

Table of Contents

概要	2
インストールとビルド設定	4
使い方	12
API References	
Limosa	24
BondedScope	26
BondedScope.ScopeState	28
ButtonState	30
ButtonStateChanged	34
ConvertedData	36
IntegerChanged	39
LimosaProtocol	41
RawData	52
ScopeAddress	55
ScopeButton	57
ScopeInfo	61
ScopeOffset	64
ServiceBindState	67
Vector4f	69
サンプルデモ	71

概要

Atmoph Window Yo SDK for Unity は、Atmoph Window Yoで動作するUnityアプリを作成するための Software Development Kit です。Atmoph Window YoはAndroidベースのOSで動作していますので、Android用アプリが動作します。SDKではAtmoph Window Yo特有の機能をアプリでサポートするためのフレームワークを提供します。

機能一覧

SDK経由でサポートが可能になる主な機能は次のものになります。

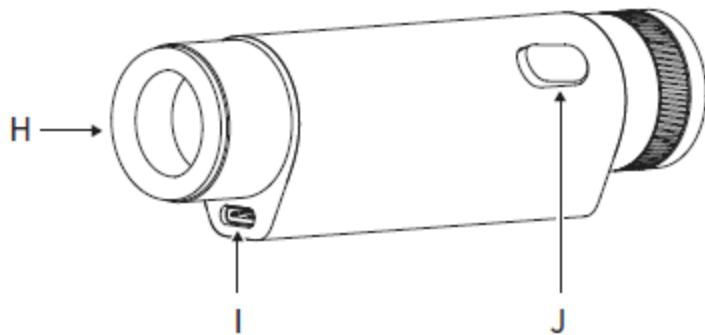
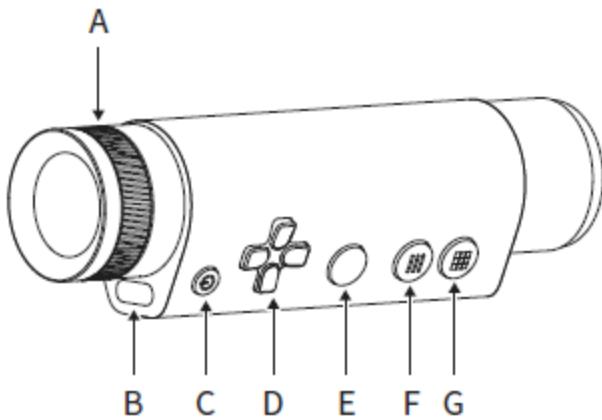
Scopeをゲームコントローラとして使用

Yoの望遠鏡型コントローラ Scope の次の入出力をUnityアプリで扱えるようになります。

1. [Scopeのボタン操作の検知](#)
 2. [Scopeのズームリング操作の検知](#)
 3. [Scopeのポインティング位置、向き、距離センター値の取得](#)
 4. [Scopeの動的情報の取得（接続状態、アドレス、バッテリー残量）](#)
 5. [Scopeの静的情報の取得\(本体型番、シリアルナンバー、FWバージョン、PnPID\)](#)
 6. [Scopeへの振動指示](#)
7. [Unityアプリの終了を本体に伝える](#)

Scopeの各部の名称

List of Components / 各部のなまえ



- | | |
|------------------------|----------------|
| A. Zoom Ring | A. ズームリング |
| B. Distance Sensor | B. 距離センサー |
| C. Power Button | C. 電源ボタン |
| D. D-Pad | D. 方向ボタン |
| E. Enter Button | E. エンターボタン |
| F. Menu Button | F. メニュー ボタン |
| G. Views Button | G. 風景ボタン |
| H. Finder | H. ファインダー |
| I. USB-C Charging Port | I. USB-C 充電ポート |
| J. Trigger Button | J. トリガーボタン |

"インストールとビルド設定"の目次

- [インストール手順](#)
- [SDKの内容](#)
- [ビルド設定](#)

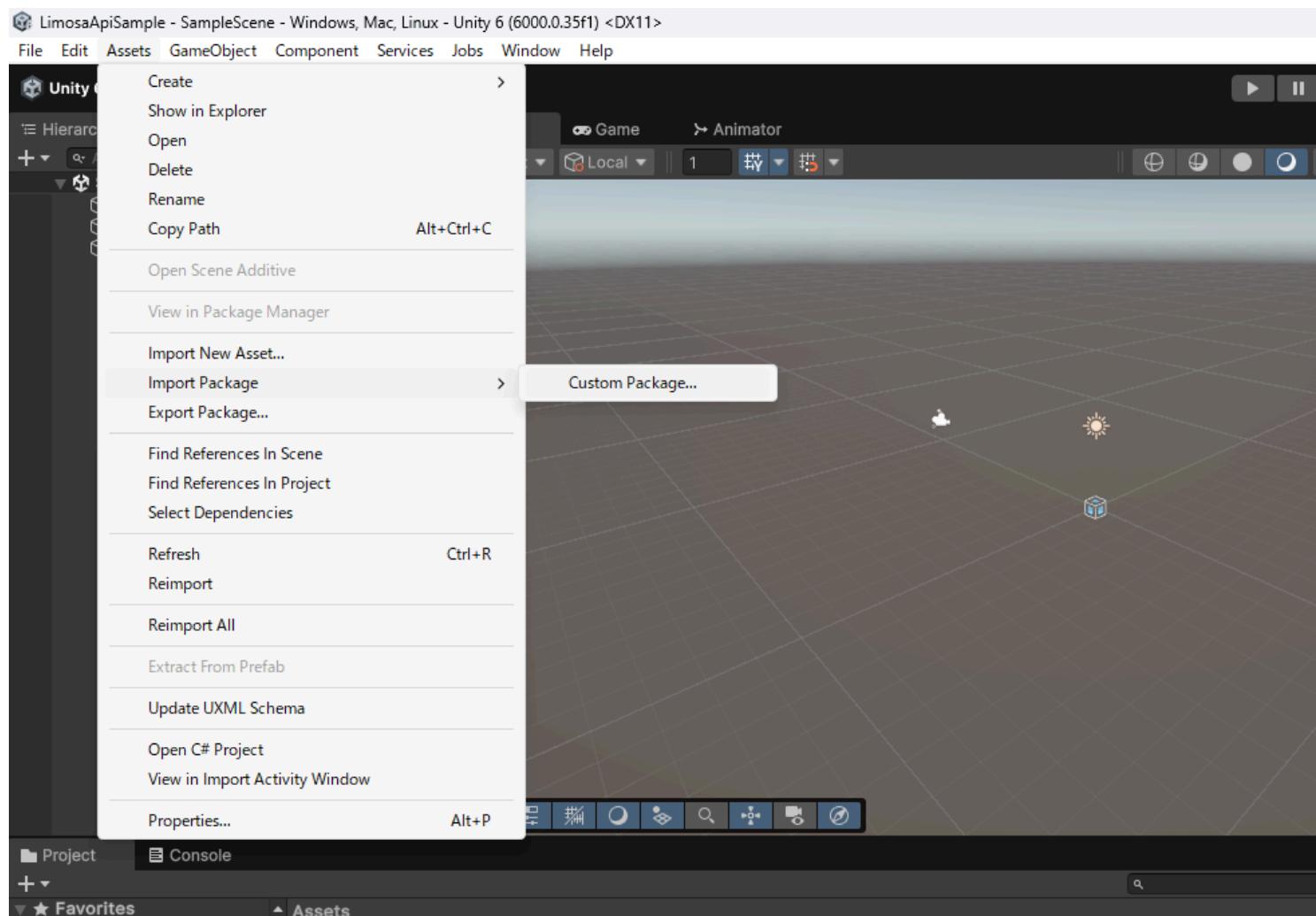
SDKのインストールは、Unityのプロジェクトに"LimosaAPI.unitypackage"をインポートすることで行います。

※ 各図はUnity6(6000.0.35f1.7976.6927)のものです。

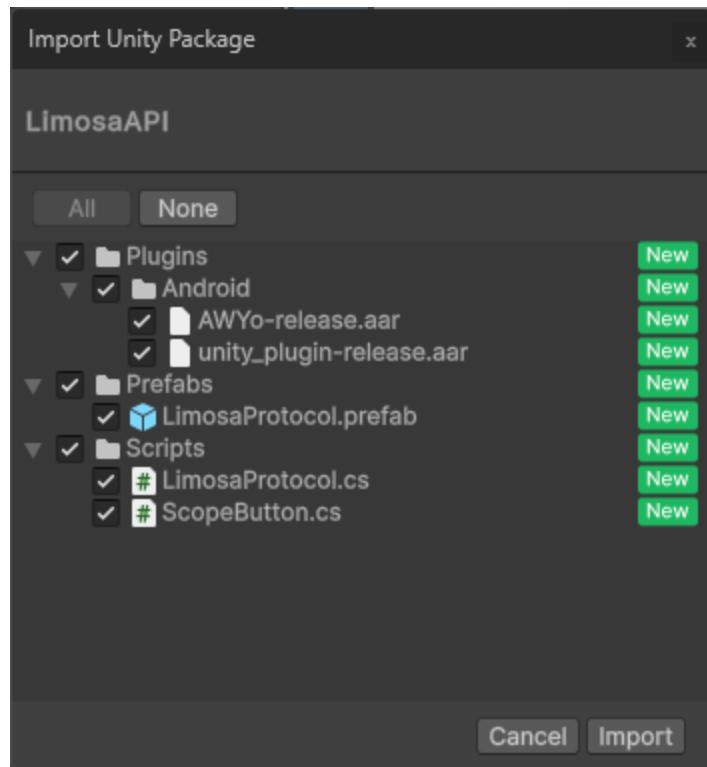
インストール手順

1. Unityのメニューバーの次のボタンから"LimosaAPI.unitypackage"をインポートする

Assets -> Import Package -> Custom Package



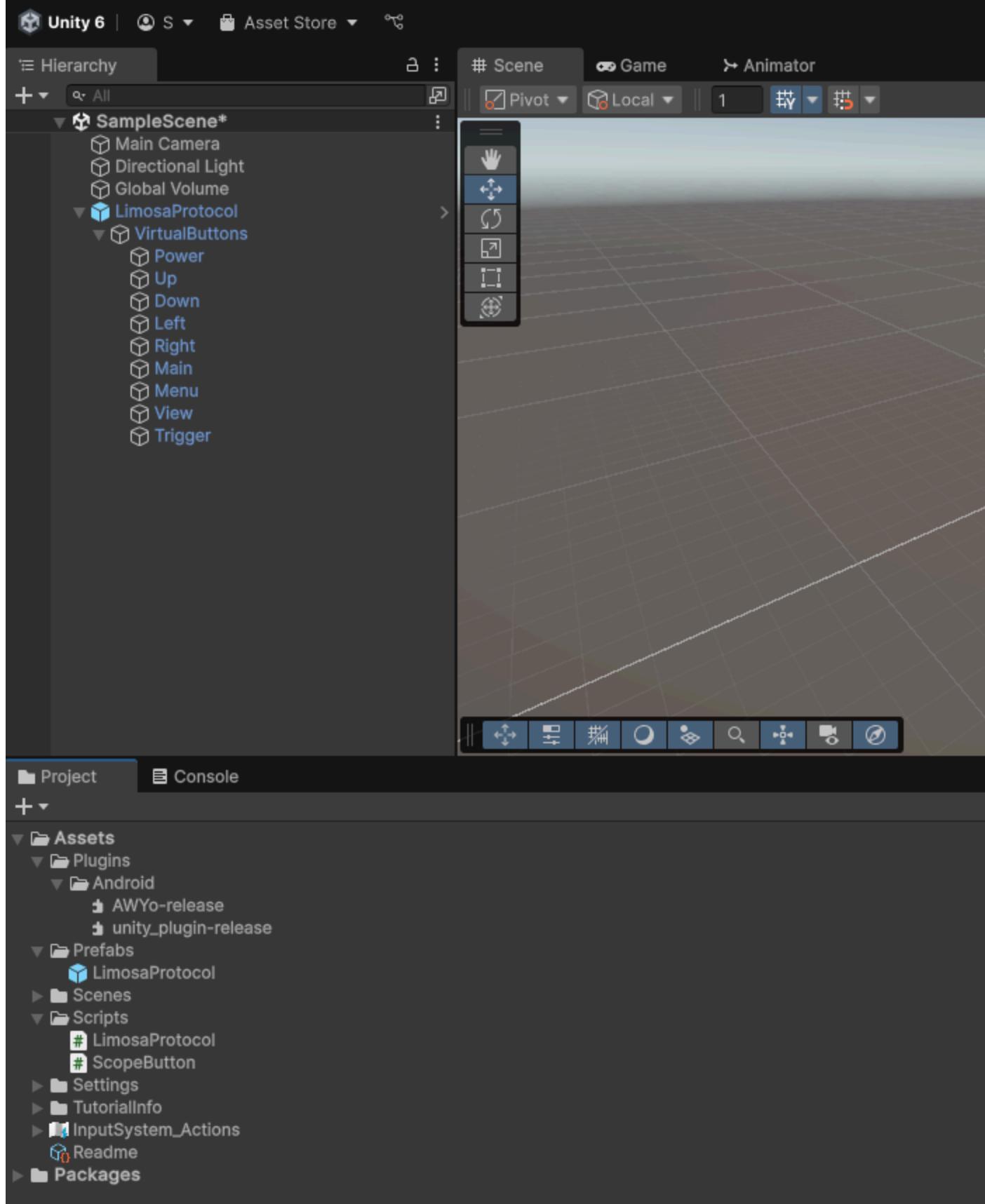
2. "Import Unity Package"のダイアログで、"Import"ボタンを推す



SDKの内容

ファイル	説明
AWYo-release.aar	Yo本体からのインテントを受信するAndroid ARchive
unity_plugin-release.aar	AWYo-release.aar で受信したインテントをトリガーにUnityに通知を送るAndroid ARchive
LimosaProtocol.prefab	Yoの機能を利用するためのプレハブアセット
LimosaProtocol.cs	Yoの機能を利用したり、Yoからの通知を受けるシングルトンクラス"LimosaProtocol"が定義されているC# ソースファイル。unity_plugin-release.aarと連携する。 LimosaProtocol.prefabにComponentとしてアタッチされている。
ScopeButton.cs	LimosaProtocolクラスで受け取ったScopeのボタン操作をUnityのInputControlにマッピングするためのクラス"ScopeButton"が定義されているC# ソースファイル。 LimosaProtocol.prefab内のScopeの各ボタンに対応するゲームオブジェクトにComponentとしてアタッチされている。

※ SDKをインストール後、LimosaProtocol.prefabをHierarchyウィンドウに追加した状態

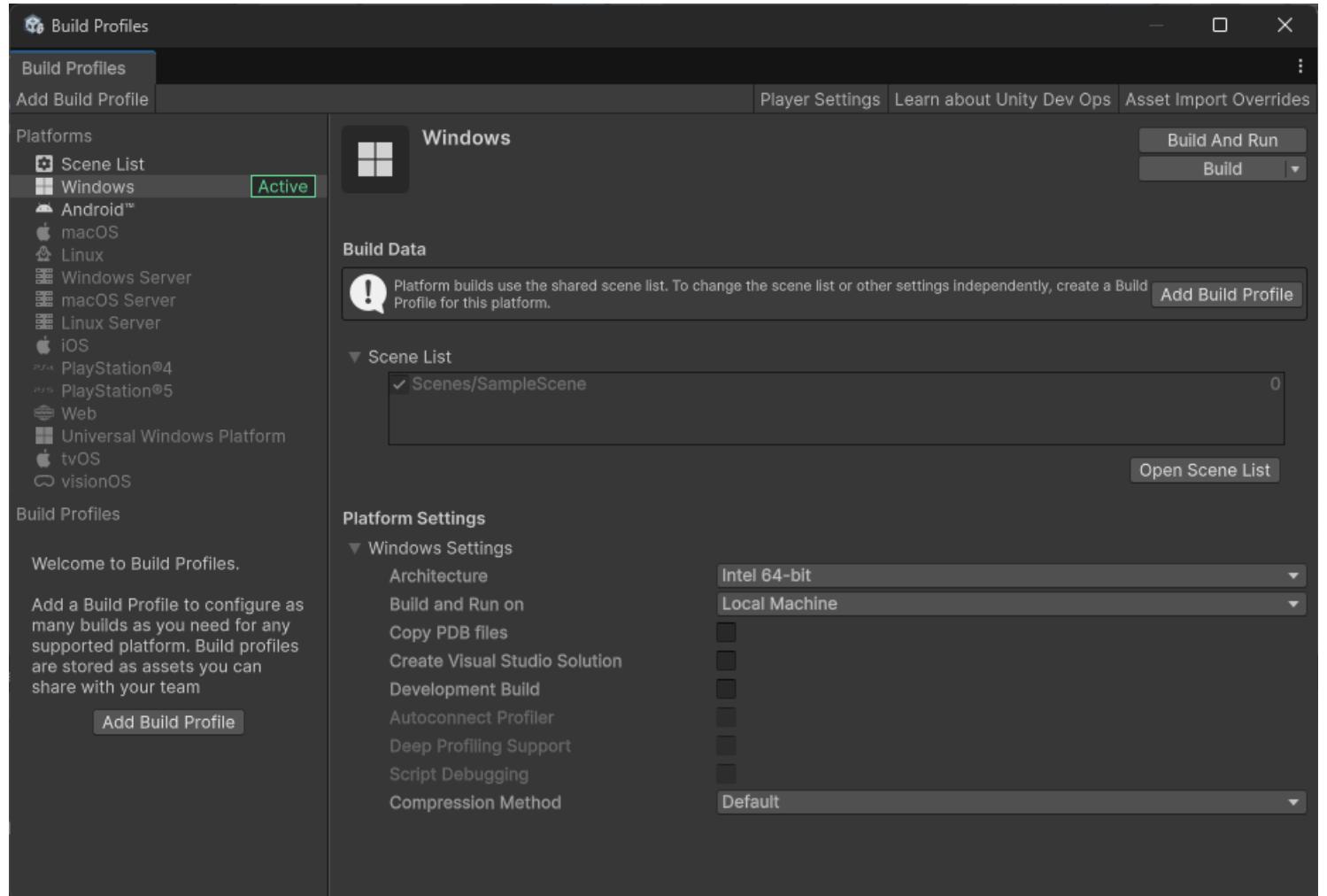


ビルド設定

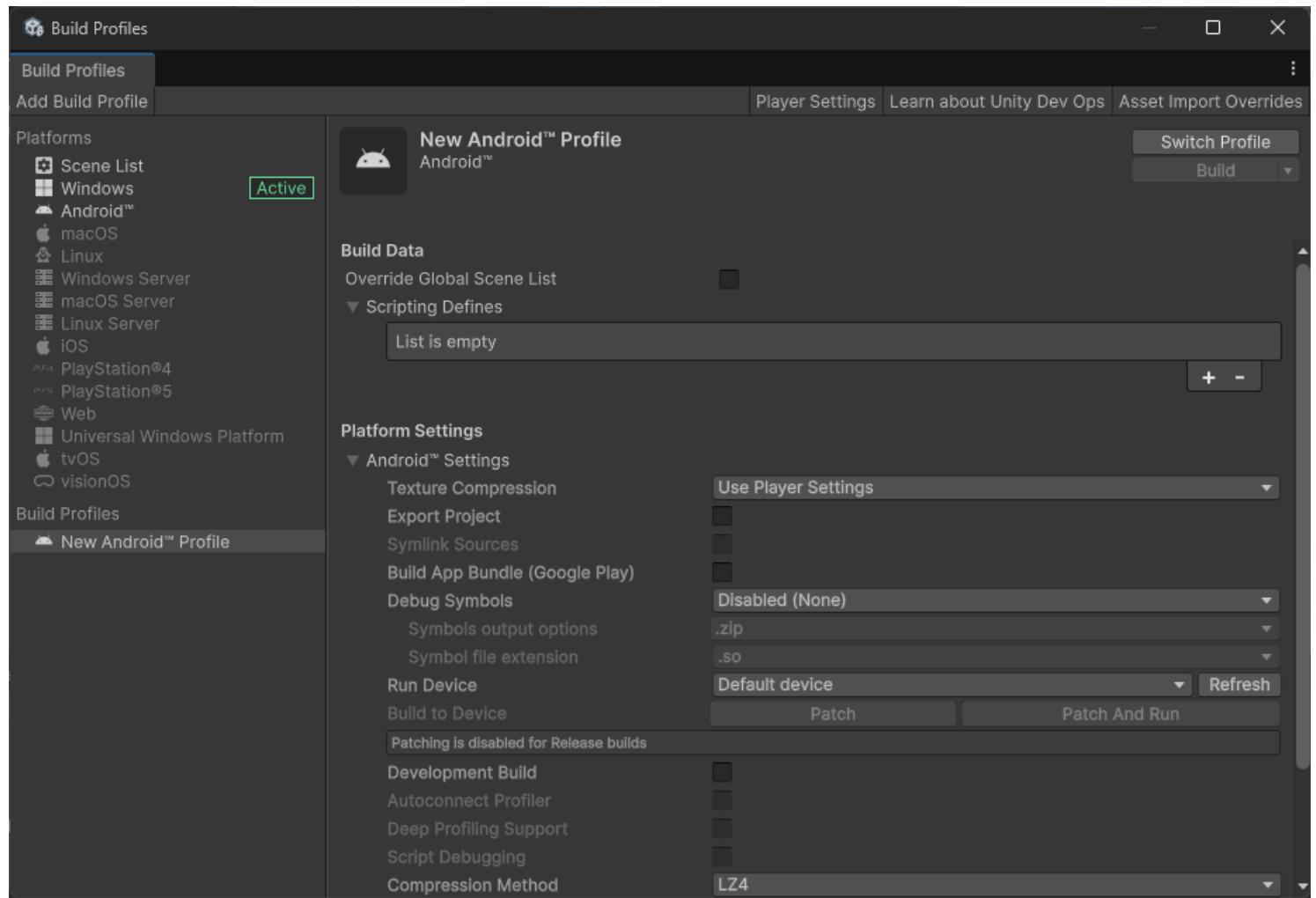
YoはAndroidベースのOSで動いていますので、ビルド設定はAndroid用プロファイルを用います。

1. Unityのメニューバーの次のボタンから"Build Profiles"ダイアログを表示する

File -> Build Profiles

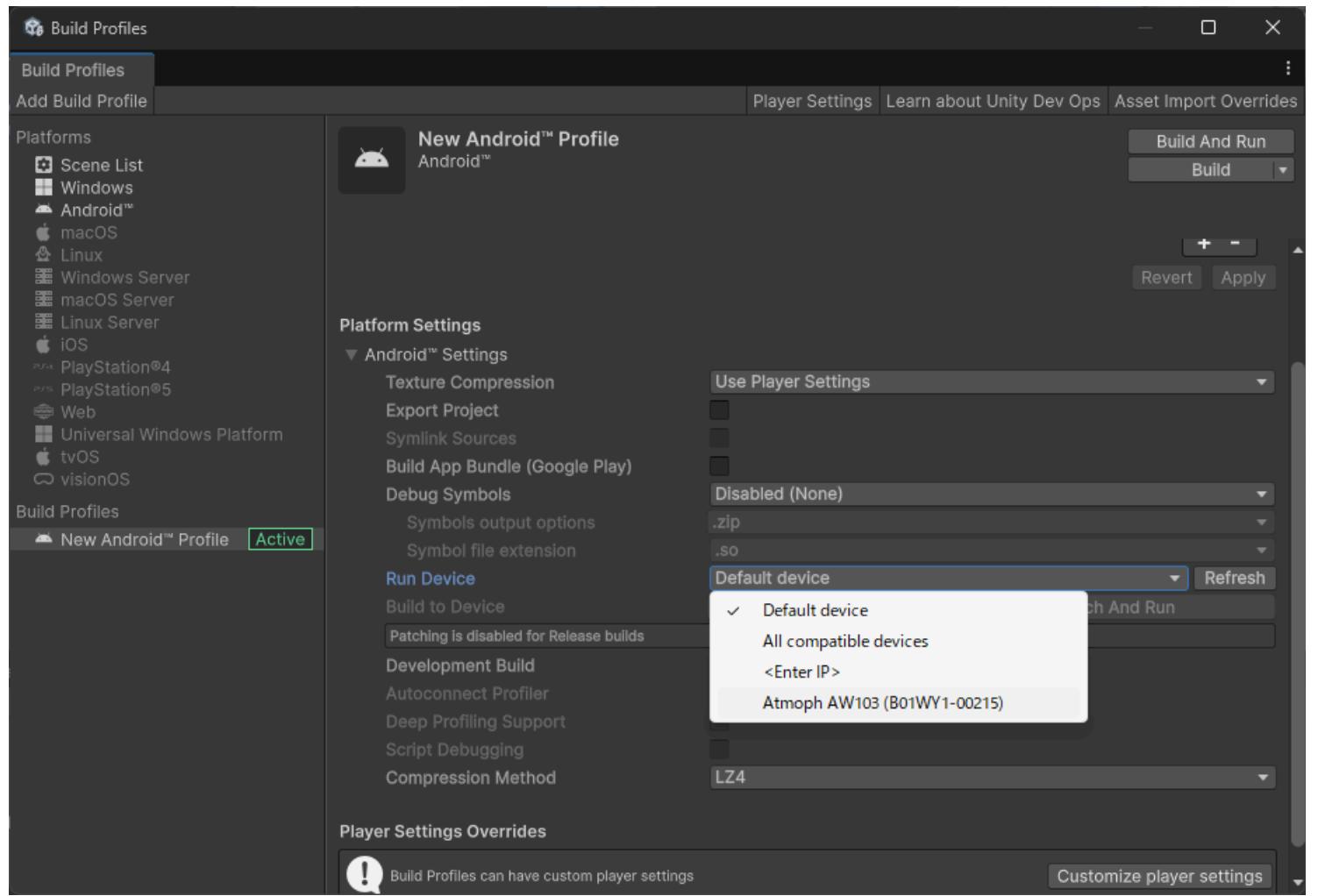


2. "Add Build Profile"ボタンで"Platform Browser"ダイアログを開き、"Android"を選択して"Add Build Profile"ボタンを押す。これで新しい"Android Prifile"ができる。



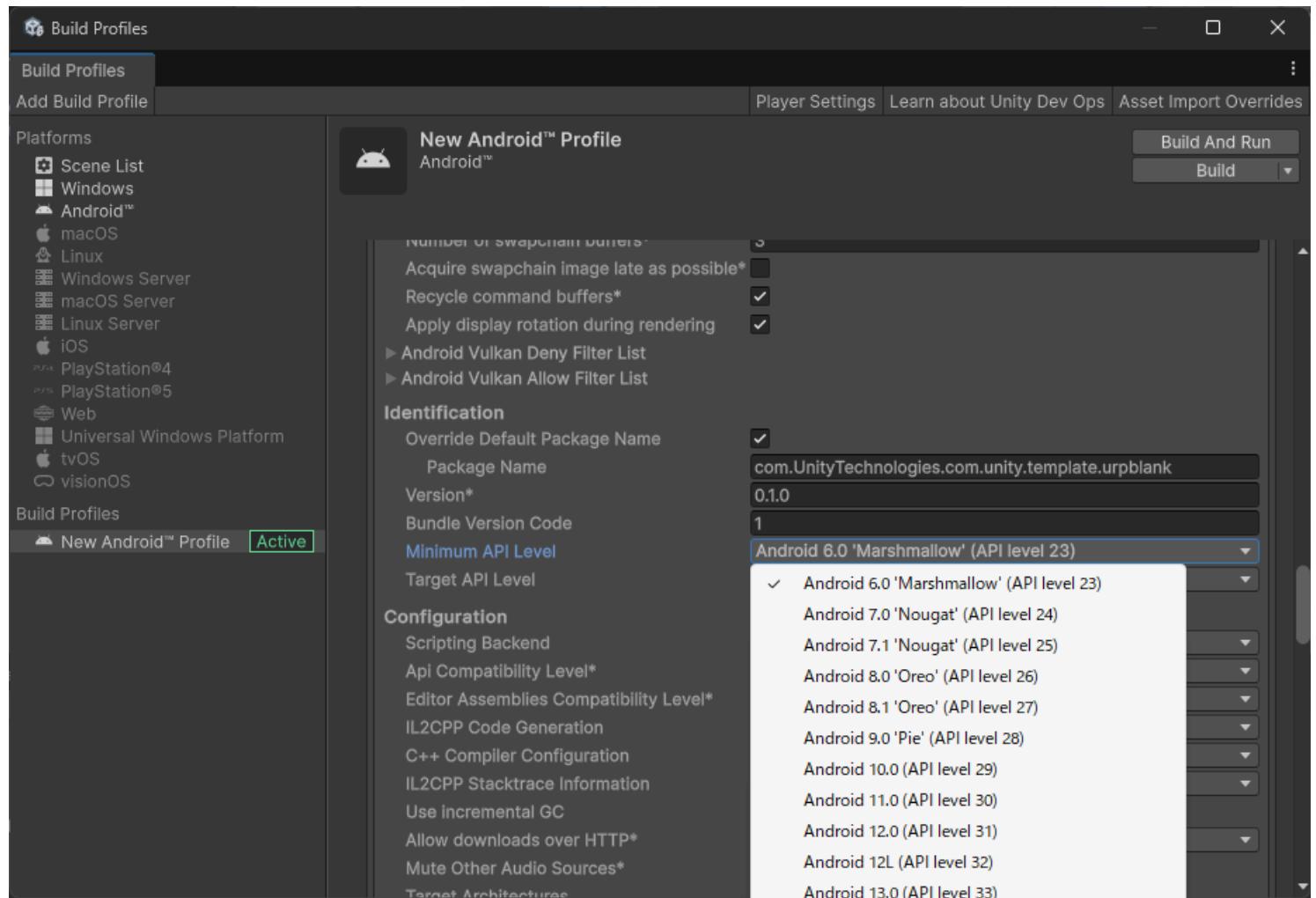
3. "Switch Profile"ボタンで2で追加したAndroid Profileに切り替える。
4. Android Profileの次の設定を変更する。4-1. "Run device"でYo開発機を選択

※図はPCにUSBでYo開発機を繋いだ状態。"Atmoph AW103(XXXXXX-XXXXXX)"がYo開発機。XXXXXX-XXXXXXのシリアルは開発機ごとに異なる。本体にIPアドレスを設定した場合、IPアドレスでも選択可能。



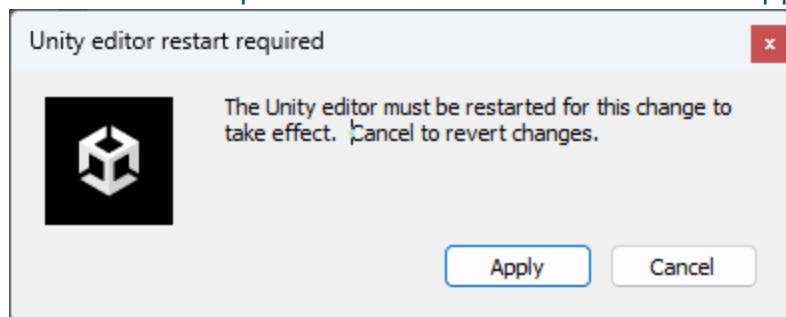
4-2. "Customize player settings"ボタンを押して設定項目を拡張

4-3. "Identification" -> "Minimum API Level"を"Android 13.0(API level 33)"に設定



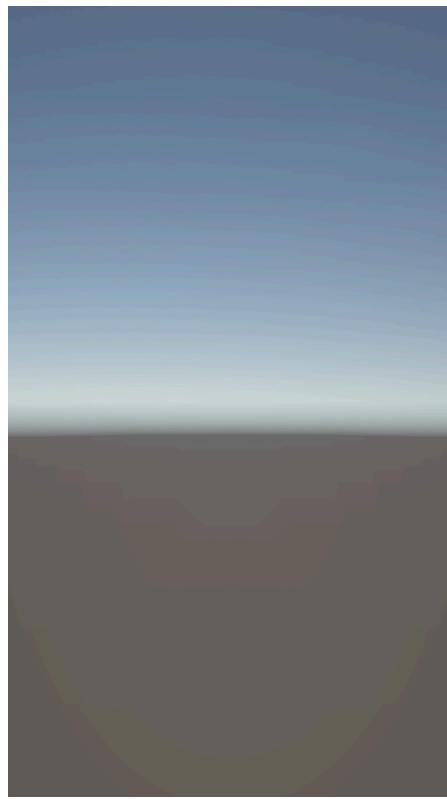
i NOTE

"Configuration" -> "Active Input Handling"はプロジェクトに合わせて設定してください。設定時に次の"Unity editor restart required"ダイアログが表示される場合は、"Apply"ボタンを押してUnity Editorを再起動します。



以上でビルド設定は完了です。Android Profileの"Build And Run"ボタンを押して、アプリがビルドされYo開発機で起動されたら成功です。

Universal 3Dテンプレートで作った空アプリでビルドと起動が成功すると、Yo開発機に次のような画面が表示されます。



※ "Build And Run"ボタンを押すとビルド結果のAPKファイルを保存するパスの設定が求められます。任意のパスを設定してください。

"使い方"の目次

- [1 基本的な使い方](#)
 - [1-1 Scopeのボタン操作の検知](#)
 - [1-1-1 ScopeボタンのInputControlへのマッピング](#)
 - [1-1-2 Scopeボタンの状態データを取得](#)
 - [1-2 Scopeのズームリングデータの取得](#)
 - [1-3 Scopeのポインティング位置、向き、距離センター値の取得](#)
 - [1-4 Scopeの動的情報の取得（接続状態、アドレス、バッテリー残量）](#)
 - [1-4-1 接続状態、アドレスの取得](#)
 - [1-4-1-1 LimosaProtocol.GetScopesメソッドを使用](#)
 - [1-4-1-2 LimosaProtocol.ScopeConnectionStateプロパティを使用](#)
 - [1-4-1-3 LimosaProtocol.OnConnectionStateChangedEventプロパティにコールバックメソッドを追加する](#)
 - [1-4-2 バッテリー残量の取得](#)
 - [1-5 Scopeの静的情報の取得\(本体型番、シリアルナンバー、FWバージョン、PnPID\)](#)
 - [1-6 Scopeへの振動指示](#)
 - [1-7 Unityアプリの終了を本体に伝える](#)
- [2 高度な使い方](#)
 - [2-1 Scopeのボタン操作のコールバックメソッドによる検知方法](#)
 - [2-2 Scopeのズームリングデータのコールバックメソッドによる取得方法](#)
 - [2-3 Scopeのバッテリー残量のコールバックメソッドによる取得方法](#)
 - [2-4 Scopeの接続状態、アドレス取得のコールバックメソッドによる取得方法](#)
 - [2-5 Scopeのポインティング位置、向き、距離センター値のコールバックメソッドによる取得方法](#)
 - [2-6 Scopeのボタン状態、ズームリングデータ、クオータニオンのコールバックメソッドによる取得方法](#)
- [3 その他の機能](#)
 - [3-1 Offset](#)
 - [3-1-1 LimosaProtocol.ScopeOffset、LimosaProtocol.ScopeRawOffset プロパティからのOffsetの取得](#)
 - [3-1-2 LimosaProtocol.OnOffsetResetコールバックによるOffsetの取得](#)

1 基本的な使い方

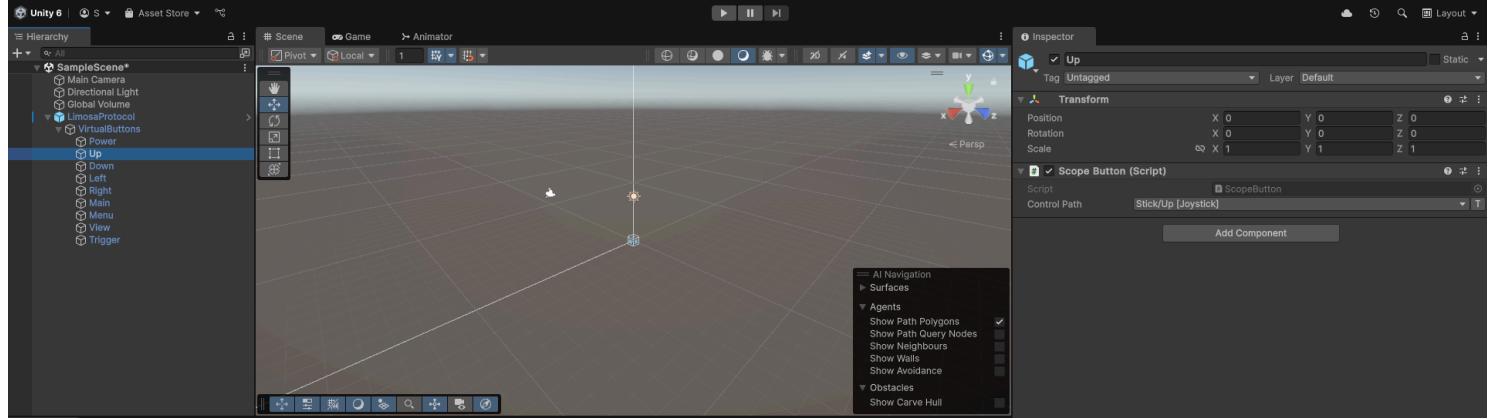
1-1 Scopeのボタン操作の検知

Scopeのボタン操作は次の二つの方法で検知できます。

1-1-1 ScopeボタンのInputControlへのマッピング

Scopeのボタン操作をUnityが扱えるボタン操作など各種InputControlにマッピングします。LimosaProtocol.prefabをHierarchyウィンドウに追加すると、その子オブジェクトのVirtualButtonsの子オブジェクトにScopeの各ボタンの名称のオ

プロジェクトでマッピング先のInputControlが設定できます。設定先はScope Button(Script)コンポーネントのControl Pathプロパティになります。



デフォルトのマッピングは次のようにになっています。

オブジェクト	InputControl
Power	(未設定)
Up	Stick/Up [Joystick]
Down	Stick/Down [Joystick]
Left	Stick/Left [Joystick]
Right	Stick/Right [Joystick]
Main	(未設定)
Menu	(未設定)
View	(未設定)
Trigger	(未設定)

Scope Button(Script)コンポーネントはカスタムのOn-Screen Controlで、ScopeButtonクラスはOnScreenControlクラスを継承しています。各ボタンが押された際にはSendValueToControl(1.0f)、離された際にはSendValueToControl(0.0f)が呼び出されます。

1-1-2 Scopeボタンの状態データを取得

[LimosaProtocol.ScopeRawData](#)プロパティからその時点のScopeの各種データを取得できます。プロパティの型は[IReadOnlyDictionary<string, RawData>](#)で、データを取得したScopeのアドレスをキーにRawDataインスタンスが取得できます。

(i) NOTE

ただし現状では同時にペアリングされるScopeは1台であるため、Dictionaryに同時に入る要素は一つであり、[IReadOnlyDictionary.FirstOrDefault](#)でもRawDataインスタンスの取得が可能です。

ボタンの状態データは[ButtonState](#)型である[RawData.button](#)プロパティに格納されています。

```
var data = Limosa.ScopeRawData.FirstOrDefault().Value;
Log($"ButtonState: {data.button}");
```

1-2 Scopeのズームリングデータの取得

[LimosaProtocol.ScopeRawData](#)プロパティからその時点のScopeの各種データを取得できます。プロパティの型は[IReadOnlyDictionary<string, RawData>](#)で、データを取得したScopeのアドレスをキーに[RawData](#)インスタンスが取得できます。

(i) NOTE

ただし現状では同時にペアリングされるScopeは1台であるため、Dictionaryに同時に入る要素は一つであり、[IReadOnlyDictionary.FirstOrDefault](#)でもRawDataインスタンスの取得が可能です。

ズームリングの状態データは[int](#)型である[RawData.zoom](#)プロパティに格納されています。

```
var data = Limosa.ScopeRawData.FirstOrDefault().Value;
Log($"Zoom: {data.zoom}");
```

1-3 Scopeのポインティング位置、向き、距離センター値の取得

[LimosaProtocol.ScopeConvertedData](#)プロパティからその時点のScopeの各種変換済みデータを取得できます。プロパティの型は[IReadOnlyDictionary<string, ConvertedData>](#)で、データを取得したScopeのアドレスをキーに[ConvertedData](#)インスタンスが取得できます。

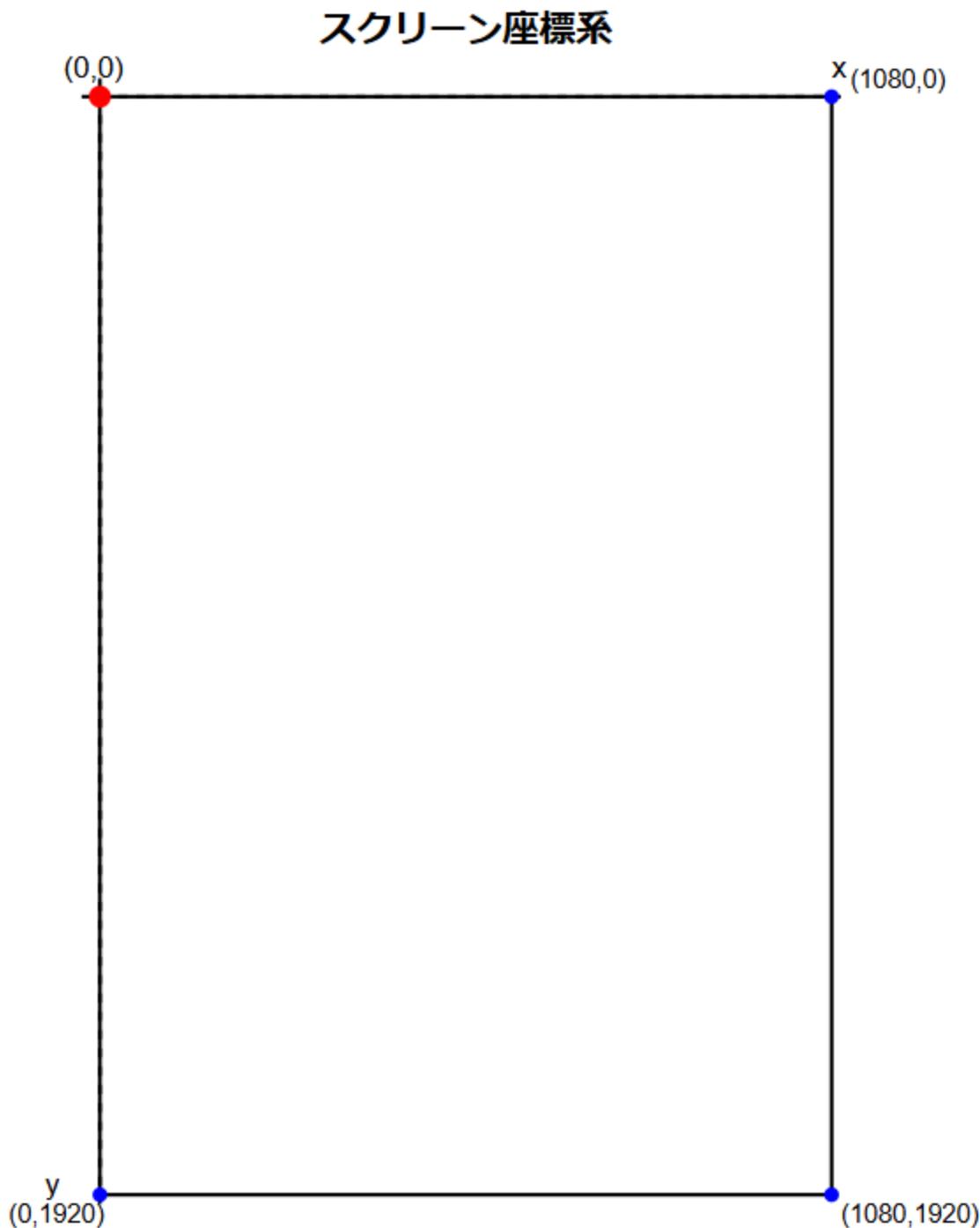
(i) NOTE

ただし現状では同時にペアリングされるScopeは1台であるため、Dictionaryに同時に入る要素は一つであり、[IReadOnlyDictionary.FirstOrDefault](#)でもRawDataインスタンスの取得が可能です。

ポインティング位置のデータは[float](#)型である[ConvertedData.x](#)、[ConvertedData.y](#)プロパティに格納されています。

```
var data = Limosa.ScopeConvertedData.FirstOrDefault().Value;  
Log($"Point position: x-{data.x},y-{data.y}");
```

ポインティング位置の座標はYoの画面に対して次のようにになります。



左上が原点(0,0)、横方向(x)は0～1080、縦方向(y)は0～1920

Scopeの向きは、[float](#)型である[ConvertedData.pitch](#)、[ConvertedData yaw](#)、[ConvertedData roll](#)プロパティに格納されています。

```
var data = Limosa.ScopeConvertedData.FirstOrDefault().Value;
Log($"Scope direction: pitch-{data.pitch}, yaw-{data.yaw}, roll-{data.roll}");
```

i NOTE

クオータニオンでScopeの向きを取得したい場合は、[LimosaProtocol.ScopeRawData](#)インスタンスから取得できます。インスタンスの取得方法は[1-1-2 Scopeボタンの状態データを取得](#)を参照してください。

距離センター値は、[float](#)型である[ConvertedData.distance](#)プロパティに格納されています。Scopeの[距離センサー](#)からScopeの筒の方向の障害物までの距離(mm)になります。

```
var data = Limosa.ScopeConvertedData.FirstOrDefault().Value;
Log($"Scope distance: {data.distance}");
```

1-4 Scopeの動的情報の取得（接続状態、アドレス、バッテリー残量）

1-4-1 接続状態、アドレスの取得

2通りの方法でScopeとの接続状態やアドレスが取得できます。

1-4-1-1 [LimosaProtocol.GetScopes](#)メソッドを使用

[LimosaProtocol.GetScopes](#)メソッドでScopeのアドレスや接続状態が取得できます。返値の型は[BondedScope](#)です。返値の要素の[BondedScope.scopes](#)は[ScopeState](#)のリストになっており、そのプロパティからScopeのアドレスや接続状態が取得できます。

```
var bondedScope = Limosa.LimosaProtocol.GetScopes();
foreach (Limosa.BondedScope.ScopeState scope in bondedScope)
{
    Log($"ScopeStateBatteryLevel: {scope}");
}
```

1-4-1-2 [LimosaProtocol.ScopeConnectionState](#)プロパティを使用

[LimosaProtocol.ScopeConnectionState](#)プロパティからScopeのアドレスと接続状態を取得できます。プロパティの型は[IReadOnlyDictionary<string, bool>](#)で、データを取得したScopeのアドレスをキーに接続状態を[bool](#)型のデータで取得できます。trueは接続中、falseは非接続を表します。ただし現状では同時にペアリングされるScopeは1台であるため、Dictionaryに同時に入る要素は一つであり、[IReadOnlyDictionary.FirstOrDefault](#)でもデータの取得が可能です。

```
var state = Limosa.ScopeConnectionState.FirstOrDefault().Value;
Log($"ScopeConnectionState: address-{state.Key}, connection state-{state.Value}");
```

1-4-1-3 [LimosaProtocol.OnConnectionStateChangedEvent](#)プロパティにコードを追加する

[LimosaProtocol.OnConnectionStateChangedEvent](#)プロパティは[UnityEvent](#)型で、追加したコールバックメソッドは新しいScopeが接続されたタイミングと、接続されていたScopeの接続が切れたタイミングで呼び出されます。

コールバックメソッドの追加や削除は[UnityEvent](#)のメソッドを使って実施してください。

```
Limosa.LimosaProtocol.OnConnectionStateChangedEvent.AddListener(OnScopeStateChanged);
private void OnScopeStateChanged()
{
    var state = Limosa.LimosaProtocol.ScopeConnectionState.FirstOrDefault();
    Debug.Log($"OnScopeStateChanged. {state.Key}: {state.Value}");
}

Limosa.LimosaProtocol.OnConnectionStateChangedEvent.RemoveListener(OnScopeStateChanged);
```

⊗ IMPORTANT

アプリ終了時には、[LimosaProtocol.OnConnectionStateChangedEvent](#)プロパティから追加したコールバックメソッドを全て削除してください。

1-4-2 バッテリー残量の取得

[LimosaProtocol.GetBatteryLevel](#)メソッドでScopeのバッテリー残量を取得できます。引数で指定したアドレスのScopeのバッテリー残業が、[int](#)型の返値で返ります。値は満充電に対するバッテリー残量の割合となっており、パーセンテージで0～100の値になります。

```
var batteryLevel = Limosa.LimosaProtocol.GetBatteryLevel("address");
Log($"BatteryLevel: {batteryLevel}");
```

i NOTE

Scopeとの接続直後など、Scopeのバッテリー残量が正確に把握できていない状態では、Scopeと接続されていてもバッテリー残量の値は'-1'になります。

1-5 Scopeの静的情報の取得(本体型番、シリアルナンバー、FWバージョン、PnPID)

[LimosaProtocol.GetScopeInfo](#)メソッドで、引数で指定したアドレスのScopeの静的情報を取得できます。返値の型は[ScopeInfo](#)です。

```
var scopeInfo = Limosa.LimosaProtocol.GetBatteryLevel("address");
Log($"ScopeInfo: {scopeInfo}");
```

1-6 Scopeへの振動指示

次の2つのメソッドで、Scopeのバイブレーション機能を起動できます。

[LimosaProtocol.Vibrate](#) [LimosaProtocol.VibrateToAll](#)

前者はtargetAddress引数で指定したScopeのみの振動指示、後者は接続している全Scopeへの振動指示となります。pattern引数で振動の種類を指示します。次の仕様になります。

Effect ID#	Waveform Name	Effect ID#	Waveform Name	Effect ID#	Waveform Name
1	Strong click – 100%	42	Long double sharp click medium 2 – 80%	83	Transition ramp up long smooth 2 – 0 to 100%
2	Strong click – 60%	43	Long double sharp click medium 3 – 60%	84	Transition ramp up medium smooth 1 – 0 to 100%
3	Strong click – 30%	44	Long double sharp tick 1 – 100%	85	Transition ramp up medium smooth 2 – 0 to 100%
4	Sharp click – 100%	45	Long double sharp tick 2 – 80%	86	Transition ramp up short smooth 1 – 0 to 100%
5	Sharp click – 60%	46	Long double sharp tick 3 – 60%	87	Transition ramp up short smooth 2 – 0 to 100%
6	Sharp click – 30%	47	Buzz 1 – 100%	88	Transition ramp up long sharp 1 – 0 to 100%
7	Soft bump – 100%	48	Buzz 2 – 80%	89	Transition ramp up long sharp 2 – 0 to 100%
8	Soft bump – 60%	49	Buzz 3 – 60%	90	Transition ramp up medium sharp 1 – 0 to 100%
9	Soft bump – 30%	50	Buzz 4 – 40%	91	Transition ramp up medium sharp 2 – 0 to 100%
10	Double click – 100%	51	Buzz 5 – 20%	92	Transition ramp up short sharp 1 – 0 to 100%
11	Double click – 60%	52	Pulsing strong 1 – 100%	93	Transition ramp up short sharp 2 – 0 to 100%
12	Triple click – 100%	53	Pulsing strong 2 – 60%	94	Transition ramp down long smooth 1 – 50 to 0%
13	Soft fuzz – 60%	54	Pulsing medium 1 – 100%	95	Transition ramp down long smooth 2 – 50 to 0%
14	Strong buzz – 100%	55	Pulsing medium 2 – 60%	96	Transition ramp down medium smooth 1 – 50 to 0%
15	750-ms alert 100%	56	Pulsing sharp 1 – 100%	97	Transition ramp down medium smooth 2 – 50 to 0%
16	1000-ms alert 100%	57	Pulsing sharp 2 – 60%	98	Transition ramp down short smooth 1 – 50 to 0%
17	Strong click 1 – 100%	58	Transition click 1 – 100%	99	Transition ramp down short smooth 2 – 50 to 0%
18	Strong click 2 – 80%	59	Transition click 2 – 80%	100	Transition ramp down long sharp 1 – 50 to 0%
19	Strong click 3 – 60%	60	Transition click 3 – 60%	101	Transition ramp down long sharp 2 – 50 to 0%
20	Strong click 4 – 30%	61	Transition click 4 – 40%	102	Transition ramp down medium sharp 1 – 50 to 0%
21	Medium click 1 – 100%	62	Transition click 5 – 20%	103	Transition ramp down medium sharp 2 – 50 to 0%
22	Medium click 2 – 80%	63	Transition click 6 – 10%	104	Transition ramp down short sharp 1 – 50 to 0%
23	Medium click 3 – 60%	64	Transition hum 1 – 100%	105	Transition ramp down short sharp 2 – 50 to 0%
24	Sharp tick 1 – 100%	65	Transition hum 2 – 80%	106	Transition ramp up long smooth 1 – 0 to 50%
25	Sharp tick 2 – 80%	66	Transition hum 3 – 60%	107	Transition ramp up long smooth 2 – 0 to 50%
26	Sharp tick 3 – 60%	67	Transition hum 4 – 40%	108	Transition ramp up medium smooth 1 – 0 to 50%
27	Short double click strong 1 – 100%	68	Transition hum 5 – 20%	109	Transition ramp up medium smooth 2 – 0 to 50%
28	Short double click strong 2 – 80%	69	Transition hum 6 – 10%	110	Transition ramp up short smooth 1 – 0 to 50%
29	Short double click strong 3 – 60%	70	Transition ramp down long smooth 1 – 100 to 0%	111	Transition ramp up short smooth 2 – 0 to 50%
30	Short double click strong 4 – 30%	71	Transition ramp down long smooth 2 – 100 to 0%	112	Transition ramp up long sharp 1 – 0 to 50%
31	Short double click medium 1 – 100%	72	Transition ramp down medium smooth 1 – 100 to 0%	113	Transition ramp up long sharp 2 – 0 to 50%
32	Short double click medium 2 – 80%	73	Transition ramp down medium smooth 2 – 100 to 0%	114	Transition ramp up medium sharp 1 – 0 to 50%
33	Short double click medium 3 – 60%	74	Transition ramp down short smooth 1 – 100 to 0%	115	Transition ramp up medium sharp 2 – 0 to 50%
34	Short double sharp tick 1 – 100%	75	Transition ramp down short smooth 2 – 100 to 0%	116	Transition ramp up short sharp 1 – 0 to 50%
35	Short double sharp tick 2 – 80%	76	Transition ramp down long sharp 1 – 100 to 0%	117	Transition ramp up short sharp 2 – 0 to 50%
36	Short double sharp tick 3 – 60%	77	Transition ramp down long sharp 2 – 100 to 0%	118	Long buzz for programmatic stopping – 100%
37	Long double sharp click strong 1 – 100%	78	Transition ramp down medium sharp 1 – 100 to 0%	119	Smooth hum 1 (No kick or brake pulse) – 50%
38	Long double sharp click strong 2 – 80%	79	Transition ramp down medium sharp 2 – 100 to 0%	120	Smooth hum 2 (No kick or brake pulse) – 40%
39	Long double sharp click strong 3 – 60%	80	Transition ramp down short sharp 1 – 100 to 0%	121	Smooth hum 3 (No kick or brake pulse) – 30%
40	Long double sharp click strong 4 – 30%	81	Transition ramp down short sharp 2 – 100 to 0%	122	Smooth hum 4 (No kick or brake pulse) – 20%
41	Long double sharp click medium 1 – 100%	82	Transition ramp up long smooth 1 – 0 to 100%	123	Smooth hum 5 (No kick or brake pulse) – 10%

1-7 Unityアプリの終了を本体に伝える

[LimosaProtocol.GoBackToLimosa](#)メソッドでアプリケーションの終了を本体に伝え、風景再生画面に戻ります。

④ IMPORTANT

アプリケーション終了時には必ず呼び出してください。

⑤ NOTE

アプリケーションの終了処理はアプリケーション自身で行う必要があります。

2 高度な使い方

④ NOTE

LimosaProtocolを継承するカスタムクラスを作成する方法を採用した場合、LimosaProtocolにアタッチされている LimosaProtocol.csコンポーネントをカスタムクラスに差し替えてください。

2-1 Scopeのボタン操作のコールバックメソッドによる検知方法

Scopeのボタン操作を検知する方法で、[1-1 Scopeのボタン操作の検知](#)よりも自由度の高い方法として、「Scopeボタンの状態が変わった際に呼ばれるコールバックメソッドをオーバーライド」があります。LimosaProtocolを継承するカスタムクラスを作成し、[LimosaProtocol.OnButtonsStateChanged](#)メソッドをオーバーライドします。このメソッドはScopeボタンの状態が変わったタイミングで呼び出されますので、必要な処理を実装することができます。

カスタム実装には[LimosaProtocol.OnButtonsStateChanged](#)メソッドの実装を参考にできます。引数のdataを使って、次のコードでButtonStateChangedインスタンスを取得できます。

```
ButtonStateChanged buttonStates = JsonUtility.FromJson<ButtonStateChanged>(data);
```

2-2 Scopeのズームリングデータのコールバックメソッドによる取得方法

Scopeのズームリングデータを取得する方法で、[1-2 Scopeのズームリングデータの取得](#)よりも自由度の高い方法として、「Scopeのズームリングの値が変わった際に呼ばれるコールバックメソッドをオーバーライド」があります。LimosaProtocolを継承するカスタムクラスを作成し、[LimosaProtocol.OnEncoderValueChanged](#)メソッドをオーバーライドします。このメソッドはScopeのズームリングの値が変わったタイミングで呼び出されますので、必要な処理を実装することができます。

カスタム実装には[LimosaProtocol.OnEncoderValueChanged](#)メソッドの実装を参考にできます。引数のdataを使って、次のコードで[IntegerChanged](#)インスタンスを取得できます。

```
IntegerChanged integerChanged = JsonUtility.FromJson<IntegerChanged>(data);
```

[IntegerChanged.previous](#)で変更前の値、[IntegerChanged.current](#)で変更後の現在の値が取得できます。

2-3 Scopeのバッテリー残量のコールバックメソッドによる取得方法

Scopeのバッテリー残量を取得する方法で、[1-4-2 バッテリー残量の取得](#)よりも自由度の高い方法として、「Scopeのバッテリー残量の値が変わった際に呼ばれるコールバックメソッドをオーバーライド」があります。LimosaProtocolを継承するカスタムクラスを作成し、[LimosaProtocol.OnBatteryLevelChanged](#)メソッドをオーバーライドします。このメソッドはScopeのバッテリー残量の値が変わったタイミングで呼び出されますので、必要な処理を実装することができます。

カスタム実装には[LimosaProtocol.OnEncoderValueChangedOnBatteryLevelChanged](#)メソッドの実装を参考にできます。引数のdataを使って、次のコードで[IntegerChanged](#)インスタンスを取得できます。

```
IntegerChanged integerChanged = JsonUtility.FromJson<IntegerChanged>(data);
```

[IntegerChanged.previous](#)で変更前の値、[IntegerChanged.current](#)で変更後の現在の値が取得できます。

2-4 Scopeの接続状態、アドレス取得のコールバックメソッドによる取得方法

Scopeの接続状態やアドレスを取得する方法で、[1-4-1 接続状態、アドレスの取得](#)よりも自由度の高い方法として、「Scopeが接続された際、また接続が切れた際に呼ばれるコールバックメソッドをオーバーライド」があります。

LimosaProtocolを継承するカスタムクラスを作成し、[LimosaProtocol.OnConnected](#)メソッド、[LimosaProtocol.OnDisconnected](#)メソッドをオーバーライドします。前者のメソッドはScopeが接続されたタイミング、後者のメソッドは接続が切れたで呼び出されますので、必要な処理を実装することができます。

カスタム実装には[LimosaProtocol.OnConnected](#)メソッド、[LimosaProtocol.OnDisconnected](#)メソッドの実装を参考にできます。引数のdataを使って、次のコードで[ScopeAddress](#)インスタンスを取得できます。これが接続された、または接続が切れたScopeのアドレスになります。

```
ScopeAddress scopeAddress = JsonUtility.FromJson<ScopeAddress>(data);
```

2-5 Scopeのポインティング位置、向き、距離センター値のコールバックメソッドによる取得方法

Scopeのポインティング位置、向き、距離センター値を取得する方法で、[1-3 Scopeのポインティング位置、向き、距離センター値の取得](#)よりも自由度の高い方法として、「Scopeの[ScopeConvertedData](#)データが更新された際に呼ばれるコールバックメソッドをオーバーライド」があります。LimosaProtocolを継承するカスタムクラスを作成し、[LimosaProtocol.OnDataConverted](#)メソッドをオーバーライドします。このメソッドはScopeの[ScopeConvertedData](#)に含まれるデータが更新されたタイミングで呼び出されますので、必要な処理を実装することができます。

カスタム実装には[LimosaProtocol.OnDataConverted](#)メソッドの実装を参考にできます。引数のdataを使って、次のコードで[ScopeConvertedData](#)インスタンスを取得できます。

```
ConvertedData convertedData = JsonUtility.FromJson<ConvertedData>(data);
```

2-6 Scopeのボタン状態、ズームリングデータ、クオータニオンのコールバックメソッドによる取得方法

Scopeのボタン状態、ズームリングデータ、クオータニオンを取得する方法で、[1-2 Scopeのズームリングデータの取得](#)や[1-1-2 Scopeボタンの状態データを取得](#)よりも自由度の高い方法として、「Scopeの[RawData](#)データが更新された際に呼ばれるコールバックメソッドをオーバーライド」があります。LimosaProtocolを継承するカスタムクラスを作成し、[LimosaProtocol.OnRawData](#)メソッドをオーバーライドします。このメソッドはScopeの[RawData](#)に含まれるデータが更新されたタイミングで呼び出されますので、必要な処理を実装することができます。

カスタム実装には[LimosaProtocol.OnRawData](#)メソッドの実装を参考にできます。引数のdataを使って、次のコードで[RawData](#)インスタンスを取得できます。

```
RawData rawData = JsonUtility.FromJson<RawData>(data);
```

3 その他の機能

3-1 Offset

Scopeには[メニュー](#)[ボタン](#)長押しで起動する「ポインティング位置のリセット」機能があります。風景再生時には機能が実行されるとポインティング位置が画面中央にリセットされます。このような機能をアプリケーションにも組み込むには、リセット時に保持されるOffset値を使います。

3-1-1 [LimosaProtocol.ScopeOffset](#)、[LimosaProtocol.ScopeRawOffset](#) プロパティからのOffsetの取得

最後にリセット機能が実行された際のScopeの[ScopeConvertedData](#)、[RawData](#)は、Offset値としてそれぞれ次のプロパティに保持されます。

- [LimosaProtocol.ScopeOffset](#)
- [LimosaProtocol.ScopeRawOffset](#)

風景再生時のようなポインティング位置のリセット機能をアプリケーションに組み込みたい場合は、これらOffset値を次のプロパティから取得できる現時点の[ScopeConvertedData](#)、[RawData](#)データから引くことで、リセット時からの遷移分のデータが取得できます。

- [LimosaProtocol.ScopeConvertedData](#)
- [LimosaProtocol.ScopeRawData](#)

3-1-2 [LimosaProtocol.OnOffsetReset](#)コールバックによるOffsetの取得

「Scopeのポインティング位置のリセット機能が実行された際に呼ばれるコールバックメソッドをオーバーライド」でもOffset値の取得が可能です。LimosaProtocolを継承するカスタムクラスを作成し、[LimosaProtocol.OnOffsetReset](#)メソッドをオーバーライドします。このメソッドはScopeのポインティング位置のリセットが実行されたタイミングで呼び出されますので、必要な処理を実装することができます。

カスタム実装には[LimosaProtocol.OnOffsetReset](#)メソッドの実装を参考にできます。引数のdataを使って、次のコードで[ScopeOffset](#)インスタンスを取得できます。

```
ScopeOffset scopeOffset = JsonUtility.FromJson<ScopeOffset>(data);
```

Namespace Limosa

Classes

[BondedScope](#)

Type for all paired Scope.

[BondedScope.ScopeState](#)

Type of Scope's state

[ButtonState](#)

Type of ButtonState

[ButtonStateChanged](#)

Type for ButtonState change This contains the previous and current ButtonState instances after the state has changed.

[ConvertedData](#)

Type of Scope converted data that is calculated from raw data see: RawData

[IntegerChanged](#)

Type for Integer value change This contains the previous and current value after the value has changed.

[LimosaProtocol](#)

Singleton class that serves as the base for Yo itself and Scope operations and data exchange

[RawData](#)

Type of Scope Raw data

[ScopeAddress](#)

Type of Scope address

[ScopeButton](#)

Class to map the Scope button to the control path. This class is set to button objects in VirtualButtons of LimosaProtocol.prefab

[ScopeInfo](#)

Type of Scope static information.

[ScopeOffset](#)

Type of Scope offset Elements are similar to elements of type ConvertedData. see: ConvertedData

[ServiceBindState](#)

Type for state indicating whether the service in AAR is bound or not.

Vector4f

Type of Vector4f This is used for quaternion. see: RawData.quaternion

Class BondedScope

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type for all paired Scope.

```
[Serializable]
public class BondedScope
```

Inheritance

[object](#) ← BondedScope

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

scopes

List of Scope's state. see: ScopeState

```
public List<BondedScope.ScopeState> scopes
```

Field Value

[List](#) <[BondedScope.ScopeState](#)>

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#) ↗

Class BondedScope.ScopeState

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope's state

```
[Serializable]
public class BondedScope.ScopeState
```

Inheritance

[object](#) ← BondedScope.ScopeState

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

address

String of Scope address

```
public string address
```

Field Value

[string](#)

state

Whether the service in AAR is bound or not. true: Bound false: Not bound

```
public bool state
```

Field Value

[bool](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#)

Class ButtonState

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of ButtonState

```
[Serializable]  
public class ButtonState
```

Inheritance

[object](#) ← ButtonState

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

down

Whether "Down" button is pressed or not. 1: Pressed 0: Not pressed

```
public int down
```

Field Value

[int](#)

left

Whether "Left" button is pressed or not. 1: Pressed 0: Not pressed

```
public int left
```

Field Value

[int](#)

main

Whether "Main" button is pressed or not. 1: Pressed 0: Not pressed

```
public int main
```

Field Value

[int](#)

menu

Whether "Menu" button is pressed or not. 1: Pressed 0: Not pressed

```
public int menu
```

Field Value

[int](#)

power

Whether "Power" button is pressed or not. 1: Pressed 0: Not pressed

```
public int power
```

Field Value

[int](#)

right

Whether "Right" button is pressed or not. 1: Pressed 0: Not pressed

```
public int right
```

Field Value

[int ↗](#)

trigger

Whether "Trigger" button is pressed or not. 1: Pressed 0: Not pressed

```
public int trigger
```

Field Value

[int ↗](#)

up

Whether "Up" button is pressed or not. 1: Pressed 0: Not pressed

```
public int up
```

Field Value

[int ↗](#)

value

Bitmap for all buttons. The order of the following elements indicates whether or not each button is pressed, and this is stored as a bit. (If it is pressed, it is 1.)

```
public int value
```

Field Value

[int](#)

view

Whether "View" button is pressed or not. 1: Pressed 0: Not pressed

```
public int view
```

Field Value

[int](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#)

Class ButtonStateChanged

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type for ButtonState change This contains the previous and current ButtonState instances after the state has changed.

```
[Serializable]  
public class ButtonStateChanged
```

Inheritance

[object](#) ← ButtonStateChanged

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

address

String of Scope address

```
public string address
```

Field Value

[string](#)

current

Current ButtonState instances see: ButtonState

```
public ButtonState current
```

Field Value

[ButtonState](#)

previous

Previous ButtonState instances see: ButtonState

```
public ButtonState previous
```

Field Value

[ButtonState](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#) ↗

Class ConvertedData

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope converted data that is calculated from raw data see: RawData

```
[Serializable]  
public class ConvertedData
```

Inheritance

[object](#) ← ConvertedData

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

address

String of Scope address. Also see: ScopeAddress

```
public string address
```

Field Value

[string](#)

distance

Destance from Scope distance sensor to obstacles Unit is mm

```
public float distance
```

Field Value

[float](#)

pitch

Pitch degree of scope orientation

```
public float pitch
```

Field Value

[float](#)

roll

Roll degree of scope orientation

```
public float roll
```

Field Value

[float](#)

x

X-axis value of pointing position on the Yo screen 0.0~1080.0

```
public float x
```

Field Value

[float](#)

y

Y-axis value of pointing position on the Yo screen 0.0~1920.0

```
public float y
```

Field Value

[float](#)

yaw

Yaw degree of scope orientation

```
public float yaw
```

Field Value

[float](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#)

Class IntegerChanged

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type for Integer value change This contains the previous and current value after the value has changed.

```
[Serializable]  
public class IntegerChanged
```

Inheritance

[object](#) ← IntegerChanged

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

address

String of Scope address

```
public string address
```

Field Value

[string](#)

current

Current int value

```
public int current
```

Field Value

[int](#)

previous

Previous int value

```
public int previous
```

Field Value

[int](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#)

Class LimosaProtocol

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Singleton class that serves as the base for Yo itself and Scope operations and data exchange

```
public class LimosaProtocol : MonoBehaviour
```

Inheritance

[object](#) ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← LimosaProtocol

Inherited Members

[MonoBehaviour.IsInvoking\(\)](#) , [MonoBehaviour.CancelInvoke\(\)](#) , [MonoBehaviour.Invoke\(string, float\)](#) ,
[MonoBehaviour.InvokeRepeating\(string, float, float\)](#) , [MonoBehaviour.CancelInvoke\(string\)](#) ,
[MonoBehaviour.IsInvoking\(string\)](#) , [MonoBehaviour.StartCoroutine\(string\)](#) ,
[MonoBehaviour.StartCoroutine\(string, object\)](#) , [MonoBehaviour.StartCoroutine\(IEnumerator\)](#) ,
[MonoBehaviour.StartCoroutine_Auto\(IEnumerator\)](#) , [MonoBehaviour.StopCoroutine\(IEnumerator\)](#) ,
MonoBehaviour.StopCoroutine(Coroutine) , [MonoBehaviour.StopCoroutine\(string\)](#) ,
MonoBehaviour.StopAllCoroutines() , [MonoBehaviour.print\(object\)](#) ,
[MonoBehaviour.destroyCancellationToken](#) , [MonoBehaviour.useGUILayout](#) ,
[MonoBehaviour.didStart](#) , [MonoBehaviour.didAwake](#) , MonoBehaviour.runInEditMode ,
Behaviour.enabled , Behaviour.isActiveAndEnabled , [Component.GetComponent\(Type\)](#) ,
Component.GetComponent<T>() , [Component.TryGetComponent\(Type, out Component\)](#) ,
[Component.TryGetComponent<T>\(out T\)](#) , [Component.GetComponent\(string\)](#) ,
[Component.GetComponentInChildren\(Type, bool\)](#) , [Component.GetComponentInChildren\(Type\)](#) ,
[Component.GetComponentInChildren<T>\(bool\)](#) , Component.GetComponentInChildren<T>() ,
[Component.GetComponentsInChildren\(Type, bool\)](#) , [Component.GetComponentsInChildren\(Type\)](#) ,
[Component.GetComponentsInChildren<T>\(bool\)](#) ,
[Component.GetComponentsInChildren<T>\(bool, List<T>\)](#) ,
Component.GetComponentsInChildren<T>() , [Component.GetComponentsInChildren<T>\(List<T>\)](#) ,
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,
[Component.GetComponentInParent<T>\(bool\)](#) , [Component.GetComponentInParent<T>\(\)](#) ,
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,
[Component.GetComponentsInParent<T>\(bool\)](#) ,
[Component.GetComponentsInParent<T>\(bool, List<T>\)](#) , Component.GetComponentsInParent<T>() ,
[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,
[Component.GetComponents<T>\(List<T>\)](#) , [Component.GetComponents<T>\(\)](#) ,
Component.GetComponentIndex() , [Component.CompareTag\(string\)](#) ,

[Component.CompareTag\(TagHandle\)](#) ,
[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,
[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,
[Component.SendMessage\(string, SendMessageOptions\)](#) ,
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,
Component.gameObject , Component.tag , [Object.GetInstanceID\(\)](#) , [Object.GetHashCode\(\)](#) ,
[Object.Equals\(object\)](#) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , [Object.InstantiateAsync<T>\(T, int\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Vector3, Quaternion\)](#) ,
[Object.InstantiateAsync<T>\(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform, Vector3, Quaternion\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform, Vector3, Quaternion, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, InstantiateParameters, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, int, InstantiateParameters, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, Vector3, Quaternion, InstantiateParameters, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Vector3, Quaternion, InstantiateParameters, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>, InstantiateParameters, CancellationToken\)](#) ,
Object.Instantiate(Object, Vector3, Quaternion) ,
[Object.Instantiate\(Object, Vector3, Quaternion, Transform\)](#) , Object.Instantiate(Object) ,
[Object.Instantiate\(Object, Scene\)](#) , Object.Instantiate<T>(T, InstantiateParameters) ,
Object.Instantiate<T>(T, Vector3, Quaternion, InstantiateParameters) ,
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,

[Object.DontDestroyOnLoad\(Object\)](#) , [Object.DestroyObject\(Object, float\)](#) ,
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>()
[Object.FindObjectsByType<T>\(FindObjectsSortMode\)](#) , [Object.FindObjectsOfType<T>\(bool\)](#) ,
[Object.FindObjectsByType<T>\(FindObjectsInactive, FindObjectsSortMode\)](#) ,
[Object.FindObjectOfType<T>\(\)](#) , [Object.FindObjectOfType<T>\(bool\)](#) ,
[Object.FindFirstObjectByType<T>\(\)](#) , [Object.FindAnyObjectByType<T>\(\)](#) ,
[Object.FindFirstObjectByType<T>\(FindObjectsInactive\)](#) ,
[Object.FindAnyObjectByType<T>\(FindObjectsInactive\)](#) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , [Object.ToString\(\)](#) , Object.name ,
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#)

Fields

OnConnectionStateChangedEvent

UnityEvent that is fired when scope connection states are changed.

```
public UnityEvent OnConnectionStateChangedEvent
```

Field Value

[UnityEvent](#)

Properties

Instance

Return the instance of LimosaProtocol

```
public static LimosaProtocol Instance { get; }
```

Property Value

IsServiceBound

Whether the service in AAR is bound or not. Before service is bound, calling the following methods will be ignore : Vibrate(string[], int) VibrateToAll(int) GetScopes() GetScopeInfo(string) GetBatteryLevel(string)

Also see: OnServiceBindStateChanged

```
public bool IsServiceBound { get; }
```

Property Value

[bool](#)

ScopeConnectionState

Dictionary of scope connection state at that point in time. Key: Scope address Value: Connection state
(true: connected, false: not connected)

```
public IReadOnlyDictionary<string, bool> ScopeConnectionState { get; }
```

Property Value

[IReadOnlyDictionary](#)<[string](#), [bool](#)>

ScopeConvertedData

Dictionary of scope converted data at that point in time. Key: Scope address Value: ConvertedData

```
public IReadOnlyDictionary<string, ConvertedData> ScopeConvertedData { get; }
```

Property Value

[IReadOnlyDictionary](#)<[string](#), [ConvertedData](#)>

ScopeOffset

Dictionary of scope converted offset converted data at that point in time. Key: Scope address Value: Converted offset data

```
public IReadOnlyDictionary<string, ConvertedData> ScopeOffset { get; }
```

Property Value

[IReadOnlyDictionary<string, ConvertedData>](#)

ScopeRawData

Dictionary of scope raw data at that point in time. Key: Scope address Value: Raw data

```
public IReadOnlyDictionary<string, RawData> ScopeRawData { get; }
```

Property Value

[IReadOnlyDictionary<string, RawData>](#)

ScopeRawOffset

Dictionary of scope offset raw data at that point in time. Key: Scope address Value: Raw offset data

```
public IReadOnlyDictionary<string, RawData> ScopeRawOffset { get; }
```

Property Value

[IReadOnlyDictionary<string, RawData>](#)

Methods

GetBatteryLevel(string)

Get Scope's current battery level Also see: OnServiceBindStateChanged

```
public virtual int GetBatteryLevel(string targetAddress)
```

Parameters

targetAddress [string](#)

Target Scope's address

Returns

[int](#)

Battery leve value. The value is the ratio of the remaining battery charge to the full charge and is a percentage value from 0 to 100.

See Also

[OnBatteryLevelChanged\(string\)](#)

GetScopeInfo(string)

Get Scope's static information

```
public virtual ScopeInfo GetScopeInfo(string targetAddress)
```

Parameters

targetAddress [string](#)

Returns

[ScopeInfo](#)

ScopeInfo instance that contains Scope's static information.

GetScopes()

Get all paired Scopes.

```
public virtual BondedScope GetScopes()
```

Returns

BondedScope

BondedScope instance that contains all paired Scope's information.

GoBackToLimosa()

Go back to Yo OS.

```
public virtual void GoBackToLimosa()
```

Remarks

This method performs just going back to Yo OS. Please handle the termination of the application yourself.

OnBatteryLevelChanged(string)

Callback method called when scope battery level is changed. The value is the ratio of the remaining battery charge to the full charge and is a percentage value from 0 to 100.

```
protected virtual void OnBatteryLevelChanged(string data)
```

Parameters

data string ↗

Json text serialized with IntegerChanged type information

OnButtonsStateChanged(string)

Callback method called when scope button state is changed (pressed or released).

```
protected virtual void OnButtonsStateChanged(string data)
```

Parameters

data [string](#)

Json text serialized with ButtonStateChanged type information

OnConnected(string)

Callback method called when Scope is connected

```
protected virtual void OnConnected(string data)
```

Parameters

data [string](#)

Json text serialized with ScopeAddress type information

OnDataConverted(string)

```
protected virtual void OnDataConverted(string data)
```

Parameters

data [string](#)

OnDisconnected(string)

Callback method called when Scope is disconnected

```
protected virtual void OnDisconnected(string data)
```

Parameters

data [string](#)

Json text serialized with ScopeAddress type information

OnEncoderValueChanged(string)

Callback method called when scope zoom ring value is changed. Clockwise rotation increases value.

```
protected virtual void OnEncoderValueChanged(string data)
```

Parameters

data [string](#)

Json text serialized with IntegerChanged type information

OnInitialized(string)

Callback method called when Scope is connected. This is called once after the scope is connected.

```
protected virtual void OnInitialized(string data)
```

Parameters

data [string](#)

Json text serialized with ScopeAddress type information

OnOffsetReset(string)

Callback method called when scope pointing position is reset.

```
protected virtual void OnOffsetReset(string data)
```

Parameters

data [string](#)

Json text serialized with ScopeOffset type information

OnRawData(string)

Callback method called when scope send new data.

```
protected virtual void OnRawData(string data)
```

Parameters

data [string](#)

Json text serialized with RawData type information

OnServiceBindStateChanged(string)

Callback method called when the service in AAR is bound or unbound. Before service is bound, calling the following methods will be ignore : Vibrate(string[], int) VibrateToAll(int) GetScopes() GetScopeInfo(string) GetBatteryLevel(string)

Also see: OnServiceBindStateChanged

```
protected virtual void OnServiceBindStateChanged(string data)
```

Parameters

data [string](#)

Json text serialized with ServiceBindState type information

OnSignalTargetSet(string)

Callback method called when the service in AAR set signal target.

```
protected virtual void OnSignalTargetSet(string data)
```

Parameters

data [string](#)

Signal Target Name

Vibrate(string[], int)

Vibrate Scope with the selected pattern.

```
public virtual void Vibrate(string[] targetAddress, int pattern)
```

Parameters

targetAddress [string](#)[]

Target Scope's addresses

pattern [int](#)

Vibration pattern.1 ~ 123

VibrateToAll(int)

```
public virtual void VibrateToAll(int pattern)
```

Parameters

pattern [int](#)

Class RawData

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope Raw data

```
[Serializable]  
public class RawData
```

Inheritance

[object](#) ← RawData

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

address

String of Scope address. Also see: ScopeAddress

```
public string address
```

Field Value

[string](#)

button

Button state of Scope. Also see: LimosaProtocol.OnButtonsStateChanged

```
public ButtonState button
```

Field Value

ButtonState

See Also

[OnButtonsStateChanged\(string\)](#)

quaternion

Quaternion for scope orientation

```
public Vector4f quaternion
```

Field Value

[Vector4f](#)

zoom

Value of Scope Zoom ring. Clockwise rotation increases value. Also see:
[LimosaProtocol.OnEncoderValueChanged](#)

```
public int zoom
```

Field Value

[int](#)

See Also

[OnEncoderValueChanged\(string\)](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#) ↗

Class ScopeAddress

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope address

```
[Serializable]  
public class ScopeAddress
```

Inheritance

[object](#) ← ScopeAddress

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

address

String of Scope address

```
public string address
```

Field Value

[string](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#) ↗

Class ScopeButton

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Class to map the Scope button to the control path. This class is set to button objects in VirtualButtons of LimosaProtocol.prefab

```
public class ScopeButton : OnScreenControl
```

Inheritance

[object](#) ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← [OnScreenControl](#) ← [ScopeButton](#)

Inherited Members

[OnScreenControl.SendValueToControl< TValue >\(TValue\)](#) , [OnScreenControl.SentDefaultValueToControl\(\)](#) ,
[OnScreenControl.OnEnable\(\)](#) , [OnScreenControl.OnDisable\(\)](#) , [OnScreenControl.controlPath](#) ,
[OnScreenControl.control](#) , [MonoBehaviour.IsInvoking\(\)](#) , [MonoBehaviour.CancelInvoke\(\)](#) ,
[MonoBehaviour.Invoke\(string, float\)](#) , [MonoBehaviour.InvokeRepeating\(string, float, float\)](#) ,
[MonoBehaviour.CancelInvoke\(string\)](#) , [MonoBehaviour.IsInvoking\(string\)](#) ,
[MonoBehaviour.StartCoroutine\(string\)](#) , [MonoBehaviour.StartCoroutine\(string, object\)](#) ,
[MonoBehaviour.StartCoroutine\(IEnumerator\)](#) , [MonoBehaviour.StartCoroutine_Auto\(IEnumerator\)](#) ,
[MonoBehaviour.StopCoroutine\(IEnumerator\)](#) , [MonoBehaviour.StopCoroutine\(Coroutine\)](#) ,
[MonoBehaviour.StopCoroutine\(string\)](#) , [MonoBehaviour.StopAllCoroutines\(\)](#) ,
[MonoBehaviour.print\(object\)](#) , [MonoBehaviour.destroyCancellationToken](#) ,
[MonoBehaviour.useGUILayout](#) , [MonoBehaviour.didStart](#) , [MonoBehaviour.didAwake](#) ,
[MonoBehaviour.runInEditMode](#) , [Behaviour.enabled](#) , [Behaviour.isActiveAndEnabled](#) ,
[Component.GetComponent\(Type\)](#) , [Component.GetComponent< T >\(\)](#) ,
[Component.TryGetComponent\(Type, out Component\)](#) , [Component.TryGetComponent< T >\(out T\)](#) ,
[Component.GetComponent\(string\)](#) , [Component.GetComponentInChildren\(Type, bool\)](#) ,
[Component.GetComponentInChildren\(Type\)](#) , [Component.GetComponentInChildren< T >\(bool\)](#) ,
[Component.GetComponentInChildren< T >\(\)](#) , [Component.GetComponentsInChildren\(Type, bool\)](#) ,
[Component.GetComponentsInChildren\(Type\)](#) , [Component.GetComponentsInChildren< T >\(bool\)](#) ,
[Component.GetComponentsInChildren< T >\(bool, List< T >\)](#) ,
[Component.GetComponentsInChildren< T >\(\)](#) , [Component.GetComponentsInChildren< T >\(List< T >\)](#) ,
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,
[Component.GetComponentInParent< T >\(bool\)](#) , [Component.GetComponentInParent< T >\(\)](#) ,
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,
[Component.GetComponentsInParent< T >\(bool\)](#) ,

[Component.GetComponentInParent<T>\(bool, List<T>\)](#) , Component.GetComponentInParent<T>() ,
[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,
[Component.GetComponents<T>\(List<T>\)](#) , [Component.GetComponents<T>\(\)](#) ,
Component.GetComponentIndex() , [Component.CompareTag\(string\)](#) ,
[Component.CompareTag\(TagHandle\)](#) ,
[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,
[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,
[Component.SendMessage\(string, SendMessageOptions\)](#) ,
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,
Component.gameObject , Component.tag , [Object.GetInstanceID\(\)](#) , [Object.GetHashCode\(\)](#) ,
[Object.Equals\(object\)](#) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , [Object.InstantiateAsync<T>\(T, int\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Vector3, Quaternion\)](#) ,
[Object.InstantiateAsync<T>\(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform, Vector3, Quaternion\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform, Vector3, Quaternion, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, InstantiateParameters, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, int, InstantiateParameters, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, Vector3, Quaternion, InstantiateParameters, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, int, Vector3, Quaternion, InstantiateParameters, CancellationToken\)](#) ,
[Object.InstantiateAsync<T>\(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>, InstantiateParameters, CancellationToken\)](#) ,
Object.Instantiate(Object, Vector3, Quaternion) ,
[Object.Instantiate\(Object, Vector3, Quaternion, Transform\)](#) , Object.Instantiate(Object) ,
[Object.Instantiate\(Object, Scene\)](#) , Object.Instantiate<T>(T, InstantiateParameters) ,
Object.Instantiate<T>(T, Vector3, Quaternion, InstantiateParameters) ,
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,

[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,
[Object.DontDestroyOnLoad\(Object\)](#) , [Object.DestroyObject\(Object, float\)](#) ,
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>()
[Object.FindObjectsByType<T>\(FindObjectsSortMode\)](#) , [Object.FindObjectsOfType<T>\(bool\)](#) ,
[Object.FindObjectsByType<T>\(FindObjectsInactive, FindObjectsSortMode\)](#) ,
[Object.FindObjectOfType<T>\(\)](#) , [Object.FindObjectOfType<T>\(bool\)](#) ,
[Object.FindFirstObjectByType<T>\(\)](#) , [Object.FindAnyObjectByType<T>\(\)](#) ,
[Object.FindFirstObjectByType<T>\(FindObjectsInactive\)](#) ,
[Object.FindAnyObjectByType<T>\(FindObjectsInactive\)](#) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , [Object.ToString\(\)](#) , Object.name ,
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#)

Properties

controlPathInternal

```
protected override string controlPathInternal { get; set; }
```

Property Value

[string](#)

Methods

OnPressed()

Call back method that is called when the button is pressed

```
public void OnPressed()
```

OnReleased()

Call back method that is called when the button is released

```
public void OnReleased()
```

Class ScopeInfo

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope static information.

```
[Serializable]
public class ScopeInfo
```

Inheritance

[object](#) ← ScopeInfo

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

firmwareRev

String of firmware revision in software revision

```
public string firmwareRev
```

Field Value

[string](#)

hardwareRev

String of hardware revision

```
public string hardwareRev
```

Field Value

[string](#)

manufacturer

String of manufacturer

```
public string manufacturer
```

Field Value

[string](#)

modelNumber

String of model number

```
public string modelNumber
```

Field Value

[string](#)

pnpId

String of PNP ID◆iPlug and Play ID◆j

```
public string pnpId
```

Field Value

[string](#)

serialNumber

String of serial number

```
public string serialNumber
```

Field Value

[string](#)

softwareRev

String of application revision in software revision

```
public string softwareRev
```

Field Value

[string](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#)

Class ScopeOffset

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope offset Elements are similar to elements of type ConvertedData. see: ConvertedData

```
[Serializable]  
public class ScopeOffset
```

Inheritance

[object](#) ← ScopeOffset

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

address

String of Scope address. Also see: ScopeAddress

```
public string address
```

Field Value

[string](#)

byUser

Whether reset is performed by user or not 0.0~1080.0

```
public bool byUser
```

Field Value

[bool](#)

pitch

Pitch degree of scope orientation

```
public float pitch
```

Field Value

[float](#)

roll

Roll degree of scope orientation

```
public float roll
```

Field Value

[float](#)

x

X-axis value of pointing position on the Yo screen 0.0~1080.0

```
public float x
```

Field Value

[float](#)

y

Y-axis value of pointing position on the Yo screen 0.0~1920.0

```
public float y
```

Field Value

[float](#)

yaw

Yaw degree of scope orientation

```
public float yaw
```

Field Value

[float](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#)

Class ServiceBindState

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type for state indicating whether the service in AAR is bound or not.

```
[Serializable]  
public class ServiceBindState
```

Inheritance

[object](#) ← ServiceBindState

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

state

Whether the service in AAR is bound or not. true: Bound false: Not bound

```
public bool state
```

Field Value

[bool](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#) ↗

Class Vector4f

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Vector4f This is used for quaternion. see: RawData.quaternion

```
[Serializable]  
public class Vector4f
```

Inheritance

[object](#) ← Vector4f

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

W

```
public float w
```

Field Value

[float](#)

X

```
public float x
```

Field Value

[float](#)

y

```
public float y
```

Field Value

[float](#)

z

```
public float z
```

Field Value

[float](#)

Methods

ToString()

```
public override string ToString()
```

Returns

[string](#)

"サンプルデモ"の目次

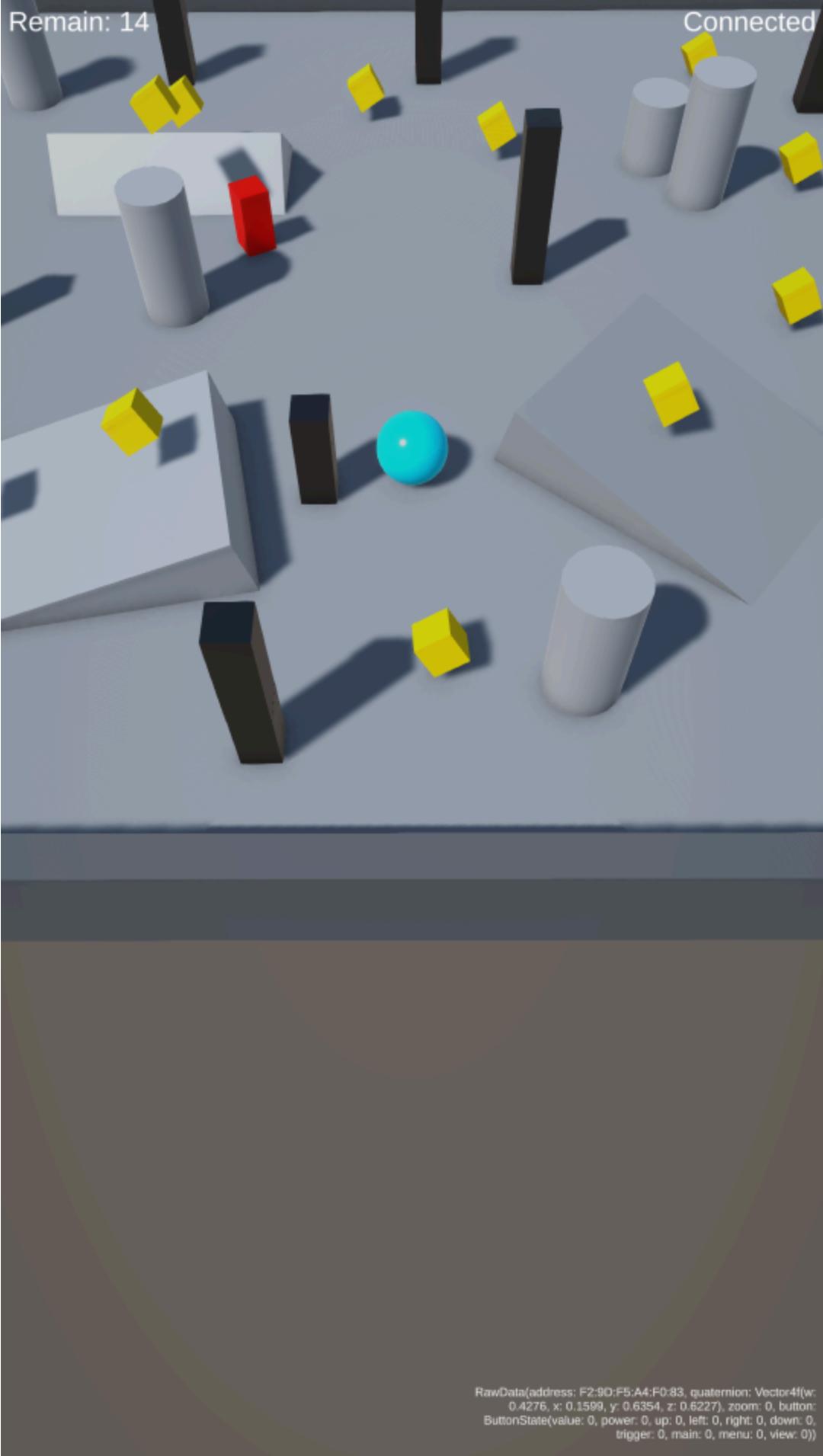
- [サンプルデモ Roll-a-Ball Game with Limosa](#)
 - [動かし方](#)
 - [使用しているSDKの機能](#)
 - [1. Scopeのボタン操作の検知 : プレイヤーをScopeのボタンでコントロールできるようにする](#)
 - [2. Scopeのポインティング位置、向き、距離センター値の取得 : 地面の角度をScopeの向きでコントロールできるようにする](#)
 - [3. Scopeのズームリングデータの取得 : ズームリング クオータニオン形式のScopeの向き、ボタン状態の画面出力](#)
 - [4. Scopeの動的情報の取得 \(接続状態、アドレス、バッテリー残量\) : 接続状態の画面表示や、バッテリー残量のログ出力](#)
 - [LimosaProtocol.ScopeConnectionStateプロパティを使用](#)
 - [LimosaProtocol.GetScopesメソッドを使用](#)
 - [LimosaProtocol.OnConnectionStateChangedEventプロパティにコールバックメソッドを追加する](#)
 - [5. Scopeの静的情報の取得\(本体型番、シリアルナンバー、FWバージョン、PnPID\)](#)
 - [6. Scopeへの振動指示 : ボタンを押した際にScopeを振動させる](#)
 - [7. Unityアプリの終了を本体に伝える : ゲーム終了時にYo本体機能に戻る](#)

サンプルデモ Roll-a-Ball Game with Limosa

Unityの一般的なサンプルデモ [Roll-a-Ball Game](#) をSDKを用いてYoで遊べるようにしたデモです。サンプルデモプロジェクトのScenes/MiniGameに実装されています。

NOTE

Roll-a-Ball Gameの内容や実装方法については、Unityの公式ドキュメント[Roll-a-Ball Game](#) を参照してください。



動かし方

1. rollaballwithlimosa.zipを展開し、Unity.HubからUnityプロジェクトとして読み込む
2. [ビルト設定](#)を実施して、"Build And Run"

(i) NOTE

"Build Profile"の"Configuration" -> "Active Input Handling"は"Both"を設定してください。これに対して警告ダイアログが出る場合がありますが、無視してください。

(i) NOTE

デモは [SDKのインストール](#) はすでに実施済みの状態になっています。

使用しているSDKの機能

このデモではSDKの次の機能を使用しています。

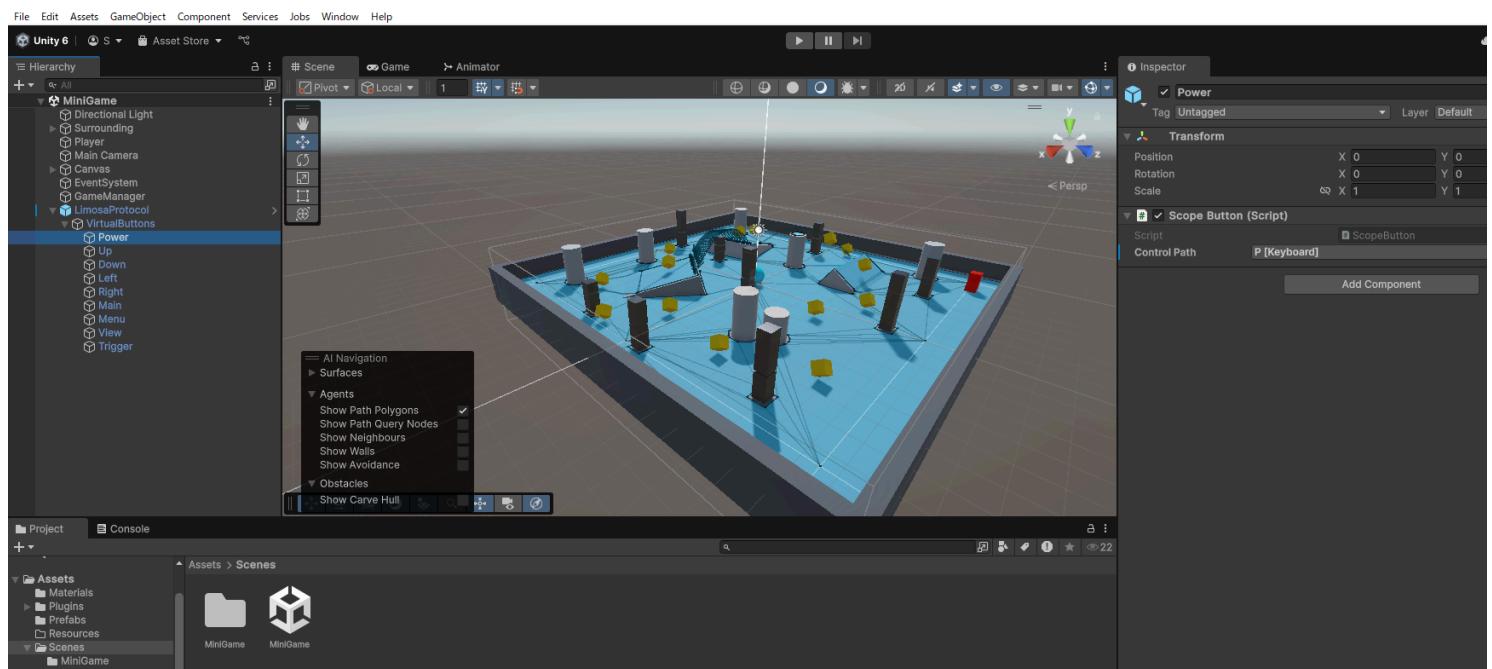
1. [Scopeのボタン操作の検知](#) : プレイヤーをScopeのボタンでコントロールできるようにする
2. [Scopeのポインティング位置、向き、距離センター値の取得](#) : 地面の角度をScopeの向きでコントロールできるようにする
3. [Scopeのズームリングデータの取得](#) : ズームリング、クオータニオン形式のScopeの向き、ボタン状態の画面出力
4. [Scopeの動的情報の取得（接続状態、アドレス、バッテリー残量）](#) : 接続状態の画面表示や、バッテリー残量のログ出力
5. [Scopeの静的情報の取得\(本体型番、シリアルナンバー、FWバージョン、PnPID\)](#) : Scope情報のログ出力
6. [Scopeへの振動指示](#) : ボタンを押した際にScopeを振動させる
7. [Unityアプリの終了を本体に伝える](#) : ゲーム終了時にYo本体機能に戻る

1. [Scopeのボタン操作の検知](#) : プレイヤーをScopeのボタンでコントロールできるようにする

LimosaProtocol.prefabをHierarchyウィンドウに追加し、その子オブジェクトのVirtualButtonsの子オブジェクトでScopeの各ボタンをInputControlにマッピングしています。マッピング先は次のようになっています。

オブジェクト	InputControl
Power	P [Keyboard]
Up	Stick/Up [Joystick]

オブジェクト	InputControl
Down	Stick/Down [Joystick]
Left	Stick/Left [Joystick]
Right	Stick/Right [Joystick]
Main	Space [Keyboard]
Menu	M [Keyboard]
View	V [Keyboard]
Trigger	Q [Keyboard]



2. Scopeのポインティング位置、向き、距離センター値の取得 : 地面の角度をScopeの向きでコントロールできるようにする

Scripts/SurroundingController.csで使用しています。

```
void FixedUpdate()
{
    // Slightly rotate the floor with Scope's gyroscope
    var offset = LimosaProtocol.Instance.ScopeOffset.FirstOrDefault().Value;
    var data = LimosaProtocol.Instance.ScopeConvertedData.FirstOrDefault().Value;
    if (data != null && offset != null)
    {
        ...
    }
}
```

```

        var pitch = Mathf.Rad2Deg * (data.pitch - offset.pitch);
        var roll = Mathf.Rad2Deg * (data.roll - offset.roll);
        var yaw = Mathf.Rad2Deg * (data.yaw - offset.yaw);
        var origin = new Vector3(pitch, roll, yaw);
        var z = -Mathf.Clamp(pitch * AngleFactorPitch, -AngleLimit, AngleLimit);
        var y = 0f; // -Mathf.Clamp(yaw * AngleFactor, -AngleLimit, AngleLimit);
    // fixed
        var x = -Mathf.Clamp(roll * AngleFactorRoll, -AngleLimit, AngleLimit);
        var rotation = new Vector3(z, y, x);
        rb.MoveRotation(Quaternion.Euler(rotation));
        // Debug.Log($"gameObject.transform.eulerAngle: {origin} -> {rotation}");
    }
}

```

[LimosaProtocol.ScopeConvertedData](#)プロパティから取得したその時点のScopeの[ConvertedData](#)からScopeの角度([ConvertedData.pitch](#)、[ConvertedData.yaw](#)、[ConvertedData.roll](#))を取得し、それを基に地面を回転しています。

```
var data = LimosaProtocol.Instance.ScopeConvertedData.FirstOrDefault().Value;
```

Scopeの「ポインティング位置のリセット」機能にも対応できるように、[3-1 Offset](#)のオフセット値も取得し、[Converted Data](#)とオフセットとの差分から角度を算出しています。

オフセット値の取得方法は[3-1-1 LimosaProtocol.ScopeOffset](#)、[LimosaProtocol.ScopeRawOffset](#) プロパティからの[Offsetの取得](#)を使用しています。

```
var offset = LimosaProtocol.Instance.ScopeOffset.FirstOrDefault().Value;
```

3. [Scopeのズームリングデータの取得](#)：ズームリング、クオータニオン形式のScopeの向き、ボタン状態の画面出力

ScopeRawDataPrinter.csのUpdateメソッドで、接続しているScopeのズームリング、クオータニオン形式のScopeの向き、ボタン状態の情報を持った[RawData](#)を画面に出力しています。

```

void Update()
{
    var data = Limosa.ScopeRawData.FirstOrDefault().Value;
    textMeshProUGUI.text = (data != null) ? data.ToString() : "";
}

```

4. Scopeの動的情報の取得（接続状態、アドレス、バッテリー残量）：接続状態の画面表示や、バッテリー残量のログ出力

GameManager.csのUpdateメソッドで「Powerボタンを押すとバッテリー残量をログに出力」の機能が実装されています。

バッテリー残量のログ出力は次のコードで実装されています。[Limosa.ScopeConnectionState](#)で取得した接続中Scopeのアドレスを用いて、Limosa.GetBatteryLevel(how_to_use.html#1-4-2-バッテリー残量の取得)メソッドでバッテリー残量を取得しています。

```
else if (current?.pKey?.wasReleasedThisFrame == true)
{
    var address = Limosa.ScopeConnectionState.FirstOrDefault()?.Key;
    if (address != null)
    {
        Limosa.Vibrate(new string[]{address}, 123);
        Debug.Log($"battery: {Limosa.GetBatteryLevel(address)}");
    }
}
```

接続状態は次の3つの方法で取得されています。

[LimosaProtocol.ScopeConnectionStateプロパティを使用](#)

上記のLimosa.GetBatteryLevelメソッドの引数に設定するScopeアドレスを取得する方法です。

[LimosaProtocol.GetScopesメソッドを使用](#)

GameManager.csのUpdateメソッドの次の処理で使われています。[BondedScope](#)インスタンスで接続状態を一括で取得しています。

```
Debug.Log($"GetScopes: {Limosa.GetScopes()}");
```

[LimosaProtocol.OnConnectionStateChangedEventプロパティにコールバックメソッドを追加する](#)

GameManager.csのSetListenersメソッドで次のように[LimosaProtocol.OnConnectionStateChangedEvent](#)へのコールバックメソッドの追加、削除が実装されています。Scopeの新規接続、接続解除のタイミングで呼び出されるコールバックメソッドで処理しています。

```
private void SetListeners(bool state)
{
    if (state)
    {
        Limosa.OnConnectionStateChangedEvent.AddListener(OnScopeStateChanged);
```

```

        player.GetComponent<PlayerController>
    () .onTouchedEvent.AddListener(OnPlayerTouched);
    }
    else
    {
        Limosa.OnConnectionStateChangedEvent.RemoveAllListeners();
        player.GetComponent<PlayerController>().onTouchedEvent.RemoveAllListeners();
    }
}

// Check Connection state changed & Show on screen.
private void OnScopeStateChanged()
{
    var state = Limosa.ScopeConnectionState.FirstOrDefault();
    Debug.Log($"OnScopeStateChanged. {state.Key}: {state.Value}");
    stateText.text = state.Value ? "Connected" : "Disconnected";
}

```

5. Scopeの静的情報の取得(本体型番、シリアルナンバー、FWバージョン、PnPID)

GameManager.csのUpdateメソッドの次の処理で使われています。[Limosa.ScopeConnectionState](#)で取得した接続中Scopeのアドレスを用いて、[Limosa.GetScopeInfo\(how_to_use.html#1-5-scope\)](#)の静的情報の取得本体型番シリアルナンバーfwバージョンpnpid)メソッドでScopeの静的情報を取得、ログに出力しています。

```

else if (current?.mKey?.wasPressedThisFrame == true)
{
    Debug.Log($"GetScopes: {Limosa.GetScopes()}");
    var address = Limosa.ScopeConnectionState.FirstOrDefault().Key;
    if (address != null)
        Debug.Log($"GetScopeInfo: {Limosa.GetScopeInfo(address)}");
}

```

6. Scopeへの振動指示：ボタンを押した際にScopeを振動させる

GameManager.csのUpdateメソッドで「Power、風景ボタンを押すとScopeが振動」の機能が実装されています。Powerボタンが押された際は [LimosaProtocol.Vibrate](#)、風景ボタンが押された際は [LimosaProtocol.VibrateToAll](#)でScopeに振動を指示しています。

```

else if (current?.pKey?.wasReleasedThisFrame == true)
{
    var address = Limosa.ScopeConnectionState.FirstOrDefault().Key;
    if (address != null)
    {

```

```
        Limosa.Vibrate(new string[]{address}, 123);
        Debug.Log($"battery: {Limosa.GetBatteryLevel(address)}");
    }
}
else if (current?.vKey?.wasReleasedThisFrame == true)
    Limosa.VibrateToAll(123);
```

7. Unityアプリの終了を本体に伝える：ゲーム終了時にYo本体機能に戻る

GameManager.csのUpdateメソッドで「トリガーボタンを押すとゲーム終了」の機能が実装されています。[Limosa Protocol.GoBackToLimosa](#)メソッドでゲームの終了をYo本体に伝えた上で、ゲームの終了処理を実施しています。

```
else if (current?.qKey?.wasReleasedThisFrame == true)
{
    Limosa.GoBackToLimosa();
    QuitGame();
}
```

i NOTE

ゲームの終了処理はゲーム自身で行う必要があります。