MotionCapture

Hi5_Unreal_Plugin_0_9_8_655_1

Generated by Doxygen 1.8.12

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

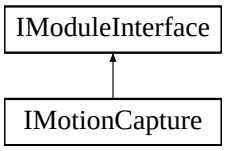
Chapter 4

Class Documentation

4.1 IMotionCapture Class Reference

`#include <IMotionCaptureModule.h>`

Inheritance diagram for IMotionCapture:



Public Member Functions

- virtual void OnReceiveGloveData (const FString &AvartarName, const TArray< uint8 > &Data)=0
- virtual bool GetLocalGloveData (EGloveMod Glove, const FString &AvatarName, TArray< uint8 > &Data)=0
- virtual void CacheHybridDataCaliProgress (int32 Pose, int32 percent)=0
- virtual void GetCachedHybridDataCaliProgress (ECalibrationPose &Pose, int32 &Percent)=0
- virtual bool GetParsedLocalJointsData (EGloveMod Glove, TArray< FVector > &Positions, TArray< FRotator > &Orientations)=0
- virtual bool GetLocalJointData (EGloveMod Glove, EMCBones::Type Bone, FVector &Position, FRotator &Orientation)=0

Static Public Member Functions

- static IMotionCapture * **GetPtr** ()
- static IMotionCapture & Get ()
- static bool IsAvailable ()

4.1.1 Detailed Description

The public interface to this module. In most cases, this interface is only public to sibling modules within this plugin.

4.1.2 Member Function Documentation

4.1.2.1 CacheHybridDataCaliProgress()

```
virtual void IMotionCapture::CacheHybridDataCaliProgress (
            int32 Pose,
            int32 percent )  [pure virtual]
```

Cache calibration progress value which from hybrid data plugin

4.1.2.2 Get()

```
static IMotionCapture& IMotionCapture::Get ( )  [inline], [static]
```

Singleton-like access to this module's interface. This is just for convenience! Beware of calling this during the shutdown phase, though. Your module might have been unloaded already.

Returns

Returns singleton instance, loading the module on demand if needed

4.1.2.3 GetCachedHybridDataCaliProgress()

```
virtual void IMotionCapture::GetCachedHybridDataCaliProgress (
            ECalibrationPose & Pose,
            int32 & Percent )  [pure virtual]
```

Get cached calibration progress

4.1.2.4 GetLocalGloveData()

```
virtual bool IMotionCapture::GetLocalGloveData (
            EGloveMod Glove,
            const FString & AvatarName,
            TArray< uint8 > & Data )  [pure virtual]
```

The return Data is type: HI5BVHData

4.1.2.5 GetLocalJointData()

```
virtual bool IMotionCapture::GetLocalJointData (
            EGloveMod Glove,
            EMCBones::Type Bone,
            FVector & Position,
            FRotator & Orientation )  [pure virtual]
```

Get the specified bone's local position and rotation.

4.1.2.6 GetParsedLocalJointsData()

```
virtual bool IMotionCapture::GetParsedLocalJointsData (
            EGloveMod Glove,
            TArray< FVector > & Positions,
            TArray< FRotator > & Orientations )  [pure virtual]
```

Get all bones' gesture data in UE4 coordinate system

4.1.2.7 IsAvailable()

```
static bool IMotionCapture::IsAvailable ( )  [inline], [static]
```

Checks to see if this module is loaded and ready. It is only valid to call Get() if IsAvailable() returns true.

Returns

True if the module is loaded and ready to use

4.1.2.8 OnReceiveGloveData()

```
virtual void IMotionCapture::OnReceiveGloveData (
            const FString & AvartarName,
            const TArray< uint8 > & Data )  [pure virtual]
```

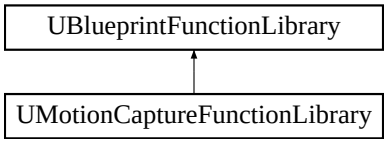
The return Data is type: GloveBVHData

The documentation for this class was generated from the following file:

- IMotionCaptureModule.h

4.2 UMotionCaptureFunctionLibrary Class Reference

Inheritance diagram for UMotionCaptureFunctionLibrary:



Static Public Member Functions

- static bool StartMocapService (EGlovePositionSource PosSrc=EGlovePositionSource::VivePosition, bool ReadLocal=true)
- static void StopMocapService ()
- static void CalibrationGlove (ECalibrationPose Pose, int32 TimeOut=5000)
- static int32 GetCalibrationProgress ()
- static EGlovePositionSource GetGlovePositionSource ()
- static bool LoadCalibrationData ()
- static bool SaveCalibrationData ()
- static void VibrateLeftGlove (const int GloveTimeSpan)
- static void VibrateRightGlove (const int GloveTimeSpan)
- static void VibrateGloves (const int LeftGloveTimeSpan, const int RightGloveTimeSpan)
- static bool GetOptiDeviceSN (int32 Id, FString &SN)
- static bool GetOptiDeviceBindState (EGloveMod Hand, FString &DeviceSN)
- static void GetGlovePowerLevel (EGlovePowerLevel &LeftGlove, EGlovePowerLevel &RightGlove)
- static void GetGloveMagneticed (EGloveMagneticedState &LeftGlove, EGloveMagneticedState &RightGlove)
- static bool IsGloveAvailable (EGloveMod Mod)
- static bool IsDongleAvailable ()
- static bool GetTrackedDeviceDataInUESpace (int DeviceId, FVector &Position, FRotator &Rotation)
- static int GetLeftTrackerId ()
- static int GetRightTrackerId ()
- static bool GetParsedLocalJointsData (EGloveMod Glove, TArray< FVector > &Positions, TArray< FRotator > &Orientations)
- static bool GetLocalJointData (EGloveMod Glove, EMCBones::Type Bone, FVector &Position, FRotator &Orientation)

4.2.1 Member Function Documentation

4.2.1.1 CalibrationGlove()

```
static void UMotionCaptureFunctionLibrary::CalibrationGlove (
            ECalibrationPose Pose,
            int32 TimeOut = 5000 )  [static]
```

Calibrate Hi5 glove hardware. In this version, you can use only B and P pose.

Parameters

in	*Pose*	The motion need to calibrate. May be one of [B, P].
in	*TimeOut*	(Unit: ms). If the calibration hasn't finished in TimeOut milliseconds, the calibration will stop.

Note

For B-Pose, the TimeOut longer than 5 seconds is recommended. And for P-Pose, the recommended TimeOut is 3 seconds. \ If no Hi5 glove is power on, the function do nothing.

4.2.1.2 GetCalibrationProgress()

```
static int32 UMotionCaptureFunctionLibrary::GetCalibrationProgress ( )  [static]
```

Query the progress of calibration.

Returns

The progress of calibration. The value returned is in range [0,100].

Note

100 represent that the calibration of current pose complete.

4.2.1.3 GetGloveMagneticed()

```
static void UMotionCaptureFunctionLibrary::GetGloveMagneticed (
            EGloveMagneticedState & LeftGlove,
            EGloveMagneticedState & RightGlove )  [static]
```

Get glove magnetization state.

Parameters

out	*LeftGlove*	Left glove magnetization state.
out	*RightGlove*	Right glove magnetization state.

4.2.1.4 GetGlovePositionSource()

```
static EGlovePositionSource UMotionCaptureFunctionLibrary::GetGlovePositionSource ( )  [static]
```

Get the source type of the location data of hand.

Returns

Vive | Alice | Other

4.2.1.5 GetGlovePowerLevel()

```
static void UMotionCaptureFunctionLibrary::GetGlovePowerLevel (
            EGlovePowerLevel & LeftGlove,
            EGlovePowerLevel & RightGlove )  [static]
```

Get glove power level.

Parameters

out	*LeftGlove*	Left glove power level.
out	*RightGlove*	Right glove power level.

See also

EGlovePowerLevel

4.2.1.6 GetLeftTrackerId()

```
static int UMotionCaptureFunctionLibrary::GetLeftTrackerId ( )  [static]
```

Get the ID of the tracker binded to left hand.

Return values

-1	The returned device id is invalid.
other	The returned device id is valid.

Note

The function is meaningless when the glove position source is Alice.

4.2.1.7 GetLocalJointData()

```
static bool UMotionCaptureFunctionLibrary::GetLocalJointData (
            EGloveMod Glove,
            EMCBones::Type Bone,
            FVector & Position,
            FRotator & Orientation )  [static]
```

Get local position and rotation of specified bone.

Parameters

in	*Glove*	Left hand or Right hand.
in	*Bone*	Specify bone.
out	*Position*	The local position.
out	*Orientation*	The local rotation.

Return values

true	Succeeded
false	Failed

Note

The ForeArm joint's data is meaningless, don't use it.

4.2.1.8 GetOptiDeviceBindState()

```
static bool UMotionCaptureFunctionLibrary::GetOptiDeviceBindState (
            EGloveMod Hand,
            FString & DeviceSN )  [static]
```

Get current binding relationship between glove and optical device.

Parameters

in	*Hand*	Left or Right hand.
out	*DeviceSN*	The serial number of optical device which be bound to Hand.

Return values

true	The hand and optical device are bound together.
false	There isn't any optical device bound to Hand.

Warning

Hand must be left or right.

4.2.1.9 GetOptiDeviceSN()

```
static bool UMotionCaptureFunctionLibrary::GetOptiDeviceSN (
            int32 Id,
            FString & SN )  [static]
```

Get the serial number of optical device via its Id.

Parameters

in	*Id*	The id of optical device.
out	*SN*	Receive the returned serial number.

Return values

true	Succeeded.
false	Failed.

4.2.1.10 GetParsedLocalJointsData()

```
static bool UMotionCaptureFunctionLibrary::GetParsedLocalJointsData (
            EGloveMod Glove,
            TArray< FVector > & Positions,
            TArray< FRotator > & Orientations )  [static]
```

Get hand local gesture data in unreal engine space.

Parameters

in	*Glove*	Left hand or Right hand.
out	*Positions*	Joints' position.
out	*Orientations*	Joints' orientation.

Return values

true	Succeeded
false	Failed.

Note

If you want to get RightHandThumb1's local Position and Orientation, you can do as follow:

```
TArray<FVector> Positions;
TArray<FRotator> Orientations;
GetParsedLocalJointsData(EGloveMod::GlM_LeftGlove, Positions, Orientations);
FVector pos = Positions[EMCBones::RightHandThumb1];
FRotator rot = Orientations[EMCBones::RightHandThumb1];
```

In Blueprint, the array index above can be constructed by [Literal enum EMCBones] node. The ForeArm joint's data is meaningless, don't use it.

Warning

Glove must be left or right.

4.2.1.11 GetRightTrackerId()

```
static int UMotionCaptureFunctionLibrary::GetRightTrackerId ( )  [static]
```

Get the ID of the tracker binded to right hand.

Return values

-1	The returned device id is invalid.
other	The returned device id is valid.

Note

The function is meaningless when the glove position source is Alice.

4.2.1.12 GetTrackedDeviceDataInUESpace()

```
static bool UMotionCaptureFunctionLibrary::GetTrackedDeviceDataInUESpace (
            int DeviceId,
            FVector & Position,
            FRotator & Rotation )  [static]
```

Get the specific vive tracker or controller's position and rotation in UE space.

Parameters

in	*DeviceId*	The id of vive tracker or controller.
out	*Position*	Device position
out	*Rotation*	Device rotation

Return values

true	The device is connected and normal tracking.
false	The device is not connected or abnormal tracking.

4.2.1.13 IsDongleAvailable()

```
static bool UMotionCaptureFunctionLibrary::IsDongleAvailable ( )  [static]
```

Check if the dongle is available.

Return values

true	Available.
false	Unavailable.

4.2.1.14 IsGloveAvailable()

```
static bool UMotionCaptureFunctionLibrary::IsGloveAvailable (
            EGloveMod Mod )  [static]
```

Check if the specific glove is available.

Parameters

in	*Mod*	Glove type.

Return values

true	Available.
false	Unavailable.

Note

When Mod is GIM_BothGloves, the returned value will be true if and only if both gloves available.

4.2.1.15 LoadCalibrationData()

```
static bool UMotionCaptureFunctionLibrary::LoadCalibrationData ( )  [static]
```

Load calibration data from default files.

Return values

true	Succeeded.
false	Failed.

Note

The default files are CalibrationData and OpticalDeviceBindInfo.xml which stored in user-based folder: $FO↩
LDERID_RoamingAppData/Hi5. Eg: C:/Users/your_name/AppData/Roaming/Hi5/CalibrationData.

4.2.1.16 SaveCalibrationData()

```
static bool UMotionCaptureFunctionLibrary::SaveCalibrationData ( )  [static]
```

Save calibration data to default files.

Return values

true	Succeeded.
false	Failed.

Note

The default files are CalibrationData and OpticalDeviceBindInfo.xml which stored in user-based folder: $FO↩
LDERID_RoamingAppData/Hi5. Eg: C:/Users/your_name/AppData/Roaming/Hi5/CalibrationData.

4.2.1.17 StartMocapService()

```
static bool UMotionCaptureFunctionLibrary::StartMocapService (
            EGlovePositionSource PosSrc = EGlovePositionSource::VivePosition,
            bool ReadLocal = true )  [static]
```

Start MocapCapture plugin.

Parameters

in	*PosSrc*	Specify the source of the location data. The value is one of [Vive, Alice, Other].
in	*ReadLocal*	If you need read glove data from local Hi5 dongle, pass true. Otherwise, pass false.

Return values

true	Succeeded.
false	Failed.

4.2.1.18 StopMocapService()

```
static void UMotionCaptureFunctionLibrary::StopMocapService ( )  [static]
```

Stop the data service

4.2.1.19 VibrateGloves()

```
static void UMotionCaptureFunctionLibrary::VibrateGloves (
            const int LeftGloveTimeSpan,
            const int RightGloveTimeSpan )  [static]
```

Vibrate left and right gloves at the same time.

Parameters

in	*LeftGloveTimeSpan*	Left hand vibration duration.(unit: ms)
in	*RightGloveTimeSpan*	Right hand vibration duration.(unit: ms)

Note

The valid value of GloveTimeSpan is in the range [0, 5000]. The value larger than 5000 is equivalent to 5000. The value 0 means stop vibration. The negative value means do nothing, it's to say the glove keeps the states when calling this function.

4.2.1.20 VibrateLeftGlove()

```
static void UMotionCaptureFunctionLibrary::VibrateLeftGlove (
            const int GloveTimeSpan )  [static]
```

Vibrate left glove.

Parameters

in	*GloveTimeSpan*	Left hand vibration duration.(unit: ms)

Note

The valid value of GloveTimeSpan is in the range [0, 5000]. The value larger than 5000 is equivalent to 5000. The value 0 means stop vibration. The negative value means do nothing, it's to say the glove keeps the states when calling this function.

4.2.1.21 VibrateRightGlove()

```
static void UMotionCaptureFunctionLibrary::VibrateRightGlove (
            const int GloveTimeSpan )  [static]
```

Vibrate right glove.

Parameters

in	*GloveTimeSpan*	Right hand vibration duration.(unit: ms)

Note

The valid value of GloveTimeSpan is in the range [0, 5000]. The value larger than 5000 is equivalent to 5000. The value 0 means stop vibration. The negative value means do nothing, it's to say the glove keeps the states when calling this function.

The documentation for this class was generated from the following file:

- MotionCaptureFunctionLibrary.h

Chapter 5

File Documentation

5.1 MotionCaptureFunctionLibrary.h File Reference

This file contains all APIs that can be used for integrating Noitom Hi5 Glove into unreal engine.

```
#include "Kismet/BlueprintFunctionLibrary.h"
#include "BoneLists.h"
#include "MotionCaptureFunctionLibrary.generated.h"
```

Classes

- class UMotionCaptureFunctionLibrary

Enumerations

- enum ECalibrationPose : uint8 {
  GCP_TPose = 0, **GCP_APose**, **GCP_PPose**, **GCP_BPose**,
  GCP_CPose }
- enum EGloveMod : uint8 { **GIM_BothGloves** = 0, **GIM_LeftGlove**, **GIM_RightGlove** }
- enum EGlovePositionSource : uint8 { EGlovePositionSource::VivePosition = 0, EGlovePositionSource::↩
  AlicePosition, EGlovePositionSource::Other }
- enum EGlovePowerLevel : uint8 { EGlovePowerLevel::Unknown, EGlovePowerLevel::Low, EGlovePower↩
  Level::Normal, EGlovePowerLevel::Full }
- enum EGloveMagneticedState : uint8 { **Unknown**, EGloveMagneticedState::Bad, EGloveMagneticedState↩
  ::Warn, EGloveMagneticedState::Normal }

5.1.1 Detailed Description

This file contains all APIs that can be used for integrating Noitom Hi5 Glove into unreal engine.

Author

Baojing Zhou

Date

2017-08-31

This plugin works in its own context. Before the context is created, we cannot found Hi5 glove hardware. So the first thing is to create context. The UMotionCaptureFunctionLibrary::StartMocapService is used to do this, and UMotionCaptureFunctionLibrary::StopMocapService is used to destroy the context.

The Hi5 Glove is designed to work with htc vive together: glove captures hand gestures and htc vive tracks the position of the whole hand. But which vive tracker tracks the left hand, and which tracker tracks the right hand? We designed a serials of special poses to solve this problem.

To setup the relationship between glove and tracker, we need to do an operation called Calibration. Calibration is doing the special poses when calling UMotionCaptureFunctionLibrary::CalibrationGlove with parameter B-Pose or P-Pose. After B-Pose calibrated correctly, the relationship will be setup. If calibration failed or not satisfied, you can redo it. UMotionCaptureFunctionLibrary::SaveCalibrationData are used for saving the calibration result to files. Accordingly UMotionCaptureFunctionLibrary::LoadCalibrationData are used for loading those information.

5.1.2 Enumeration Type Documentation

5.1.2.1 ECalibrationPose

```
enum ECalibrationPose : uint8 [strong]
```

Defines the calibration pose type.

5.1.2.2 EGloveMagneticedState

```
enum EGloveMagneticedState : uint8 [strong]
```

Defines Hi5 Glove's magnetization level.

Enumerator

Bad	Serious magnetization.
Warn	Medium magnetization.
Normal	No magnetization.

5.1.2.3 EGloveMod

```
enum EGloveMod : uint8 [strong]
```

Defines the hand type.

5.1.2.4 EGlovePositionSource

```
enum EGlovePositionSource : uint8 [strong]
```

Defines the type of Hi5 Glove's position data source.

Enumerator

VivePosition	Hi5 Glove's position is supplied by HTC vive.
AlicePosition	Not used now.
Other	Not used now.

5.1.2.5 EGlovePowerLevel

```
enum EGlovePowerLevel : uint8 [strong]
```

Defines Hi5 Glove's battery level.

Enumerator

Unknown	Cannot get battery info.
Low	Battery level is low.
Normal	Battery level is normal and enough for normal use.
Full	Battery level is enough for normal use.

Index