最新のドキュメントはGitHubにあります。
質問や問題があればGitHubのissueまで ご投稿ください。
https://github.com/kshoji/Unity-MIDI-Plugin-supports

プラグインのインストール方法、機能について説明しています。

目次

MIDI Plugin for Mobile and Desktop

このプラグインはモバイルアプリ(iOS, Android)、デスクトップアプリ(Windows, OSX, Linux)、WebGLアプリにMIDI送受信の機能を追加します。
現在は`「MIDI 1.0プロトコル」`のみ実装されています。

プラットフォームで利用可能なMIDIインタフェース

各プラットフォームで対応しているMIDIインタフェースは下記の通りです。

Platform	Bluetooth MIDI	USB MIDI	Network MIDI (RTP-MIDI)
iOS	○	○	○
Android	○	○	△(試験的に対応)
Universal Windows Platform	-	○	△(試験的に対応)
Standalone OSX, Unity Editor OSX	○	○	○
Standalone Linux, Unity Editor Linux	○	○	△(試験的に対応)
Standalone Windows, Unity Editor Windows	-	○	△(試験的に対応)
WebGL	○	○	-

制限事項

Android

- USB MIDI は API Level 12 (Android 3.1) 以上で利用できます。
- Bluetooth MIDI は API Level 18 (Android 4.3) 以上で利用できます。

iOS / OSX

- iOS 11.0 以上で動作します。
- Bluetooth MIDIはCentralモードのみサポートします。

UWP

- UWPのバージョン 10.0.10240.0 以上で動作します。
- Bluetooth MIDIはサポートしていません。
- Network MIDI(RTP-MIDI) 機能を使う場合は、`Project Settings > Player > Capabilities` の設定にある `PrivateNetworkClientServer` を有効にしてください。

Windows

- Bluetooth MIDIはサポートしていません。

WebGL

- サポートされるMIDIデバイスはOSやブラウザに依存します。
- WebGL はUnityWebRequestを使って他のサーバーのリソースにアクセスできない場合があるので、SMFなどのリソースファイルを `StreamingAssets` に置いてください。
- `WebGLTemplates` ディレクトリの `index.html` ファイルを下記のように変更する 必要があります。 `unityInstance` 変数を経由してUnityのランタイムにアクセスできようにしています。
  - もしくは、 `MIDI/Samples/WebGLTemplates` のファイルを `Assets/WebGLTemplates` にコピーし、 `Project Settings > Player > Resolution and Presentation > WebGL Template` の設定から、 `Default-MIDI` か `Minimal-MIDI` のテンプレートを選択します。
  - 詳しくは、[Unity公式のWebGL Templatesドキュメント]() を参照してください。

オリジナルの抜粋:

```
      script.onload = () => {
        createUnityInstance(canvas, config, (progress) => {
          progressBarFull.style.width = 100 * progress + "%";
        }).then((unityInstance) => {
```

修正後: グローバル変数 `unityInstance` を追加

```
      var unityInstance = null; // <- HERE
      script.onload = () => {
        createUnityInstance(canvas, config, (progress) => {
          progressBarFull.style.width = 100 * progress + "%";
        }).then((unityInst) => { // <- HERE
          unityInstance = unityInst; // <- HERE
```

Network MIDI

- エラー訂正(RTP MIDIジャーナリング)プロトコルは上記に示された「試験的」のプラットフォームでは実装されていません。

プラグインのインストール

1. アセットストアViewからunitypackageをインポートします。
2. プラットフォーム(iOSやAndroid)を選択して、サンプルアプリをビルドします。
    - サンプルシーンは Assets/MIDI/Samples ディレクトリにあります。

ビルドの後処理(PostProcessing)について

PostProcessing: iOS

- ビルドの後処理によって、追加のフレームワークが自動的に追加されます。
    - 追加のフレームワーク: `CoreMIDI framework`, `CoreAudioKit.framework`
- `Info.plist` が自動的に調整されます。
    - 追加のプロパティ: `NSBluetoothAlwaysUsageDescription`

PostProcessing: Android

- ビルドの後処理によって、 `AndroidManifest.xml` が自動的に調整されます。
    - 追加のパーミッション: `android.permission.BLUETOOTH`, `android.permission.BLUETOOTH_ADMIN`, `android.permission.ACCESS_FINE_LOCATION`, `android.permission.BLUETOOTH_SCAN`, `android.permission.BLUETOOTH_CONNECT`, `android.permission.BLUETOOTH_ADVERTISE`.
    - 追加のfeature: `android.hardware.bluetooth_le`, `android.hardware.usb.host`
- Oculus Quest 2 でUSB MIDI機能を使いたい場合、接続を検知するために下記のコードをコメントアウト解除する必要があります。

`PostProcessBuild.cs` の一部

```
    public class ModifyAndroidManifest : IPostGenerateGradleAndroidProject
    {
        public void OnPostGenerateGradleAndroidProject(string basePath)
        {
            :

            // androidManifest.AddUsbIntentFilterForOculusDevices(); // Oculus Quest 2のためには、この行をコメント解除する
```

機能の実装方法

プラグインの初期化

1. MonoBehaviourの `Awake` メソッドから `MidiManager.Instance.InitializeMidi` メソッドを呼び出します。
    - `MidiManager` という名前の GameObject が、ヒエラルキービューの `DontDestroyOnLoad` の下に自動的に作成されます。

    > NOTE: EventSystem コンポーネントが既に他の場所に存在する場合には、`gameObject.AddComponent<EventSystem>()` メソッドの呼出を `MidiManager.Instance.InitializeMidi` メソッドから削除してください。

2. (BLE MIDI のみ)
    - `MidiManager.Instance.StartScanBluetoothMidiDevices` メソッドを呼び出して、周囲 の BLE MIDI デバイスを探します。
        - このメソッドは `InitializeMidi` メソッドのコールバックAction内から呼ばれるべきです。
3. (RTP-MIDI のみ)
    - `MidiManager.Instance.StartRtpMidi` メソッドに セッション名 と udpポート番号 を指定して呼び出すと、RTP-MIDI セッションの受け付けが開始されます。

```
private void Awake()
{
    MidiManager.Instance.RegisterEventHandleObject(gameObject);
    MidiManager.Instance.InitializeMidi(() =>
    {
        MidiManager.Instance.StartScanBluetoothMidiDevices(0);
    });
}
```

図1 Awake メソッドはこのようになります

プラグインの終了

1. MonoBehaviour の `OnDestroy` メソッド内で `MidiManager.Instance.TerminateMidi` を呼び出します。
    - このメソッドはMIDI機能を使い終わって、シーンが終了する際に呼ばれるべきです。
2. (RTP-MIDI のみ)
    - `MidiManager.Instance.StopRtpMidi` メソッドを呼び出して、RTP-MIDIセッションの通信を終了します。

```
private void OnDestroy()
{
    MidiManager.Instance.TerminateMidi();
}
```

図2 MidiManager 終了時の記述

RTP-MIDI を使う (iOS以外のプラットフォームでの試験的な機能)

- RTP-MIDIセッションの開始:
    - `MidiManager.Instance.StartRtpMidi` メソッドに セッション名 と udpポート番号 ををを指定して呼び出すと、RTP-MIDI セッションの受け付けが開始されます。
    - これにより 指定したポート番号でUDPポートの受信が開始され、他のコンピュータからアプリに接続できるようになります。
- RTP-MIDI セッションの停止:
    - `MidiManager.Instance.StopRtpMidi` メソッドを呼び出すと、RTP-MIDIセッションの通信が終了します。
- RTP-MIDIが実行されている他のコンピュータに接続:
    - `MidiManager.Instance.ConnectToRtpMidiClient` メソッドを呼び出すと、他のコン ピュータとの接続を行います。

```
// UDP 5004 ポートで、"RtpMidiSession" というセッション名で受け付けを開始
MidiManager.Instance.StartRtpMidi("RtpMidiSession", 5004);
...
// セッションの停止
MidiManager.Instance.StartRtpMidi(5004);

// RTP-MIDIが実行されている他のコンピュータに接続
MidiManager.Instance.ConnectToRtpMidiClient("RtpMidiSession", 5004, new
IPEndPoint(IPAddress.Parse("192.168.0.111"), 5004));
```

図3 RTP-MIDI 機能の利用

MIDIデバイスの接続・切断のイベントハンドリング

1. `MidiManager.Instance.RegisterEventHandleObject` メソッドを用いてイベントを受信するためのGameObjectを登録します。
2. イベントを受信するため、インタフェース `IMidiDeviceEventHandler` を実装します。
    - 新しいMIDIデバイスが接続された際には、 `OnMidiInputDeviceAttached`, `OnMidiOutputDeviceAttached` が呼ばれます。
    - MIDIデバイスが接続解除された際には、 `OnMidiInputDeviceDetached`, `OnMidiOutputDeviceDetached` が呼ばれます。

```
public void OnMidiInputDeviceAttached(string deviceId)
{
}

public void OnMidiOutputDeviceAttached(string deviceId)
{
    receivedMidiMessages.Add($"MIDI device attached. deviceId: {deviceId}, name:
{MidiManager.Instance.GetDeviceName(deviceId)}");
}

public void OnMidiInputDeviceDetached(string deviceId)
{
}

public void OnMidiOutputDeviceDetached(string deviceId)
{
    receivedMidiMessages.Add($"MIDI device detached. deviceId: {deviceId}, name:
{MidiManager.Instance.GetDeviceName(deviceId)}");
}
```

図4 デバイスの接続・切断イベント処理

コードの全容は `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` ファイルにあります。

MIDIイベントの受信

1. IMidiEventHandler.cs に定義されている受信インタフェースを実装します。実装するクラス名 は→のようなものです `IMidiXXXXXEventHandler`.
    - Note On イベントを受信したい場合には、 `IMidiNoteOnEventHandler` インタフェースを実装します。
2. イベント受信をするGameObjectを登録するため、 `MidiManager.Instance.RegisterEventHandleObject` メソッドを呼び出します。
3. MIDIイベントが受信されたら、実装したメソッドが呼ばれます。

```
public class MidiSampleScene : MonoBehaviour, IMidiAllEventsHandler,
IMidiDeviceEventHandler
{
    private void Awake()
    {
        MidiManager.Instance.RegisterEventHandleObject(gameObject);
        ...
```

図5 IMidiAllEventHandler の実装の記述と、 Awake内で RegisterEventHandlerObject メソッドを呼び出す例です。

```
public void OnMidiNoteOn(string deviceId, int group, int channel, int note, int
velocity)
{
    receivedMidiMessages.Add($"OnMidiNoteOn channel: {channel}, note: {note},
velocity: {velocity}");
}

public void OnMidiNoteOff(string deviceId, int group, int channel, int note, int
velocity)
{
    receivedMidiMessages.Add($"OnMidiNoteOff channel: {channel}, note: {note},
velocity: {velocity}");
}
```

図6 MIDI Note On/ Note Offイベントを受信するハンドラ

コードの全容は `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` ファイルにあります。

MIDIイベントの送信

1. コードのどこかで `MidiManager.Instance.SendMidiXXXXXX` メソッドを呼び出します。 一例:

```
MidiManager.Instance.SendMidiNoteOn("deviceId", 0/*groupId*/, 0/*channel*/,
60/*note*/, 127/*velocity*/);
```

2. 指定する deviceId は `MidiManager.Instance.DeviceIdSet` プロパティから取得できま す。 (型は HashSet<string>です)

```
if (GUILayout.Button("NoteOn"))
{
    MidiManager.Instance.SendMidiNoteOn(deviceIds[deviceIdIndex], 0, (int)channel,
(int)noteNumber, (int)velocity);
}
```

図7 MIDI Note Onメッセージの送信

コードの全容は `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` ファイルにあります。

シーケンサーの作成と開始

```
var isSequencerOpened = false;
var sequencer = new SequencerImpl(() => { isSequencerOpened = true; });
sequencer.Open();
```

図8 SequencerImplのインスタンス生成と開始

コードの全容は `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` ファイルにあります。

SMFをシーケンスとして読み出し、再生する

```
sequencer.UpdateDeviceConnections();

using var stream = new FileStream(smfPath, FileMode.Open, FileAccess.Read);
sequencer.SetSequence(stream);
sequencer.Start();

...

sequencer.Stop();
```

図9 SMFを読んで再生する。

シーケンスを記録する

```
sequencer.UpdateDeviceConnections();

sequencer.SetSequence(new Sequence(Sequence.Ppq, 480));
sequencer.StartRecording();

...

sequencer.Stop();
```

図10 記録するためのシーケンスを設定し、MIDIデータの記録を開始する

シーケンスをSMFとして書き出す

```
var sequence = sequencer.GetSequence();
if (sequence.GetTickLength() > 0)
{
    using var stream = new FileStream(recordedSmfPath, FileMode.Create,
FileAccess.Write);
    MidiSystem.WriteSequence(sequence, stream);
}
```

図11 記録したシーケンスからSMFを書き出す

Android: BLE MIDIデバイスの検索にCompanionDeviceManagerを使う

AndroidのBLE MIDIデバイスの接続にCompanionDeviceManagerが使えます。

この機能を有効にするには、`Scripting Define Symbols`の設定に`FEATURE_ANDROID_COMPANION_DEVICE`を追加します。

```
Project Settings > Other Settings > Script Compilation > Scripting Define Symbols
```

テストしたデバイス

- Android: Pixel 4a, Oculus Quest2
- iOS: iPod touch 7th gen
- UWP/Standalone Windows/Unity Editor Windows: Surface Go 2
- Standalone OSX/Unity Editor OSX: Mac mini 3,1
- Standalone Linux/Unity Editor Linux: Ubuntu 20.04 on VirtualBox
- MIDI devices:
    - Quicco mi.1 (BLE MIDI)
    - Miselu C.24 (BLE MIDI)
    - TAHORNG Elefue (BLE MIDI)
    - Roland UM-ONE (USB MIDI)
        - NOTE: このデバイスはiOSでは動きませんでした。
    - Gakken NSX-39 (USB-MIDI)
    - MacOS Audio MIDI Setup (RTP-MIDI)

バージョン履歴

- v1.0 初期リリース
- v1.1 更新リリース
    - 追加: MIDI シーケンサ(MIDIシーケンスの再生・録音)
    - 追加: SMFの読み取り・書き出し
    - 追加: AndroidでのBLE MIDI Peripheral機能
    - 修正: AndroidでのUSB MIDI受信時の問題
    - 修正: Android・iOSでのBLE MIDI送信時の問題
    - 修正: AndroidでのBLE MIDI送信(velocity = 0でのNoteOn)の問題
- v1.2.0 更新リリース
    - 追加: Androidほかプラットフォーム向けの、試験的な RTP-MIDIサポート
    - 追加: Universal Windows Platform(UWP)向けのUSB MIDIサポート
    - 追加: Android 12の新しいBluetoothパーミッションのサポート
    - 修正: iOS, AndroidでのMIDI送受信のパフォーマンス向上
    - 修正: シーンが複数回追加された際にEventSystemが複数個作成されるエラー
    - 修正: AndroidのBLE MIDIでのタイムスタンプの問題
- v1.2.1 バグ修正
    - 修正: シーケンサーのスレッドが閉じたあとも残る問題
    - 修正: AndroidでのProgramChangeメッセージの受信が失敗する
    - 修正: サンプルシーンでのSystem exclusiveのログが正しく表示されない
    - 修正: UWPでThreadInterruptExceptionが発生する
    - 修正: SMFでSystem exclusiveを読み書きすると起きる問題
    - 修正: いくつかのパフォーマンス改善
- v1.3.0 更新リリース
    - 追加: Standalone OSX, Windows, Linuxプラットフォーム対応
    - 追加: WebGLプラットフォーム対応
    - 追加: Unity Editor OSX, Windows, Linux対応
    - 変更: シーケンサーの実装を Thread から Coroutine に
    - 修正: iOS/OSX でのデバイス接続・切断時の問題
- v1.3.1 バグ修正
    - Issue connecting to Quest 2 via cable
    - Sample scene stops working.
    - Byte is obsolete on android
    - Any way of negotiating MTU?
    - Can't get it to work on iOS
    - Have errors with sample scene
    - Androidのパーミッション要求についての問題を解消
    - AndroidのCompanionDeviceManager経由での接続をサポート
- v1.3.2 バグ修正
    - Androidのコンパイルエラーを修正
    - SMF再生時のMIDIイベントの順序が正しくなるよう修正

連絡先

GitHubでの不具合報告、サポート

- GitHubのサポートリポジトリ https://github.com/kshoji/Unity-MIDI-Plugin-supports
    - 問題の検索と報告: https://github.com/kshoji/Unity-MIDI-Plugin-supports/issues

プラグイン作者

- Kaoru Shoji/庄司 薫 : 0x0badc0de@gmail.com
- github: https://github.com/kshoji

使用した自作のオープンソースソフトウェア

- Android Bluetooth MIDI library: https://github.com/kshoji/BLE-MIDI-for-Android
- Android USB MIDI library: https://github.com/kshoji/USB-MIDI-Driver
- iOS MIDI library: https://github.com/kshoji/Unity-MIDI-Plugin-iOS
- MidiSystem for .NET(sequencer, SMF importer/exporter): https://github.com/kshoji/MidiSystem-for-.NET
- RTP-MIDI for .NET: https://github.com/kshoji/RTP-MIDI-for-.NET
- Unity MIDI Plugin UWP: https://github.com/kshoji/Unity-MIDI-Plugin-UWP
- Unity MIDI Plugin Linux: https://github.com/kshoji/Unity-MIDI-Plugin-Linux
- Unity MIDI Plugin OSX: https://github.com/kshoji/Unity-MIDI-Plugin-OSX

他者提供による、使用したサンプルMIDIデータ

UnityWebRequest's URLとして指定しています。SMFのバイナリデータ自体はパッケージには含まれていません。

- Prelude and Fugue in C minor BWV 847 Music by J.S. Bach
    - The MIDI, audio(MP3, OGG) and video files of Bernd Krueger are licensed under the cc-by-sa Germany License.
    - This means, that you can use and adapt the files, as long as you attribute to the copyright holder
    - Name: Bernd Krueger
    - Source: http://www.piano-midi.de
    - The distribution or public playback of the files is only allowed under identical license conditions.