
If true load all waves when application is started else load when need when playing (default)
- static string [MPTK_PathToResources](#) [get]
This path could change depending your project. Change the path before any actions in MPTK. DEPRECATED, WILL BE REMOVED.
- static TimeSpan [MPTK_TimeToLoadSoundFont](#) [get]
Load time for the current SoundFont
- static TimeSpan [MPTK_TimeToLoadWave](#) [get]
Load time for the wave
- static UnityEvent? [OnEventPresetLoaded](#) [get, set]
Event triggered at end of loading a soundfont. Warning: when defined by script, this event is not triggered at first load of MPTK because [MidiPlayerGlobal](#) is loaded before any other gamecomponent. Set this event in the Inspector of [MidiPlayerGlobal](#) to get at first load this information.

Detailed Description

Singleton class to manage all global features of MPTK. More information here: <https://paxstellar.fr/midiplayerglobal/>

Singleton class to manage all global features of MPTK.

Member Function Documentation

static float MPTK_DistanceToListener (Transform *trf*) [static]

Calculate distance with the AudioListener.

Parameters

<i>trf</i>	Transform of the object to calculate the distance.
------------	--

Returns

static int MPTK_FindMidi (string *name*) [static]

Find index of a Midi by name. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.

Parameters

<i>name</i>	name of the midi without path nor extension
-------------	---

Returns

-1 if not found else return the index of the midi.

static bool MPTK_IsReady (float *delay* = 0.5f) [static]

Check if SoudFont is loaded. Add a default wait time because Unity AudioSource need a delay to be really ready to play. Hummm, like a diesel motor ?

Parameters

<i>delay</i>	
--------------	--

Returns

static bool MPTK_LoadLiveSF (string *pPathSF* = null, int *defaultBank* = -1, int *drumBank* = -1, bool *restartPlayer* = true) [static]

[MPTK PRO] - Load a SoundFont on the fly when application is running. SoundFont is loaded from a local file or from the web. If some Midis are playing they are restarted. Loading is done in background (coroutine), so method return immediately

Parameters

<i>pPathSF</i>	Full path to SoudFont file. Must start with file:// for local desktop loading or with or http:// or https:// for loading from web resource. if null, use MPTK_LiveSoundFont
<i>defaultBank</i>	default bank to use for instrument, default or -1 to select the first bank
<i>drumBank</i>	bank to use for drum kit, default or -1 to select the last bank
<i>restartPlayer</i>	Restart midi player if need, default is true

Returns

true if loading is in progress, false if an error is detected in parameters

static void MPTK_Quit () [static]

Stop all Midi Synthesizer dans Midi Sequencer and exit application

static void MPTK_SelectBankDrum (int *nbank*) [static]

Change current bank on fly

Parameters

<i>nbank</i>	Number of the SoundFont Bank to load for drum.
--------------	--

static void MPTK_SelectBankInstrument (int *nbank*) [static]

Change default current bank on fly

Parameters

<i>nbank</i>	Number of the SoundFont Bank to load for instrument.
--------------	--

static void MPTK_SelectSoundFont (string *name*, bool *restartPlayer* = true)[static]

[MPTK PRO] - Changing the current Soundfont on fly. If some Midis are playing they are restarted.

Parameters

<i>name</i>	SoundFont name
<i>restartPlayer</i>	if a midi is playing, restart the current playing midi

static void MPTK_Stop () [static]

Stop all Midi Synthesizer dans Midi Sequencer

Member Data Documentation

int MPTK_CountWaveLoaded [static]

Count of wave loaded

List<MPTKListItem> MPTK_ListBank [static]

Get the list of banks available

List<MPTKListItem> MPTK_ListDrum [static]

Get the list of presets available

List<MPTKListItem> MPTK_ListMidi [static]

List of midi(s) available

List<MPTKListItem> MPTK_ListPreset [static]

Get the list of presets available for instruments for the selected bank

List<MPTKListItem> MPTK_ListPresetDrum [static]

Get the list of presets available for instrument

string MPTK_LiveSoundFont

[MPTK PRO] - Full path to SoundFont file (.sf2) or URL to load. Defined in the [MidiPlayerGlobal](#) editor inspector. Must start with [file://](#) or [http://](#) or [https://](#).

bool MPTK_SoundFontLoaded = false [static]

True if soundfont is loaded

Property Documentation

int MPTK_CountPresetLoaded [static], [get]

Count of preset loaded

List<string> MPTK_ListSoundFont [static], [get]

List of Soundfont(s) available

bool? MPTK_LoadSoundFontAtStartup [static], [get], [set]

If true load soundfont when startup

bool? MPTK_LoadWaveAtStartup [static], [get], [set]

If true load all waves when application is started else load when need when playing (default)

string MPTK_PathToResources [static], [get]

This path could change depending your project. Change the path before any actions in MPTK. DEPRECATED, WILL BE REMOVED.

TimeSpan MPTK_TimeToLoadSoundFont [static], [get]

Load time for the current SoundFont

TimeSpan MPTK_TimeToLoadWave [static], [get]

Load time for the wave

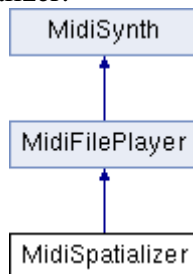
UnityEvent? OnEventPresetLoaded [static], [get], [set]

Event triggered at end of loading a soundfont. Warning: when defined by script, this event is not triggered at first load of MPTK because [MidiPlayerGlobal](#) is loaded before any other gamecomponent. Set this event in the Inspector of [MidiPlayerGlobal](#) to get at first load this information.

MidiSpatializer

[MPTK PRO] - Script associated to the prefab [MidiSpatializer](#). It's quite light because the major job is done with [MidiSynth](#) There is no specific API for this prefab. Scripting is necessary to defined position of channel or instrument in your 3D env.

Inheritance diagram for MidiSpatializer:



Additional Inherited Members

Detailed Description

[MPTK PRO] - Script associated to the prefab [MidiSpatializer](#). It's quite light because the major job is done with [MidiSynth](#) There is no specific API for this prefab. Scripting is necessary to defined position of channel or instrument in your 3D env.

```
public void ArrangeInLine(bool fromUI)
{
    isPositionByInstrument = false;
    //Debug.Log($"ArrangeInLine {midiFilePlayer.MPTK_DedicatedChannel}");
    if (!fromUI)
    {
        // Get the synth associate to this channel
```

```

        TestSpatializerFly tsf =
midiSpatializer.gameObject.GetComponent<TestSpatializerFly>();
        tsf.PosSynth = new Vector3((midiSpatializer.MPTK_DedicatedChannel *
118) - 950, tsf.PosSynth.y, 0f);
    }
    else
    {
        // Exec from the UI, applied to each MidiFilePlayer (MidiSynth)
        foreach (MidiFilePlayer mfp in MidiFilePlayer.SpatialSynths)
        {
            TestSpatializerFly tsf =
mfp.gameObject.GetComponent<TestSpatializerFly>();
            tsf.PosSynth = new Vector3((mfp.MPTK_DedicatedChannel * 118) -
950, tsf.PosSynth.y, 0f);
        }
    }
}

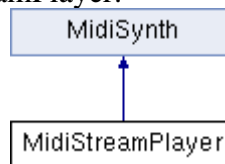
```

See full example in TestSpatializerFly.cs Available with V2.83.

MidiStreamPlayer

Play generated notes. Any Midi file is necessary rather create music from your own algorithm with [MPTK_PlayEvent\(\)](#). Duration can be set in the [MPTKEvent](#), but a note can also be stopped with [MPTK_StopEvent\(\)](#).

Inheritance diagram for MidiStreamPlayer:



Public Member Functions

- [MPTKChordBuilder MPTK_PlayChordFromLib](#) ([MPTKChordBuilder](#) chord)
[MPTK PRO] Play a chord from the chord library. See file ChordLib.csv in folder Resources/GeneratorTemplate. The Tonic is used to build the chord
- [MPTKChordBuilder MPTK_PlayChordFromRange](#) ([MPTKChordBuilder](#) chord)
[MPTK PRO] Play a chord from the current selected range (MPTK_RangeSelected), Tonic and Degree defined in parameter MPTKChord chord. Major range is selected if no range defined. See file GammeDefinition.csv in folder Resources/GeneratorTemplate
- void [MPTK_PlayEvent](#) (List< [MPTKEvent](#) > events)
Play a list of midi events with a thread so the call return immediately.
- void [MPTK_PlayEvent](#) ([MPTKEvent](#) evnt)
Play one midi event with a thread so the call return immediately.
- void [MPTK_StopChord](#) ([MPTKChordBuilder](#) chord)
Stop playing the chord. All samples associated to the chord are stopped by sending a noteoff.
- void [MPTK_StopEvent](#) ([MPTKEvent](#) pnote)
Stop playing the note. All waves associated to the note are stop by sending a noteoff.

Properties

- string? [MPTK_RangeName](#) [get]
Name of range selected (musical scale)
- int [MPTK_RangeSelected](#) [get, set]
Current selected range (musical scale)

Additional Inherited Members

Detailed Description

Play generated notes. Any Midi file is necessary rather create music from your own algorithm with [MPTK_PlayEvent\(\)](#). Duration can be set in the [MPTKEvent](#), but a note can also be stopped with [MPTK_StopEvent\(\)](#).

Member Function Documentation

[MPTKChordBuilder](#) [MPTK_PlayChordFromLib](#) ([MPTKChordBuilder](#) *chord*)

[MPTK PRO] Play a chord from the chord library. See file ChordLib.csv in folder Resources/GeneratorTemplate. The Tonic is used to build the chord

Parameters

<i>chord</i>	required: Tonic and FromLib on top of the classical Midi parameters
--------------	---

Returns

```
//private void PlayOneChordFromLib()
//{
//    // Start playing a new chord
//    ChordLibPlaying = new MPTKChordBuilder(true)
//    {
//        // Parameters to build the chord
//        Tonic = CurrentNote,
//        FromLib = CurrentChord,
//
//        // Midi Parameters how to play the chord
//        Channel = StreamChannel,
//        // delay in milliseconds between each notes of the chord
//        Arpeggio = ArpeggioPlayChord,
//        // millisecond, -1 to play indefinitely
//        Duration = Convert.ToInt64(NoteDuration * 1000f),
//        // Sound can vary depending on the velocity
//        Velocity = Velocity,
//        Delay = Convert.ToInt64(NoteDelay * 1000f),
//    };
//    midiStreamPlayer.MPTK_PlayChordFromLib(ChordLibPlaying);
//}
```

[MPTKChordBuilder](#) [MPTK_PlayChordFromRange](#) ([MPTKChordBuilder](#) *chord*)

[MPTK PRO] Play a chord from the current selected range (MPTK_RangeSelected), Tonic and Degree defined in parameter MPTKChord chord. Major range is selected if no range defined. See file GammeDefinition.csv in folder Resources/GeneratorTemplate

Parameters

<i>chord</i>	required: Tonic and Degree on top of the classical Midi parameters
--------------	--

Returns

```

/*
private void PlayOneChord()
{
    // Start playing a new chord
    ChordPlaying = new MPTKChordBuilder(true)
    {
        // Parameters to build the chord
        Tonic = CurrentNote,
        Count = CountNoteChord,
        Degree = DegreeChord,

        // Midi Parameters how to play the chord
        Channel = StreamChannel,
        //delay in milliseconds between each notes of the chord
        Arpeggio = ArpeggioPlayChord,
        // millisecond, -1 to play indefinitely
        Duration = Convert.ToInt64(NoteDuration * 1000f),
        //Sound can vary depending on the velocity
        Velocity = Velocity,
        Delay = Convert.ToInt64(NoteDelay * 1000f),
    };
    midiStreamPlayer.MPTK_PlayChordFromRange(ChordPlaying);
}
*/
*

```

void MPTK_PlayEvent (List< [MPTKEvent](#) > events)

Play a list of midi events with a thread so the call return immediately.

```

private void PlayOneNote()
{
    //Debug.Log($"{StreamChannel}
{midiStreamPlayer.MPTK_ChannelPresetGetName(StreamChannel)}");
    // Start playing a new note
    NotePlaying = new MPTKEvent()
    {
        Command = MPTKCommand.NoteOn,
        Value = CurrentNote,
        Channel = StreamChannel,
        Duration = Convert.ToInt64(NoteDuration * 1000f), // millisecond,
-1 to play indefinitely
        Velocity = Velocity, // Sound can vary depending on the velocity
        Delay = Convert.ToInt64(NoteDelay * 1000f),
    };
    midiStreamPlayer.MPTK_PlayEvent(NotePlaying);
}

```

void MPTK_PlayEvent ([MPTKEvent](#) evt)

Play one midi event with a thread so the call return immediately.

```

midiStreamPlayer.MPTK_PlayEvent
(
    new MPTKEvent()
    {
        Channel = 9,
        Duration = 999999,
        Value = 48,
        Velocity = 100
    }
);

```

void MPTK_StopChord ([MPTKChordBuilder](#) *chord*)

Stop playing the chord. All samples associated to the chord are stopped by sending a noteoff.

Parameters

<i>chord</i>	
--------------	--

void MPTK_StopEvent ([MPTKEvent](#) *pnote*)

Stop playing the note. All waves associated to the note are stop by sending a noteoff.

Parameters

<i>pnote</i>	
--------------	--

Property Documentation

string? MPTK_RangeName [get]

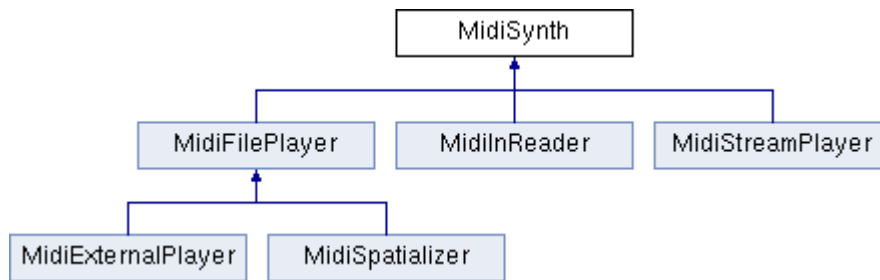
Name of range selected (musical scale)

int MPTK_RangeSelected [get], [set]

Current selected range (musical scale)

MidiSynth

Inheritance diagram for MidiSynth:



Public Member Functions

- int [MPTK_ChannelBankGetIndex](#) (int channel)
Get channel bank.
- int [MPTK_ChannelCount](#) ()
Get channel count. The midi norm is 16, but MPTK can manage up to 32 channels.
- bool [MPTK_ChannelEnableGet](#) (int channel)
Is channel is enabled or disabled.
- void [MPTK_ChannelEnableSet](#) (int channel, bool enable)
Enable or disable a channel.
- int [MPTK_ChannelForcedPresetGet](#) (int channel)
Set forced preset on the channel. Midi will always playing with this preset even if a Midi Preset Change message is received . Set to -1 to disable this behavior.
- bool [MPTK_ChannelForcedPresetSet](#) (int channel, int preset)
Set forced preset on the channel. Midi will always playing with this preset even if a Midi Preset Change message is received . Set to -1 to disable this behavior.
- int [MPTK_ChannelNoteCount](#) (int channel)
Get count of notes played since the start of the Midi.
- bool [MPTK_ChannelPresetChange](#) (int channel, int preset, int newbank=-1)
Change the preset and bank for the channel. When playing a Midi file, the preset is set by channel with the Midi message Patch Change. The bank is changed with a ControlChange Midi message.

The new value of the bank is local for the channel, the preset list is not updated. To change globally the bank, use instead the golbal methods: [MidiPlayerGlobal.MPTK_SelectBankInstrument](#) or [MidiPlayerGlobal.MPTK_SelectBankDrum](#)
- int [MPTK_ChannelPresetGetIndex](#) (int channel)
Get channel preset indx.
- string [MPTK_ChannelPresetGetName](#) (int channel)
Get channel current preset name.
- float [MPTK_ChannelVolumeGet](#) (int channel)
Get the volume of the channel

- void [MPTK_ChannelVolumeSet](#) (int channel, float volume)
Set the volume for a channel (between 0 and 1). New with V2.82, works only in Core mode.
- void [MPTK_ChorusSetDefault](#) ()
[MPTK PRO] - Set Chorus Unity default value as defined with Unity.
- void [MPTK_ClearAllSound](#) (bool destroyAudioSource=false, int _idSession=-1)
Clear all sound by sending note off. That could take some seconds because release time for sample need to be played.
- void [MPTK_InitSynth](#) (int channelCount=16)
Initialize the synthesizer: channel, voices, modulator. It's not usefull to call this method if you are using prefabs ([MidiFilePlayer](#), [MidiStreamPlayer](#), ...). Each gameObjects created from these prefabs have their own, autonomous and isolated synth.
- void [MPTK_ResetStat](#) ()
Reset voices statistics
- void [MPTK_ReverbSetDefault](#) ()
[MPTK PRO] - Set Reverb Unity default value as defined with Unity.
- void [MPTK_StartSequencerMidi](#) ()
Start the Midi sequencer: each midi events are read and play in a dedicated thread. This thread is automatically started by prefabs [MidiFilePlayer](#), [MidiListPlayer](#), [MidiExternalPlayer](#).
- void [MPTK_StopSynth](#) ()
Stop processing samples by the synth and the Midi sequencer.
- IEnumerator [MPTK_WaitAllNotesOff](#) (int _idSession=-1)
*Wait until all notes are off. That could take some seconds due to the samples release time. Therefore, the method exit after a timeout of 3 seconds. *** Use this method only as a coroutine ****

Public Attributes

- bool [MPTK_ApplyModLfo](#)
Apply LFO effect defined in the SoundFont
- bool [MPTK_ApplyRealTimeModulator](#)
Apply real time modulatoreffect defined in the SoundFont: pitch bend, control change, envelope modulation
- bool [MPTK_ApplySFChorus](#)
[MPTK PRO] - Apply chorus effect as defined in the SoundFont. This effect is processed with the fluidsynth also independently on each voices but with a small decrease of performace (10%).
- bool [MPTK_ApplySFFilter](#)

[MPTK PRO] - Apply frequency low-pass filter as defined in the SoundFont. This effect is processed with the fluidsynth algo independently on each voices but with a small decrease of performace (40%).

- bool [MPTK_ApplySFReverb](#)
[MPTK PRO] - Apply reverberation effect as defined in the SoundFont. This effect is processed with the fluidsynth algo independently on each voices but with a small decrease of performace (40%).
- bool [MPTK_ApplyVibLfo](#)
Apply vibrato effect defined in the SoundFont
- int [MPTK_AutoCleanVoiceLimit](#)
Free voices older than MPTK_AutoCleanVoiceLimit are removed when count is over than MPTK_AutoCleanVoiceTime
- bool [MPTK_CorePlayer](#)
If true then Midi events are read and play from a dedicated thread. If false, [MidiSynth](#) will use AudioSource gameobjects to play sound. This properties must be defined before running the application from the inspector. The default is true. The non core mode player will be removed with the next major version (V3)
- bool [MPTK_DirectSendToPlayer](#)
If true (default) then Midi events are sent automatically to the midi player. Set to false if you want to process events without playing sound. OnEventNotesMidi Unity Event can be used to process each notes.
- bool [MPTK_EnableChangeTempo](#)
Should accept change tempo from Midi Events ?
- bool [MPTK_EnablePanChange](#)
Should change pan from Midi Events or from SoundFont ? Pan is disabled when Spatialization is activated.
- bool [MPTK_EnablePresetDrum](#)
Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.
- bool [MPTK_KillByExclusiveClass](#) = true
V2.83 Find the exclusive class of this voice. If set, kill all voices that match the exclusive class and are younger than the first voice process created by this noteon event.
- bool [MPTK_LogWave](#)
Log for each wave to be played
- bool [MPTK_ReleaseSameNote](#) = true
V2.83. If the same note is hit twice on the same channel, then the older voice process is advanced to the release stage. It's the default Midi processing.
- uint [MPTK_ReleaseTimeMin](#) = 500000

[Only when CorePlayer=False] Define a minimum release time at noteoff in 100 iem nanoseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard. Useless when MPTK_CorePlayer is true.

- float [MPTK_ReleaseTimeMod](#) = 1f
Multiplier to increase or decrease the default release time defined in the SoundFont. Recommended values between 0.1 and 2. Default is 1 (no modification of the release time)
- float [MPTK_SFChorusAmplify](#)
[MPTK PRO] - Chorus level is defined in the SoundFont for each notes. This parameter increase or decrease the default SoundFont value.
- float [MPTK_SFFilterFreqOffset](#) = 0f
[MPTK PRO] - Frequency cutoff is defined in the SoundFont for each notes. This parameter increase or decrease the default SoundFont value.
- float [MPTK_SFReverbAmplify](#)
[MPTK PRO] - Reverberation level is defined in the SoundFont for each notes. This parameter increase or decrease the default SoundFont value.
- int [MPTK_StatVoiceCountActive](#)
Count of the active voices (playing) - Readonly
- int [MPTK_StatVoiceCountFree](#)
Count of the free voices for reusing on need. Older than AutoCleanVoiceTime are removed when count is over than AutoCleanVoiceLimit - Readonly
- int [MPTK_StatVoicePlayed](#)
Count of voice played since the start of the synth
- float [MPTK_StatVoiceRatioReused](#)
Percentage of voice reused during the synth life. 0: any reuse, 100:all voice reused (unattainable, of course!)
- bool [MPTK_WeakDevice](#)
Should play on a weak device (cheaper smartphone) ? Apply only with AudioSource mode (MPTK_CorePlayer=False) Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.
- EventSynthClass [OnEventSynthAwake](#)
Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.
- EventSynthClass [OnEventSynthStarted](#)
Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.
- bool [playOnlyFirstWave](#)
Preset are often composed with 2 or more samples. Classically for left and right channel. Check this to play only the first sample found

Static Public Attributes

- static [MidiFilePlayer\[\] SpatialSynths](#)
Contains each [MidiSynth](#) for each channel when the prefab [MidiSpatializer](#) is used and `IsMidiChannelSpace=true`. Warning: only one [MidiSpatializer](#) can be used in a hierarchy.

Properties

- bool [MPTK_ApplyUnityChorus](#) [get, set]
[MPTK PRO] - Apply Chorus Unity effect to the AudioSource. The effect is applied to all voices.
- bool [MPTK_ApplyUnityReverb](#) [get, set]
[MPTK PRO] - Apply Reverb Unity effect to the AudioSource. The effect is applied to all voices.
- float [MPTK_ChorusDelay](#) [get, set]
[MPTK PRO] - Chorus delay in ms. 0.1 to 100. Default = 40 ms.
- float [MPTK_ChorusDepth](#) [get, set]
[MPTK PRO] - Chorus modulation depth. 0 to 1. Default = 0.03.
- float [MPTK_ChorusDryMix](#) [get, set]
[MPTK PRO] - Volume of original signal to pass to output. 0 to 1. Default = 0.5.
- float [MPTK_ChorusRate](#) [get, set]
[MPTK PRO] - Chorus modulation rate in hz. 0 to 20. Default = 0.8 hz.
- float [MPTK_ChorusWetMix1](#) [get, set]
[MPTK PRO] - Volume of 1st chorus tap. 0 to 1. Default = 0.5.
- float [MPTK_ChorusWetMix2](#) [get, set]
[MPTK PRO] - Volume of 2nd chorus tap. This tap is 90 degrees out of phase of the first tap. 0 to 1. Default = 0.5.
- float [MPTK_ChorusWetMix3](#) [get, set]
[MPTK PRO] - Volume of 3rd chorus tap. This tap is 90 degrees out of phase of the second tap. 0 to 1. Default = 0.5.
- int [MPTK_DedicatedChannel](#) [get]
Dedicated Channel for this [MidiSynth](#) when the prefab [MidiSpatializer](#) is used. The [MidiSynth](#) reader (from a midi file) has no channel because no voice is played, so `DedicatedChannel` is set to -1
- int [MPTK_IndexSynthBuffSize](#) [get, set]
Set or Get sample rate output of the synth. -1:default, 0:24000, 1:36000, 2:48000, 3:60000, 4:72000, 5:84000, 6:96000. It's better to stop playing before changing on fly to avoid bad noise.
- int [MPTK_IndexSynthRate](#) [get, set]

Set or Get sample rate output of the synth. -1:default, 0:24000, 1:36000, 2:48000, 3:60000, 4:72000, 5:84000, 6:96000. It's better to stop playing before changing on fly to avoid bad noise.

- float [MPTK_MaxDistance](#) [get, set]
If MPTK_Spatialize is enabled, the volume of the audio source depends on the distance between the audio source and the listener. Beyond this distance, the volume is set to 0 and the midi player is paused. No effect if MPTK_Spatialize is disabled.
- bool [MPTK_PauseOnDistance](#) [get, set]
[obsolete] replaced by "MPTK_Spatialize"; V2.83
- float [MPTK_ReverbDecayHFRatio](#) [get, set]
[MPTK PRO] - Decay HF Ratio : High-frequency to low-frequency decay time ratio. Ranges from 0.1 to 2.0.
- float [MPTK_ReverbDecayTime](#) [get, set]
[MPTK PRO] - Reverberation decay time at low-frequencies in seconds. Ranges from 0.1 to 20. Default is 1.
- float [MPTK_ReverbDelay](#) [get, set]
[MPTK PRO] - Late reverberation delay time relative to first reflection in seconds. Ranges from 0 to 0.1. Default is 0.04
- float [MPTK_ReverbDensity](#) [get, set]
[MPTK PRO] - Reverberation density (modal density) in percent. Ranges from 0 to 1.
- float [MPTK_ReverbDiffusion](#) [get, set]
[MPTK PRO] - Reverberation diffusion (echo density) in percent. Ranges from 0 to 1. Default is 1.
- float [MPTK_ReverbDryLevel](#) [get, set]
[MPTK PRO] - Mix level of dry signal in output. Ranges from 0 to 1.
- float [MPTK_ReverbHFReference](#) [get, set]
[MPTK PRO] - Reference high frequency in Hz. Ranges from 1000 to 20000. Default is 5000
- float [MPTK_ReverbLevel](#) [get, set]
[MPTK PRO] - Late reverberation level relative to room effect. Ranges from 0 to 1.
- float [MPTK_ReverbLFReference](#) [get, set]
[MPTK PRO] - Reference low-frequency in Hz. Ranges from 20 to 1000. Default is 250
- float [MPTK_ReverbReflectionDelay](#) [get, set]
[MPTK PRO] - Late reverberation level relative to room effect. Ranges from -10000.0 to 2000.0. Default is 0.0.
- float [MPTK_ReverbReflectionLevel](#) [get, set]
[MPTK PRO] - Early reflections level relative to room effect. Ranges from 0 to 1.

- float [MPTK_ReverbRoom](#) [get, set]
[MPTK PRO] - Room effect level at low frequencies. Ranges from 0 to 1.
- float [MPTK_ReverbRoomHF](#) [get, set]
[MPTK PRO] - Room effect high-frequency level. Ranges from 0 to 1.
- float [MPTK_ReverbRoomLF](#) [get, set]
[MPTK PRO] - Room effect low-frequency level. Ranges from 0 to 1.
- float [MPTK_SFFilterQModOffset](#) [get, set]
[MPTK PRO] - Quality Factor is defined in the SoundFont for each notes. This parameter increase or decrease the default SoundFont value.
- bool [MPTK_Spatialize](#) [get, set]
Should the Spatialization effect must be enabled? See here how to setup spatialization with Unity <https://paxstellar.fr/midi-file-player-detailed-view-2/#Foldout-Spatialization-Parameters>
- int [MPTK_SynthRate](#) [get]
Current synth rate defined
- int [MPTK_Transpose](#) [get, set]
Transpose note from -24 to 24
- float [MPTK_Volume](#) [get, set]
Volume of midi playing. Must be >=0 and <= 1

Detailed Description

[MPTK PRO] - class extention

Base class for Midi Synthesizer. Migrated from fluidsynth. It's not recommended to instanciate this class. Instead use [MidiFilePlayer](#) or [MidiStreamPlayer](#).

Member Function Documentation

int MPTK_ChannelBankGetIndex (int *channel*)

Get channel bank.

Parameters

<i>channel</i>	must be between 0 and 15
----------------	--------------------------

int MPTK_ChannelCount ()

Get channel count. The midi norm is 16, but MPTK can manage up to 32 channels.

Parameters

<i>channel</i>	must be between 0 and 15
----------------	--------------------------

Returns

channel count

bool MPTK_ChannelEnableGet (int *channel*)

Is channel is enabled or disabled.

Parameters

<i>channel</i>	channel, must be between 0 and 15
----------------	-----------------------------------

Returns

true if channel is enabled

void MPTK_ChannelEnableSet (int *channel*, bool *enable*)

Enable or disable a channel.

Parameters

<i>channel</i>	must be between 0 and 15
<i>enable</i>	true to enable

int MPTK_ChannelForcedPresetGet (int *channel*)

Set forced preset on the channel. Midi will always playing with this preset even if a Midi Preset Change message is received . Set to -1 to disable this behavior.

Parameters

<i>channel</i>	
----------------	--

Returns

preset index, -1 if not set

```
GUILayout.Label($"{midiFilePlayer.MPTK_ChannelForcedPresetGet(channel)}/{midiFileP  
layer.MPTK_ChannelBankGetIndex(channel)}");  
int forced =  
    (int)GUILayout.HorizontalSlider(  
        midiFilePlayer.MPTK_ChannelForcedPresetGet(channel),  
        -1f, 127f);  
if (forced != midiFilePlayer.MPTK_ChannelForcedPresetGet(channel))  
    midiFilePlayer.MPTK_ChannelForcedPresetSet(channel, forcedPreset);  
!
```

bool MPTK_ChannelForcedPresetSet (int *channel*, int *preset*)

Set forced preset on the channel. Midi will always playing with this preset even if a Midi Preset Change message is received . Set to -1 to disable this behavior.

Parameters

<i>channel</i>	0 to 15 channel
<i>preset</i>	0 to 127 preset

Returns

```
GUILayout.Label($"{midiFilePlayer.MPTK_ChannelForcedPresetGet(channel)}/{midiFileP  
layer.MPTK_ChannelBankGetIndex(channel)}");  
int forced =  
    (int)GUILayout.HorizontalSlider(  
        midiFilePlayer.MPTK_ChannelForcedPresetGet(channel),  
        -1f, 127f);  
if (forced != midiFilePlayer.MPTK_ChannelForcedPresetGet(channel))  
    midiFilePlayer.MPTK_ChannelForcedPresetSet(channel, forcedPreset);  
!
```

int MPTK_ChannelNoteCount (int *channel*)

Get count of notes played since the start of the Midi.

Parameters

<i>channel</i>	must be between 0 and 15
----------------	--------------------------

bool MPTK_ChannelPresetChange (int *channel*, int *preset*, int *newbank* = -1)

Change the preset and bank for the channel. When playing a Midi file, the preset is set by channel with the Midi message Patch Change. The bank is changed with a ControlChange Midi message.

The new value of the bank is local for the channel, the preset list is not updated. To change globally the bank, use instead the golbal methods: [MidiPlayerGlobal.MPTK_SelectBankInstrument](#) or [MidiPlayerGlobal.MPTK_SelectBankDrum](#)

Parameters

<i>channel</i>	0 to 15. There is 16 channels available in the Midi norm.
<i>preset</i>	The count of presets is dependant of the soundfont selected
<i>newbank</i>	optionnal, use the default bank defined globally

Returns

true if preset change is done

int MPTK_ChannelPresetGetIndex (int *channel*)

Get channel preset indx.

Parameters

<i>channel</i>	must be between 0 and 15
----------------	--------------------------

string MPTK_ChannelPresetGetName (int *channel*)

Get channel current preset name.

Parameters

<i>channel</i>	must be between 0 and 15
----------------	--------------------------

float MPTK_ChannelVolumeGet (int *channel*)

Get the volume of the channel

Parameters

<i>channel</i>	must be between 0 and 15
----------------	--------------------------

Returns

volume of the channel, between 0 and 1

void MPTK_ChannelVolumeSet (int *channel*, float *volume*)

Set the volume for a channel (between 0 and 1). New with V2.82, works only in Core mode.

Parameters

<i>channel</i>	must be between 0 and 15
<i>volume</i>	volume for the channel, must be between 0 and 1

void MPTK_ChorusSetDefault ()

[MPTK PRO] - Set Chorus Unity default value as defined with Unity.

void MPTK_ClearAllSound (bool *destroyAudioSource* = false, int *_idSession* = -1)

Clear all sound by sending note off. That could take some seconds because release time for sample need to be played.

Parameters

<i>destroyAudioSource</i> <i>e</i>	usefull only in non core mode
<i>_idSession</i>	clear only for sample playing with this session, -1 for all (default)

```
if (GUILayout.Button("Clear"))
    midiStreamPlayer.MPTK_ClearAllSound(true);
!
```

///

void MPTK_InitSynth (int *channelCount* = 16)

Initialize the synthesizer: channel, voices, modulator. It's not usefull to call this method if you are using prefabs ([MidiFilePlayer](#), [MidiStreamPlayer](#), ...). Each gameObjects created from these prefabs have their own, autonomous and isolated synth.

Parameters

<i>channelCount</i>	Number of channel to create, default 16. Any other values are experimental!
---------------------	---

void MPTK_ResetStat ()

Reset voices statistics

void MPTK_ReverbSetDefault ()

[MPTK PRO] - Set Reverb Unity default value as defined with Unity.

void MPTK_StartSequencerMidi ()

Start the Midi sequencer: each midi events are read and play in a dedicated thread. This thread is automatically started by prefabs [MidiFilePlayer](#), [MidiListPlayer](#), [MidiExternalPlayer](#).

void MPTK_StopSynth ()

Stop processing samples by the synth and the Midi sequencer.

IEnumerator MPTK_WaitAllNotesOff (int *_idSession* = -1)

Wait until all notes are off. That could take some seconds due to the samples release time. Therefore, the method exit after a timeout of 3 seconds. *** Use this method only as a coroutine ***

```
// Call this method with: StartCoroutine(NextPreviousWithWait(false));
// See TestMidiFilePlayerScripting.cs
public IEnumerator NextPreviousWithWait(bool next)
{
    midiFilePlayer.MPTK_Stop();
    yield return midiFilePlayer.MPTK_WaitAllNotesOff(midiFilePlayer.IdSession);
    if (next)
        midiFilePlayer.MPTK_Next();
    else
        midiFilePlayer.MPTK_Previous();
    CurrentIndexPlaying = midiFilePlayer.MPTK_MidiIndex;
    yield return 0;
}
!
```

Parameters

<i>_idSession</i>	clear only for samples playing with this session, -1 for all
-------------------	--

Returns

Member Data Documentation

bool MPTK_ApplyModLfo

Apply LFO effect defined in the SoundFont

bool MPTK_ApplyRealTimeModulator

Apply real time modulator effect defined in the SoundFont: pitch bend, control change, envelope modulation

bool MPTK_ApplySFChorus

[MPTK PRO] - Apply chorus effect as defined in the SoundFont. This effect is processed with the fluidsynth algo independently on each voices but with a small decrease of performance (10%).

bool MPTK_ApplySFFilter

[MPTK PRO] - Apply frequency low-pass filter as defined in the SoundFont. This effect is processed with the fluidsynth algo independently on each voices but with a small decrease of performance (40%).

bool MPTK_ApplySFReverb

[MPTK PRO] - Apply reverberation effect as defined in the SoundFont. This effect is processed with the fluidsynth algo independently on each voices but with a small decrease of performance (40%).

bool MPTK_ApplyVibLfo

Apply vibrato effect defined in the SoundFont

int MPTK_AutoCleanVoiceLimit

Free voices older than MPTK_AutoCleanVoiceLimit are removed when count is over than MPTK_AutoCleanVoiceTime

bool MPTK_CorePlayer

If true then Midi events are read and play from a dedicated thread. If false, [MidiSynth](#) will use AudioSource gameobjects to play sound. This properties must be defined before running the

application from the inspector. The default is true. The non core mode player will be removed with the next major version (V3)

bool MPTK_DirectSendToPlayer

If true (default) then Midi events are sent automatically to the midi player. Set to false if you want to process events without playing sound. OnEventNotesMidi Unity Event can be used to process each notes.

bool MPTK_EnableChangeTempo

Should accept change tempo from Midi Events ?

bool MPTK_EnablePanChange

Should change pan from Midi Events or from SoundFont ? Pan is disabled when Spatialization is activated.

bool MPTK_EnablePresetDrum

Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.

bool MPTK_KillByExclusiveClass = true

V2.83 Find the exclusive class of this voice. If set, kill all voices that match the exclusive class and are younger than the first voice process created by this noteon event.

bool MPTK_LogWave

Log for each wave to be played

bool MPTK_ReleaseSameNote = true

V2.83. If the same note is hit twice on the same channel, then the older voice process is advanced to the release stage. It's the default Midi processing.

uint MPTK_ReleaseTimeMin = 500000

[Only when CorePlayer=False] Define a minimum release time at noteoff in 100 iem nanoseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard. Useless when MPTK_CorePlayer is true.

float MPTK_ReleaseTimeMod = 1f

Multiplier to increase or decrease the default release time defined in the SoundFont. Recommended values between 0.1 and 2. Default is 1 (no modification of the release time)

float MPTK_SFChorusAmplify

[MPTK PRO] - Chorus level is defined in the SoundFont for each notes. This parameter increase or decrease the default SoundFont value.

float MPTK_SFFilterFreqOffset = 0f

[MPTK PRO] - Frequency cutoff is defined in the SoundFont for each notes. This parameter increase or decrease the default SoundFont value.

float MPTK_SFReverbAmplify

[MPTK PRO] - Reverberation level is defined in the SoundFont for each notes. This parameter increase or decrease the default SoundFont value.

int MPTK_StatVoiceCountActive

Count of the active voices (playing) - Readonly

int MPTK_StatVoiceCountFree

Count of the free voices for reusing on need. Older than AutoCleanVoiceTime are removed when count is over than AutoCleanVoiceLimit - Readonly

int MPTK_StatVoicePlayed

Count of voice played since the start of the synth

float MPTK_StatVoiceRatioReused

Percentage of voice reused during the synth life. 0: any reuse, 100:all voice reused (unattainable, of course!)

bool MPTK_WeakDevice

Should play on a weak device (cheaper smartphone) ? Apply only with AudioSource mode (MPTK_CorePlayer=False) Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.

EventSynthClass OnEventSynthAwake

Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.

```
...
if (!midiStreamPlayer.OnEventSynthAwake.HasEvent())
    midiStreamPlayer.OnEventSynthAwake.AddListener(StartLoadingSynth);
...
public void StartLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loading", name);
}
!
```

EventSynthClass OnEventSynthStarted

Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.

```
...
if (!midiStreamPlayer.OnEventStartSynth.HasEvent())
    midiStreamPlayer.OnEventStartSynth.AddListener(EndLoadingSynth);
...
public void EndLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loaded", name);
    midiStreamPlayer.MPTK_PlayEvent(
        new MPTKEvent() { Command = MPTKCommand.PatchChange, Value =
CurrentPatchInstrument, Channel = StreamChannel});
}
!
```

bool playOnlyFirstWave

Preset are often composed with 2 or more samples. Classically for left and right channel. Check this to play only the first sample found

[MidiFilePlayer](#) [] SpatialSynths [static]

Contains each [MidiSynth](#) for each channel when the prefab [MidiSpatializer](#) is used and `IsMidiChannelSpace=true`. Warning: only one [MidiSpatializer](#) can be used in a hierarchy.

Property Documentation

bool MPTK_ApplyUnityChorus [get], [set]

[MPTK PRO] - Apply Chorus Unity effect to the AudioSource. The effect is applied to all voices.

bool MPTK_ApplyUnityReverb [get], [set]

[MPTK PRO] - Apply Reverb Unity effect to the AudioSource. The effect is applied to all voices.

float MPTK_ChorusDelay [get], [set]

[MPTK PRO] - Chorus delay in ms. 0.1 to 100. Default = 40 ms.

float MPTK_ChorusDepth [get], [set]

[MPTK PRO] - Chorus modulation depth. 0 to 1. Default = 0.03.

float MPTK_ChorusDryMix [get], [set]

[MPTK PRO] - Volume of original signal to pass to output. 0 to 1. Default = 0.5.

float MPTK_ChorusRate [get], [set]

[MPTK PRO] - Chorus modulation rate in hz. 0 to 20. Default = 0.8 hz.

float MPTK_ChorusWetMix1 [get], [set]

[MPTK PRO] - Volume of 1st chorus tap. 0 to 1. Default = 0.5.

float MPTK_ChorusWetMix2 [get], [set]

[MPTK PRO] - Volume of 2nd chorus tap. This tap is 90 degrees out of phase of the first tap. 0 to 1. Default = 0.5.

float MPTK_ChorusWetMix3 [get], [set]

[MPTK PRO] - Volume of 3rd chorus tap. This tap is 90 degrees out of phase of the second tap. 0 to 1. Default = 0.5.

int MPTK_DedicatedChannel [get]

Dedicated Channel for this [MidiSynth](#) when the prefab [MidiSpatializer](#) is used. The [MidiSynth](#) reader (from a midi file) has no channel because no voice is played, so DedicatedChannel is set to -1

int MPTK_IndexSynthBuffSize [get], [set]

Set or Get sample rate output of the synth. -1:default, 0:24000, 1:36000, 2:48000, 3:60000, 4:72000, 5:84000, 6:96000. It's better to stop playing before changing on fly to avoid bad noise.

int MPTK_IndexSynthRate [get], [set]

Set or Get sample rate output of the synth. -1:default, 0:24000, 1:36000, 2:48000, 3:60000, 4:72000, 5:84000, 6:96000. It's better to stop playing before changing on fly to avoid bad noise.

float MPTK_MaxDistance [get], [set]

If MPTK_Spatialize is enabled, the volume of the audio source depends on the distance between the audio source and the listener. Beyond this distance, the volume is set to 0 and the midi player is paused. No effect if MPTK_Spatialize is disabled.

bool MPTK_PauseOnDistance [get], [set]

[obsolete] replaced by MPTK_Spatialize"); V2.83

float MPTK_ReverbDecayHFRatio [get], [set]

[MPTK PRO] - Decay HF Ratio : High-frequency to low-frequency decay time ratio. Ranges from 0.1 to 2.0.

float MPTK_ReverbDecayTime [get], [set]

[MPTK PRO] - Reverberation decay time at low-frequencies in seconds. Ranges from 0.1 to 20. Default is 1.

float MPTK_ReverbDelay [get], [set]

[MPTK PRO] - Late reverberation delay time relative to first reflection in seconds. Ranges from 0 to 0.1. Default is 0.04

float MPTK_ReverbDensity [get], [set]

[MPTK PRO] - Reverberation density (modal density) in percent. Ranges from 0 to 1.

float MPTK_ReverbDiffusion [get], [set]

[MPTK PRO] - Reverberation diffusion (echo density) in percent. Ranges from 0 to 1. Default is 1.

float MPTK_ReverbDryLevel [get], [set]

[MPTK PRO] - Mix level of dry signal in output. Ranges from 0 to 1.

float MPTK_ReverbHFReference [get], [set]

[MPTK PRO] - Reference high frequency in Hz. Ranges from 1000 to 20000. Default is 5000

float MPTK_ReverbLevel [get], [set]

[MPTK PRO] - Late reverberation level relative to room effect. Ranges from 0 to 1.

float MPTK_ReverbLFReference [get], [set]

[MPTK PRO] - Reference low-frequency in Hz. Ranges from 20 to 1000. Default is 250

float MPTK_ReverbReflectionDelay [get], [set]

[MPTK PRO] - Late reverberation level relative to room effect. Ranges from -10000.0 to 2000.0. Default is 0.0.

float MPTK_ReverbReflectionLevel [get], [set]

[MPTK PRO] - Early reflections level relative to room effect. Ranges from 0 to 1.

float MPTK_ReverbRoom [get], [set]

[MPTK PRO] - Room effect level at low frequencies. Ranges from 0 to 1.

float MPTK_ReverbRoomHF [get], [set]

[MPTK PRO] - Room effect high-frequency level. Ranges from 0 to 1.

float MPTK_ReverbRoomLF [get], [set]

[MPTK PRO] - Room effect low-frequency level. Ranges from 0 to 1.

float MPTK_SFFilterQModOffset [get], [set]

[MPTK PRO] - Quality Factor is defined in the SoundFont for each notes. This parameter increase or decrease the default SoundFont value.

bool MPTK_Spatialize [get], [set]

Should the Spatialization effect must be enabled? See here how to setup spatialization with Unity <https://paxstellar.fr/midi-file-player-detailed-view-2/#Foldout-Spatialization-Parameters>

int MPTK_SynthRate [get]

Current synth rate defined

int MPTK_Transpose [get], [set]

Transpose note from -24 to 24

float MPTK_Volume [get], [set]

Volume of midi playing. Must be ≥ 0 and ≤ 1

MPTKChordBuilder

[MPTK PRO] Chord builder class for MPTK. Usefull to generate Midi Music with [MidiStreamPlayer](#) - V2.82 See example in TestMidiStream.cs and ExtStreamPlayerPro.cs

Public Member Functions

- [MPTKChordBuilder](#) (bool log=false)
Create a default chord: tonic=C4, degree=1, count note=3.
- void [MPTK_BuildFromLib](#) (int pindex)
[MPTK PRO] Build a chord from the current chord in the lib ChordLib.csv in folder Resources/GeneratorTemplate.csv
- void [MPTK_BuildFromRange](#) ([MPTKRangeLib](#) range=null)
[MPTK PRO] Build a chord from the current selected range (MPTK_RangeSelected), Tonic and Degree are to be defined in parameter MPTKChord chord. Major range is selected if no range defined. After the call, Events contains all notes for the chord.

Public Attributes

- long [Arpeggio](#)
Delay in millisecond between each notes in the chord (play an arpeggio).
- int [Channel](#)
Midi channel fom 0 to 15 (9 for drum)
- int [Count](#)
Count of notes to compose the chord. Between 2 and 20.
- int [Degree](#)
Scale Degree. Between 1 and 7.
- long [Delay](#)
Delay in millisecond before playing the chord.
- long [Duration](#)
Duration of the chord in millisecond. Set -1 to play indefinitely.
- List< [MPTKEvent](#) > [Events](#)
List of midi events played for this chord. This list is build when call to MPTK_PlayChord or MPTK_PlayChordFromLib is done else null.
- int [FromLib](#)
Index of the chord in the libraries file ChordLib.csv in folder Resources/GeneratorTemplate.csv. To be used with MidiStreamPlayer.MPTK_PlayChordFromLib(MPTKChord chord)

- int [Tonic](#)
Tonic (Root) for the chord. 48=C4, ... , 60=C5, 61=C5#, 62=D5, ... , 72=C6,
- int [Velocity](#)
Velocity between 0 and 127

Detailed Description

[MPTK PRO] Chord builder class for MPTK. Usefull to generate Midi Music with [MidiStreamPlayer](#) - V2.82 See example in TestMidiStream.cs and ExtStreamPlayerPro.cs

Constructor & Destructor Documentation

[MPTKChordBuilder](#) (bool *log* = false)

Create a default chord: tonic=C4, degree=1, count note=3.

Parameters

<i>log</i>	True to display log
------------	---------------------

Member Function Documentation

void MPTK_BuildFromLib (int *pindex*)

[MPTK PRO] Build a chord from the current chord in the lib ChordLib.csv in folder Resources/GeneratorTemplate.csv

Parameters

<i>pindex</i>	position from 0 in ChordLib.csv
---------------	---------------------------------

void MPTK_BuildFromRange ([MPTKRangeLib](#) *range* = null)

[MPTK PRO] Build a chord from the current selected range (MPTK_RangeSelected), Tonic and Degree are to be defined in parameter MPTKChord chord. Major range is selected if no range defined. After the call, Events contains all notes for the chord.

Parameters

<i>range</i>	
--------------	--

Member Data Documentation

long Arpeggio

Delay in millisecond between each notes in the chord (play an arpeggio).

int Channel

Midi channel fom 0 to 15 (9 for drum)

int Count

Count of notes to compose the chord. Between 2 and 20.

int Degree

Scale Degree. Between 1 and 7.

- I Tonic First
- II Supertonic Second
- III Mediant Maj or min Third
- IV Subdominant Fourth
- V Dominant Fifth
- VI Submediant Maj or min Sixth
- VII Leading Tone/Subtonic Maj or min Seventh Good reading here:
https://lotusmusic.com/lm_chordnames.html

long Delay

Delay in millisecond before playing the chord.

long Duration

Duration of the chord in millisecond. Set -1 to play indefinitely.

List<[MPTKEvent](#)> Events

List of midi events played for this chord. This list is build when call to MPTK_PlayChord or MPTK_PlayChordFromLib is done else null.

int FromLib

Index of the chord in the libraries file ChordLib.csv in folder Resources/GeneratorTemplate.csv. To be used with `MidiStreamPlayer.MPTK_PlayChordFromLib(MPTKChord chord)`

int Tonic

Tonic (Root) for the chord. 48=C4, ... , 60=C5, 61=C5#, 62=D5, ... , 72=C6,

int Velocity

Velocity between 0 and 127

MPTKChordLib

[MPTK PRO] - Load library of chord from ChordLib.csv in folder Resources/GeneratorTemplate.csv - V2.82 new

Public Attributes

- int [Count](#)
Count of notes in the chord
- int [Index](#)
Position in the list
- string [Modifier3](#)
Some indicator when available.
- string [Modifier7](#)
- string [Name](#)
Long name of the scale

Properties

- static int [ChordCount](#) [get]
Count of chords availables
- static List< [MPTKChordLib](#) > [Chords](#) [get]
List of chords availables.
- int [this\[int index\]](#) [get]

Delta in 1/2 ton from the tonic, so first index=0 return 0 regardless the chord selected.

Detailed Description

[MPTK PRO] - Load library of chord from ChordLib.csv in folder Resources/GeneratorTemplate.csv - V2.82 new

Member Data Documentation

int Count

Count of notes in the chord

int Index

Position in the list

string Modifier3

Some indicator when available.

- M = major
- m = minor
- A = augmented
- D = diminished
- S = Suspended
- empty = undetermined

string Modifier7

Chord contains a 7iem

- 7 = major
- empty = undetermined

string Name

Long name of the scale

Property Documentation

int ChordCount [static], [get]

Count of chords availables

List<[MPTKChordLib](#)> Chords [static], [get]

List of chords availables.

int this[int index] [get]

Delta in 1/2 ton from the tonic, so first index=0 return 0 regardless the chord selected.

Parameters

<i>index</i>	Position in the scale. If exceed count of notes in the scale, the delta in 1/2 tons is taken from the next octave.
--------------	--

Returns

Delta in 1/2 ton from the tonic

MPTKEvent

Midi Event class for MPTK. Use this class to generate Midi Music with [MidiStreamPlayer](#) or to read midi events from a Midi file with [MidiLoad](#) or to receive midi events from [MidiFilePlayer](#) OnEventNotesMidi. With this class, you can: play and stop a note, change instrument (preset, patch, ...), change some control as modulation
Inherits ICloneable.

Public Types

- enum [EnumLength](#)
Note length as https://en.wikipedia.org/wiki/Note_value

Public Member Functions

- [MPTKEvent](#) (ulong data)
Create a MPTK Midi event from a midi input message
- ulong [ToData](#) ()
Build a packet midi message from a [MPTKEvent](#). Example: 0x00403C90 for a noton (90h, 3Ch note, 40h volume)
- override string [ToString](#) ()
Build a string description of the Midi event. V2.83 removes on each returns string

Public Attributes

- int [Channel](#)

Midi channel fom 0 to 15 (9 for drum)

- [MPTKCommand Command](#)
Midi Command code. Defined the type of message (Note On, Control Change, Patch Change...)
- [MPTKController Controller](#)
Controller code. When the Command is ControlChange, contains the code fo the controller to change (Modulation, Pan, Bank Select ...). Value will contains the value of the controller.
- long [Delay](#)
Delay before playing the note in millisecond. New with V2.82, works only in Core mode.
- long [Duration](#)
Duration of the note in millisecond. Set -1 to play indefinitely.
- int [IdSession](#)
V2.84 Define an Id associated with this event. Can be used with MPTK_ClearAllSound to clear only a subset of sound associated with this session.
- string [Info](#)
Information hold by textual meta event when Command=MetaEvent
- int [Length](#)
Duration of the note in Midi Tick. [MidiFilePlayer.MPTK_NoteLength](#) can be used to convert this duration. Not used for [MidiStreamPlayer](#), length is set only when reading a Midi file. https://en.wikipedia.org/wiki/Note_value
- [MPTKMeta Meta](#)
MetaEvent Code. When the Command is MetaEvent, contains the code of the meta event (Lyric, TimeSignature, ...). . Info will contains the value of the meta.
- uint [Source](#)
Origine of the message. Midi ID if from Midi Input else zero. V2.83: rename source to Source et set public.
- long [Tick](#)
Time in Midi Tick (part of a Beat) of the Event since the start of playing the midi file. This time is independent of the Tempo or Speed. Not used for [MidiStreamPlayer](#).
- long [Track](#)
Index of track.
- int [Value](#)
Contains a value between 0 and 127 in relation with the Command. For:
- int [Velocity](#)
Velocity between 0 and 127

- List< fluid_voice > [Voices](#)
List of voices associated to this Event for playing a NoteOn event.

Detailed Description

Midi Event class for MPTK. Use this class to generate Midi Music with [MidiStreamPlayer](#) or to read midi events from a Midi file with [MidiLoad](#) or to receive midi events from [MidiFilePlayer](#) OnEventNotesMidi. With this class, you can: play and stop a note, change instrument (preset, patch, ...), change some control as modulation

```
!  
! // Change instrument to Marimba for channel 0  
! NotePlaying = new MPTKEvent() {  
!     Command = MPTKCommand.NoteOn,  
!     Value = 12, // generally Marimba but depend on the SoundFont selected  
!     Channel = 0 }; // Instrument are defined by channel. So at any time, only 16  
différents instruments can be used simultaneously.  
! midiStreamPlayer.MPTK_PlayEvent(NotePlaying);  
!  
! // Play a C5 during one second with the Marimba instrument  
! NotePlaying = new MPTKEvent() {  
!     Command = MPTKCommand.NoteOn,  
!     Value = 60, // play a C5 note  
!     Channel = 0,  
!     Duration = 1000, // one second  
!     Velocity = 100 };  
! midiStreamPlayer.MPTK_PlayEvent(NotePlaying);  
!
```

Member Enumeration Documentation

enum [EnumLength](#) [strong]

Note length as https://en.wikipedia.org/wiki/Note_value

Constructor & Destructor Documentation

[MPTKEvent](#) (ulong *data*)

Create a MPTK Midi event from a midi input message

Parameters

<i>data</i>	
-------------	--

Member Function Documentation

ulong ToData ()

Build a packet midi message from a [MPTKEvent](#). Example: 0x00403C90 for a noton (90h, 3Ch note, 40h volume)

Returns

override string ToString ()

Build a string description of the Midi event. V2.83 removes
on each returns string

Returns

Member Data Documentation

int Channel

Midi channel fom 0 to 15 (9 for drum)

[MPTKCommand](#) Command

Midi Command code. Defined the type of message (Note On, Control Change, Patch Change...)

[MPTKController](#) Controller

Controller code. When the Command is ControlChange, contains the code fo the controller to change (Modulation, Pan, Bank Select ...). Value will contains the value of the controller.

long Delay

Delay before playing the note in millisecond. New with V2.82, works only in Core mode.

long Duration

Duration of the note in millisecond. Set -1 to play indefinitely.

int IdSession

V2.84 Define an Id associated with this event. Can be used with MPTK_ClearAllSound to clear only a subset of sound associated with this session.

string Info

Information hold by textual meta event when Command=MetaEvent

int Length

Duration of the note in Midi Tick. [MidiFilePlayer.MPTK_NoteLength](#) can be used to convert this duration. Not used for [MidiStreamPlayer](#), length is set only when reading a Midi file. https://en.wikipedia.org/wiki/Note_value

MPTKMeta Meta

MetaEvent Code. When the Command is MetaEvent, contains the code of the meta event (Lyric, TimeSignature, ...). . Info will contains the value of the meta.

uint Source

Origine of the message. Midi ID if from Midi Input else zero. V2.83: rename source to Source et set public.

long Tick

Time in Midi Tick (part of a Beat) of the Event since the start of playing the midi file. This time is independent of the Tempo or Speed. Not used for [MidiStreamPlayer](#).

long Track

Index of track.

int Value

Contains a value between 0 and 127 in relation with the Command. For:

- If Command = NoteOn then Value contains midi note. 60=C5, 61=C5#, ..., 72=C6,
- If Command = ControlChange then Value contains controller value, see MPTKController
- If Command = PatchChange then Value contains patch/preset/instrument value. See the current SoundFont to find value associated to each instrument.

int Velocity

Velocity between 0 and 127

List<fluid_voice> Voices

List of voices associated to this Event for playing a NoteOn event.

MPTKRangeLib

[MPTK PRO] - Load library of scale from GammeDefinition.csv in folder Resources/GeneratorTemplate.csv - V2.82 new

Static Public Member Functions

- static [MPTKRangeLib Range](#) (int index, bool log=false)
Get a scale from an index. Scales are read from GammeDefinition.csv in folder Resources/GeneratorTemplate.csv.

Public Attributes

- int [Count](#)
Count of notes in the range
- string [Flag](#)
Some indicator when available.
- int [Index](#)
Position in the list (from the library)
- bool [Main](#)
Common scale if true else exotic
- string [Name](#)
Long name of the scale
- string [Short](#)
Short name of the scale

Properties

- static int [RangeCount](#) [get]
Count of scales availables in the library GammeDefinition.csv in folder Resources/GeneratorTemplate.csv

- int [this\[int index\]](#) [get]
Delta in 1/2 ton from the tonic, so first position (index=0) always return 0 regardless the range selected.

Detailed Description

[MPTK PRO] - Load library of scale from GammeDefinition.csv in folder Resources/GeneratorTemplate.csv - V2.82 new

Member Function Documentation

static [MPTKRangeLib](#) Range (int *index*, bool *log = false*) [static]

Get a scale from an index. Scales are read from GammeDefinition.csv in folder Resources/GeneratorTemplate.csv.

Parameters

<i>index</i>	
<i>log</i>	

Returns

Member Data Documentation

int Count

Count of notes in the range

string Flag

Some indicator when available.

- M = major scale
- m = minor scale
- _ = undetermined

int Index

Position in the list (from the library)

bool Main

Common scale if true else exotic

string Name

Long name of the scale

string Short

Short name of the scale

Property Documentation

int RangeCount [static], [get]

Count of scales availables in the library GammeDefinition.csv in folder Resources/GeneratorTemplate.csv

int this[int index] [get]

Delta in 1/2 ton from the tonic, so first position (index=0) always return 0 regardless the range selected.

Parameters

<i>index</i>	Position in the scale. If greater than count of notes in the scale, the delta in 1/2 tons is taken from the next octave.
--------------	--

Returns

Delta in 1/2 ton from the tonic

Index

INDEX