スタートガイド (インストール、初期化、送受信)

インストール

1. Asset Store から `.unitypackage` をインポートします。
    - 古いバージョンからアップデートした場合は、以下のディレクトリを確認してください：`Assets/MIDI/Plugins/Android` 古い、または重複している `.aar` ファイルがある場合は削除してください。
2. ターゲットプラットフォーム (iOS / Android / Standalone / WebGL / UWP) に切り替えます。
3. サンプルシーンをビルドして実行します：
    - `Assets/MIDI/Samples/Scenes/`

以前にオプションの依存関係（Nearby など）をインストールしていた場合は、それらを最新の状態に更新してください。

MIDI 1.0 クイックスタート (受信 + オプションの BLE スキャン)

推奨されるライフサイクル：

```
using UnityEngine;
using jp.kshoji.unity.midi;

public sealed class Midi1QuickStart : MonoBehaviour, IMidiDeviceEventHandler,
IMidiNoteOnEventHandler
{
    private void Awake()
    {
        // イベントハンドラオブジェクトを登録
        MidiManager.Instance.RegisterEventHandleObject(gameObject);

        // MIDI の初期化
        MidiManager.Instance.InitializeMidi(() =>
        {
#if (UNITY_ANDROID || UNITY_IOS || UNITY_WEBGL) && !UNITY_EDITOR
            // BLE MIDI のみ (サポートされている場合): 初期化完了後にスキャンを開始。
            MidiManager.Instance.StartScanBluetoothMidiDevices(0);
#endif
        });
    }

    private void OnDestroy()
    {
        // MIDI の終了
        MidiManager.Instance.TerminateMidi();
    }

    public void OnMidiInputDeviceAttached(string deviceId)
        => Debug.Log($"入力デバイス接続: {deviceId}");

    public void OnMidiInputDeviceDetached(string deviceId)
        => Debug.Log($"入力デバイス切断: {deviceId}");

    public void OnMidiOutputDeviceAttached(string deviceId)
        => Debug.Log($"出力デバイス接続: {deviceId}");

    public void OnMidiOutputDeviceDetached(string deviceId)
        => Debug.Log($"出力デバイス切断: {deviceId}");

    public void OnMidiNoteOn(string deviceId, int group, int channel, int note,
int velocity)
        => Debug.Log($"NoteOn デバイス:{deviceId} ch:{channel} note:{note} vel:
{velocity}");
}
```

MIDI 1.0 送信

```
// Note On を送信
MidiManager.Instance.SendMidiNoteOn(
    "deviceId",
    0 /*group*/,
    0 /*channel*/,
    60 /*note*/,
    127 /*velocity*/
);
```

出力デバイスの ID は以下から取得できます：

- `MidiManager.Instance.OutputDeviceIdSet` (type: `HashSet<string>`)

MIDI 2.0 クイックスタート

```
using UnityEngine;
using jp.kshoji.unity.midi;

public sealed class Midi2QuickStart : MonoBehaviour, IMidi2DeviceEventHandler,
IMidi2NoteOnEventHandler
{
    private void Awake()
    {
        MidiManager.Instance.RegisterEventHandleObject(gameObject);
        MidiManager.Instance.InitializeMidi2(() => { });
    }

    private void OnDestroy()
    {
        MidiManager.Instance.TerminateMidi2();
    }

    public void OnMidi2InputDeviceAttached(string deviceId)
        => Debug.Log($"MIDI2 入力接続: {deviceId}");

    public void OnMidi2InputDeviceDetached(string deviceId)
        => Debug.Log($"MIDI2 入力切断: {deviceId}");

    public void OnMidi2OutputDeviceAttached(string deviceId)
        => Debug.Log($"MIDI2 出力接続: {deviceId}");

    public void OnMidi2OutputDeviceDetached(string deviceId)
        => Debug.Log($"MIDI2 出力切断: {deviceId}");

    public void OnMidi2NoteOn(string deviceId, int group, int channel, int note,
int velocity, int attributeType, int attributeData)
        => Debug.Log($"MIDI2 NoteOn デバイス:{deviceId} g:{group} ch:{channel} note:
{note} vel:{velocity}");
}
```

次に読むべきドキュメント

- プラットフォームと制限事項
- ビルド後の処理 (PostProcessing) とスクリプト定義シンボル
- トランスポートとプラットフォームに関する注意点
- サンプル