モーションキャプチャ システム

OptiTrack™

NatNet SDK
ユーザーガイド

Version 2.10.0
2016年5月

オプティトラック・ジャパン株式会社

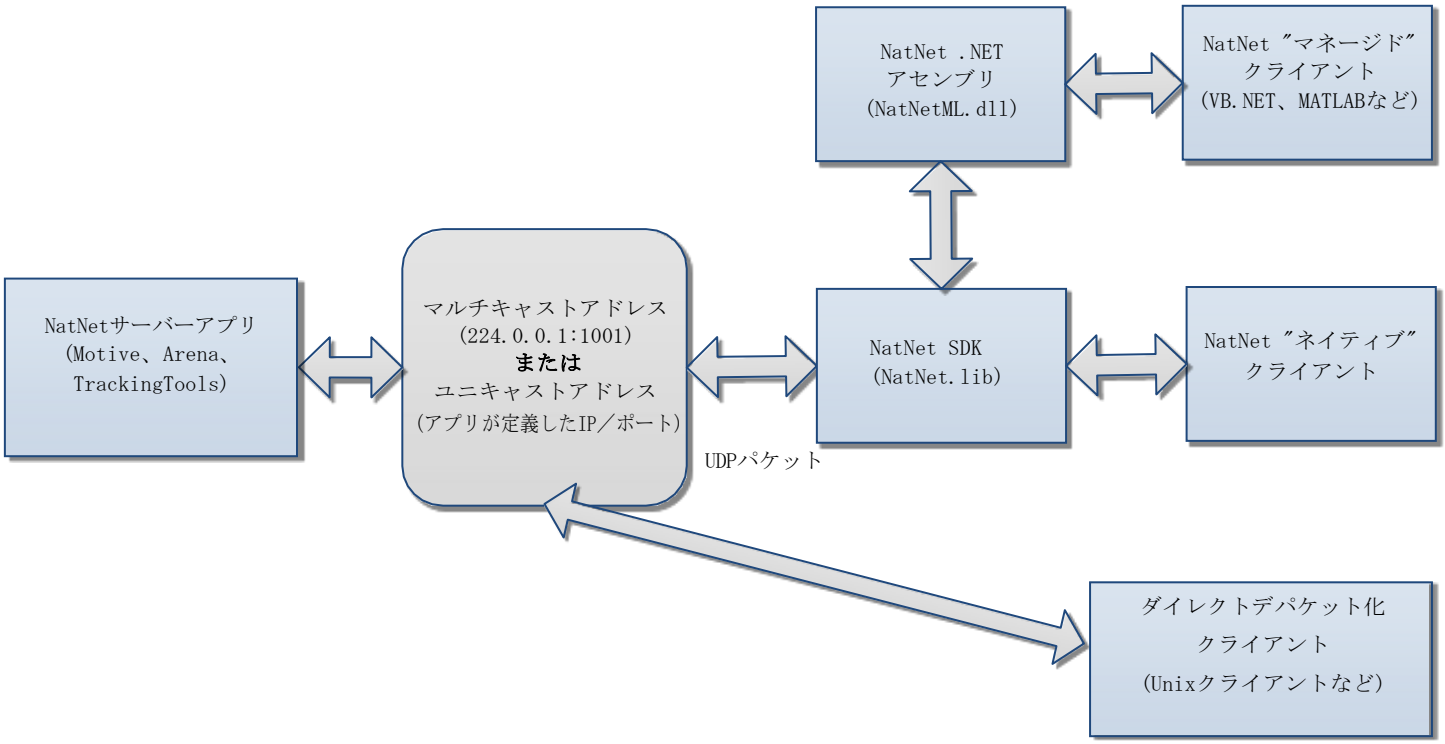
目次

NatNet概要

NatNet SDKは、クライアント／サーバーネットワーキング用のSDKです。ネットワーク上のNaturalPointデータの送受信を行います。NatNetでのデータ送信では、Point-To-Pointユニキャストまたは IPマルチキャスティングのいずれかに、UDPプロトコルを併用しています。

ここで紹介する図では、典型的なNatNet設定における主要なコンポーネントの関係の概要を示しています。

図表1： NatNetコンポーネント概要



NatNet サーバーには、2つのスレッドと2つのソケットがあり、1つはデータ送信用、1つはコマンドの送受信用となっています。

NatNetサーバーおよびクライアントは、同じコンピューターに設置することも、異なるコンピューターに設置することも可能です。また、複数の NatNetクライアントを、ひとつのNatNetサーバーに接続することもできます。IPマルチキャストを使用するようにNatNetサーバーを構成すると、データはマルチキャストグループに1度だけ送信されます。

SDKコンテンツ

NatNet SDKは次の要素で構成されています

NatNet Library ネイティブ C++用ライブラリ（ヘッダファイル、静的ライブラリ（.lib)、動的インポートライブラリ（.lib/.dll))。

NatNet Assembly .NET対応クライアントで使用するための.NETアセンブリ（NatNetML.dll)。

NatNet Samples お客様独自のコードに素早く統合させるために設計されたサンプルプロジェクトおよび実行ファイル。

フォルダコンテンツ

フォルダ	コンテンツ
¥include	NatNet SDKのヘッダファイル。クライアントアプリケーションに含まれています。
¥lib	NatNet SDKの静的ラブラリ（lib）および動的ライブラリ(dll)
¥lib¥x64	ライブラリファイルの64ビット版
¥Samples	VisualStudio 2013のサンプル。すべてのサンプルプロジェクトを開くために、このソリューションファイルを使用します。
¥Samples¥bin	コンパイル済みのサンプルと、サンプルデータファイル。
¥Samples¥BroadcastSample	XMLフォーマットのUDP ブロードキャストパケットを使ってMotiveの遠隔録画トリガーを使用する方法を示すサンプルアプリケーション。
¥Samples¥Matlab （下記で説明）	MATLABを.NETアセンブリ(NatNetML.dll)と併せて使用するためのサンプルMATLABコードファイル（.m)。
¥Samples¥MatlabWrapper	MATLAB用の.NETアセンブリラッパークラス
¥Samples¥MayaPlugIn	https://github.com/mocap-ca/mayaMocap/tree/master/mayaMotive
¥Samples¥NatCap	キャプチャ開始／中止ブロードキャストアプリのサンプル。
¥Samples¥PacketClient	NatNetマルチキャストストリームを接続し、NatNet SDKを使用せずにNatNetパケットを直接デコードする方法を示す例。
¥Samples¥PythonClient	NatNetストリーミングと併せてPythonを使用するためのPythonコードファイル（.py)のサンプル。

¥Samples¥SampleClient (説明は下記)	NatNetサーバーに接続し、データストリームを受信、Ascii ファイルにデータストリームを書き込むNatNetコンソールアプリのサンプル。
¥Samples¥SampleClient3D (説明は下記)	NatNetサーバーに接続し、データストリームを受信、 OpenGL 3ウィンドウにデータを表示するNatNetコンソールアプリのサンプル。
¥Samples¥SimpleServer	NatNetサーバーを使用してマーカーデータを作成および送信するための最低コード要件。
¥Samples¥TimingClient	NatNetサーバーに接続し、パケットのタイミング情報の定義を素早くチェックするためのプログラム。
¥Samples¥Unity3D (説明は下記)	NatNetサーバーに接続し、データストリームを受信、スケルトンをXMLにエンコードし、UDPを通じてXMLをUnityにローカル出力するためのプログラム。
¥Samples¥VCRedist	32ビット版Visual C++ Redistributable 2005(ランタイムライブラリ)の実行ファイル。
¥Samples¥WinFormsSample (説明は下記)	C#.NETサンプル。.NETアセンブリ (NatNETML.dll)の使用方法を示します。

サンプルを実行する

コンパイル済みバージョンのNatNet サンプルが¥Samples¥binフォルダに収納されています。コンパイル済みバージョンはアプリケーションを素早くテストするために使用できます。各サンプルを実行するには、このセクションを参照してください。

注意！ サンプルを実行するためには、Visual C++ランタイムライブラリが必要です。サンプルを実行する際にエラーメッセージが表示された場合、Samples¥VCRedist内のVC runtime redistributable packageをインストールしてください（Visual C++がインストールされていないコンピューターの場合）。それでもエラーメッセージが表示される場合、Visual C++を使用してサンプルをリビルドするか、サポートセンターまでご連絡ください。

コンソール出力のサンプルを実行する（SampleClient）

1. OptiTrackサーバー（Motiveなど）を起動し、ストリーミングパネルよりデータのストリーミングを開始します。

2. クライアントを起動します：
   SampleClient.exe [IPAddress] [OutputFilename.txt]

クライアントウィンドウまたはテキストファイル内でデータストリーミングが開始されます。

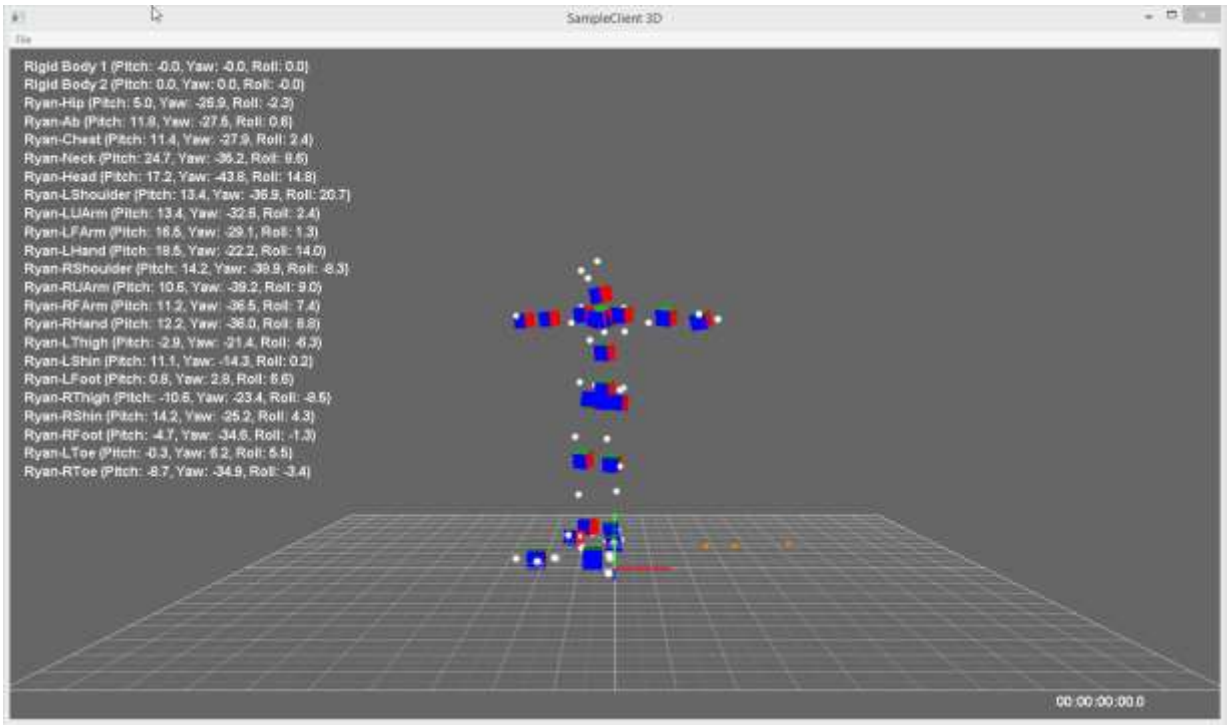
注意

- [Parameters] はオプションです。
- IPアドレスが指定されていない場合、クライアントはサーバーが同じコンピューター上にあると仮定します（ローカルマシン）。

剛体のサンプルを実行する (SampleClient3D)

剛体サンプル (SampleClient3D)は、剛体およびスケルトンセグメントデータをOptiTrackクォータニオンフォーマットのNatNet 6DOFからオイラー角にデコードし、OpenGL 3Dビューアに表示するための方法を示します。また、FrameOfMocapDataパケットにストリームされるIDを利用して剛体／スケルトンセグメントの名前とIDを関連づける方法を示します。

SampleClient3D -ラベル付けされた剛体のポジションと姿勢(6DOF) データをデコードおよび表示します。



クライアント／サーバーが同じコンピューターに存在する場合：

1. [Motive] 剛体またはスケルトンの定義およびデータセットを読み込みます。
2. [Motive] ネットワークストリーミングを有効にします。
   (Data Streaming Pane -> Broadcast Frame Dataを確認)
3. [Motive] 剛体データのストリーミングを有効にします。
   (設定を確認：Stream Options -> Stream Rigid Bodies = True)
4. [Sample3D] File -> Connectからサーバーへ接続します。

クライアント／サーバーが別のコンピューターに存在する場合；

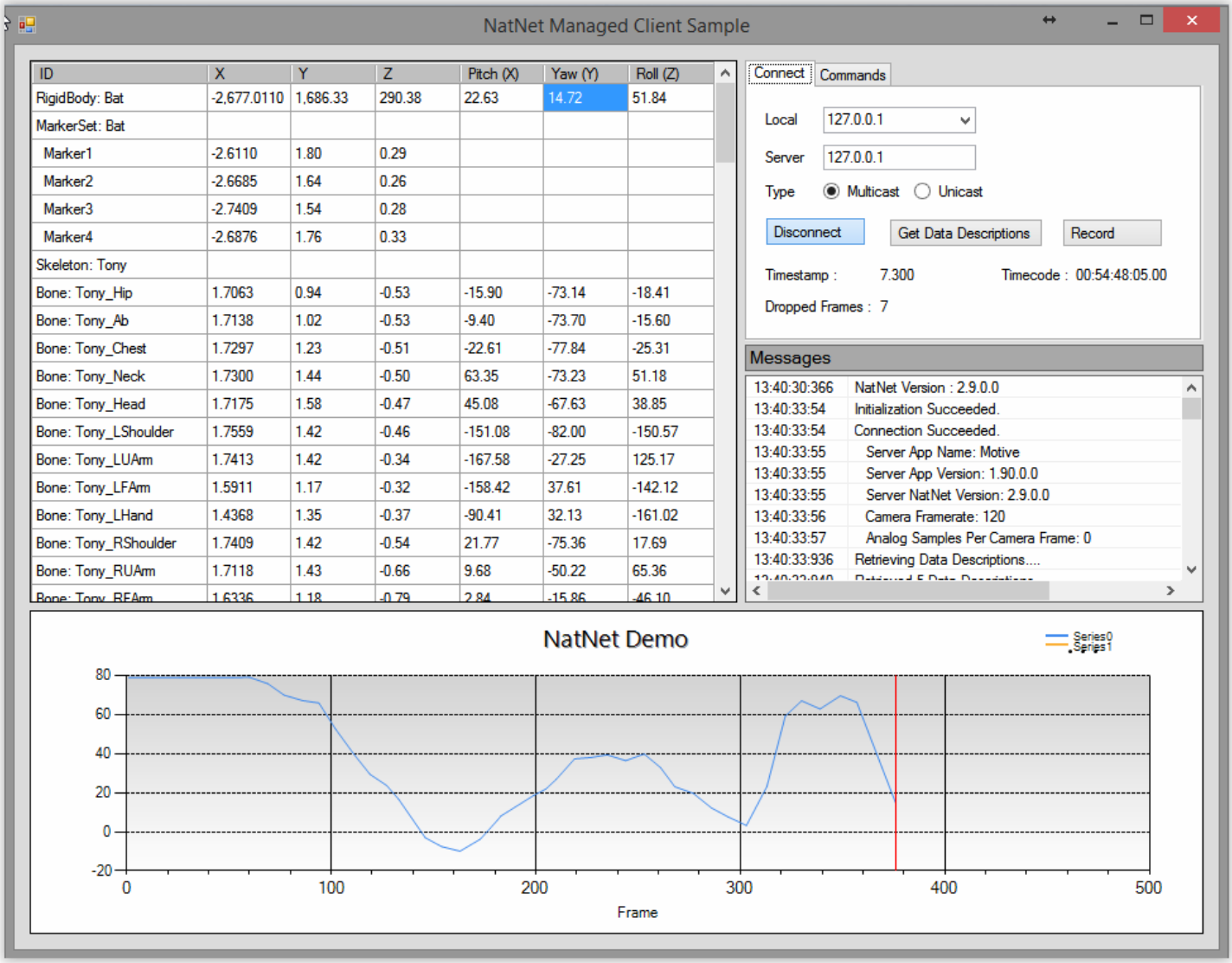
1. [Motive] 剛体またはスケルトンの定義およびデータセットを読み込みます。
2. [Motive] Network Interface Selection -> Local Interfaceからストリームするための IPアドレスを設定します。
3. [Motive] ネットワークストリーミングを有効にします。
   (Data Streaming Pane -> Broadcast Frame Dataを確認)
4. [Motive] 剛体データのストリーミングを有効にします。
   (設定を確認：Stream Options -> Stream Rigid Bodies = True)
5. [Sample3D] クライアントとサーバーのIPアドレスを設定します。
6. [Sample3D] File -> Connectからサーバーへ接続します。

- IP Address 使用するクライアントネットワークのIPアドレス。
- Server IP Address 前述のステップ2で入力したサーバーのIPアドレス。

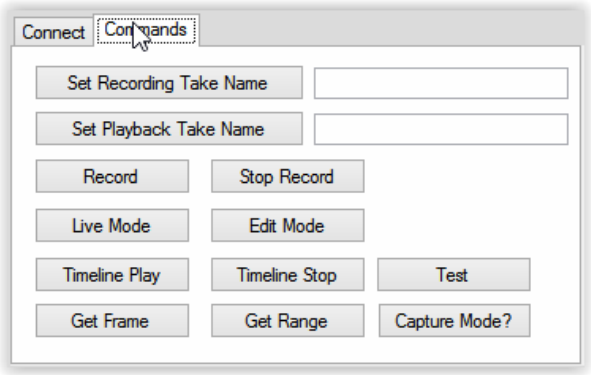
.NETのサンプルを実行する

1. [Motive] NatNetサーバーアプリケーション（Motiveなど）を起動します。
2. [Motive] サーバーアプリケーションからNatNetストリーミングを有効にします。
3. [WinFormTestApp] NatNetのサンプルフォルダからWinFormsサンプルアプリケーションを起動させます。
4. [WinFormTestApp] 必要に応じ「Local」および「Server」のIPアドレスを更新します。
5. [WinFormTestApp] 「Connect」ボタンをクリックし、サーバーに接続します。
6. [WinFormTestApp] 「GetDataDesc」ボタンをクリックし、サーバーに現在ストリーミングされてるオブジェクトの詳細記述を表示するようにリクエストします。
7. [WinFormTestApp] DataGridに含まれる行を選択して、グラフに値を表示します。

.NET環境でNatNetデータを受信する



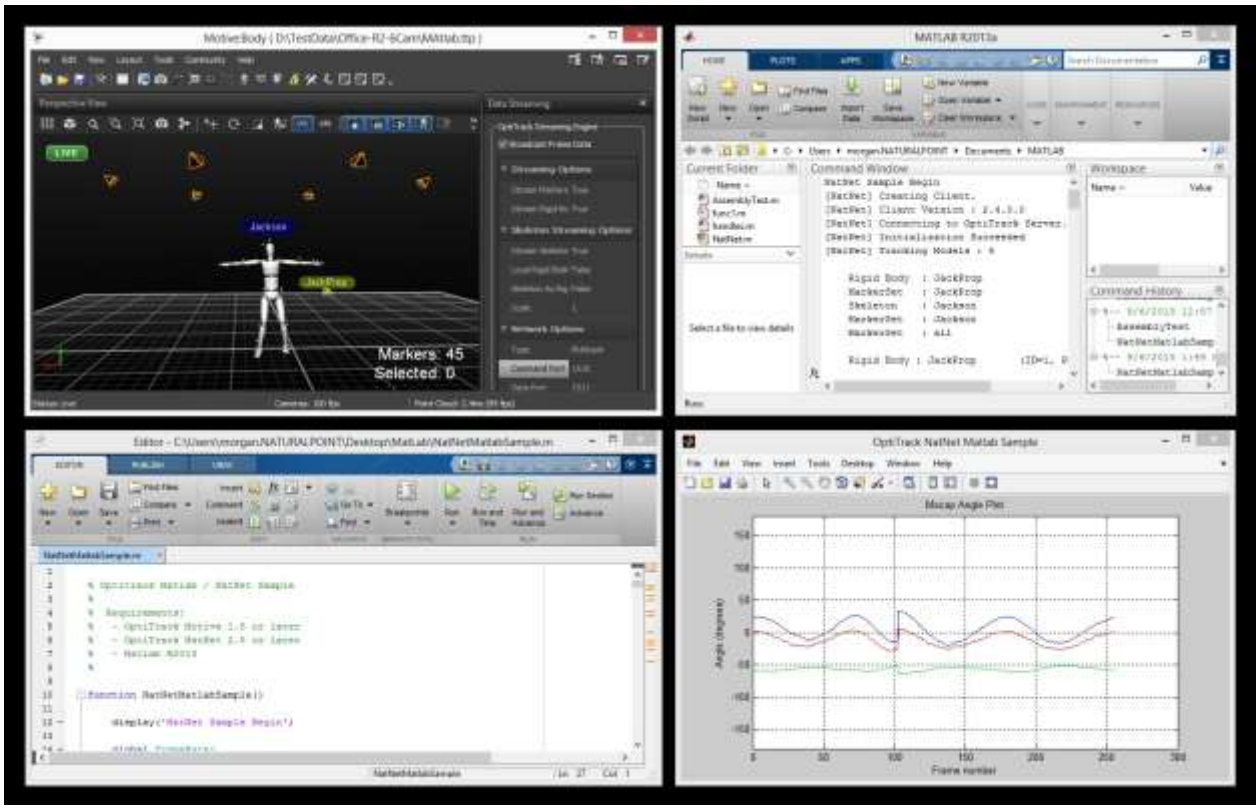
Motiveへの遠隔コントロールコマンドを発行する



MATLABサンプルを実行する

1. [**Motive**]　NatNetサーバーアプリケーション（Motiveなど）を起動します。
2. [**Motive**]　サーバーアプリケーションからNatNetストリーミングを有効にします。
3. [**MATLAB**]　MATLABを起動します
4. [**MATLAB**]　NatNetMatlabSample.mファイルを開きます。
5. [**MATLAB**]　Editor windowから**Run**をクリックします。

Motiveのモーキャプデータを*MATLAB*でリアルタイムストリーミングする

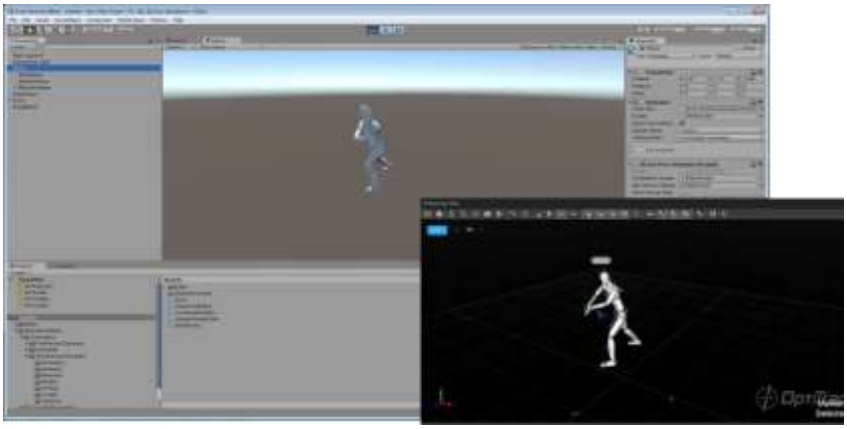


Unity3Dサンプルを実行する

NatNetフォルダに収納されているUnity3Dサンプルアプリケーションおよび C#スクリプトでは、UDPを通してMotiveの剛体およびスケルトンのトラッキングデータをUnityでストリーミングする方法を示します。下記のステップでは、このサンプルを使用してUnityでキャラクターを動かす方法を示しています。

1. [**Motive**] Motiveを起動します。
2. [**Motive**] スケルトントラッキングデータを準備します。過去にキャプチャしたテイクを開くことも、トラッキングをライブで直接ストリーミングすることも可能です。
3. [**Motive**] Data Streaming paneを開き、Local Rigid Body および Stream Skeletons をTrueに設定します。
4. [**Motive**] Network Interace Local Interface.を使用し、ストリーミングネットワークのアドレスを指定します。
5. [**Motive**] サーバーアプリケーションから、ネットワークストリーミングを有効にします。(MotiveのData Streaming Paneで Broadcast Frame Dataにチェックを入れます)
6. [**UnitySample**] ¥NatNet SDK¥Samples¥binフォルダから**UnitySample.exe**を実行します。このプログラムでは、NatNetサーバーに接続し、Motiveからトラッキングデータを受信、データをUnityに送信します。アプリケーションがホストを見つけることに失敗した場合、Motiveに戻り、他のネットワークインターフェイスでのストリーミングを試みます。
7. [**Unity**] Unityを起動します。
8. [**Unity**] 新規プロジェクトを作成します。
9. [**Unity**] ¥NatNet SDK¥Sample¥Unity3Dフォルダに収納されているC#スクリプトまたはアセットをUnityにインポートします。
10. [**Unity**] 空のゲームオブジェクトを作成し (GameObject Create Empty) 、SlipStreamと名前を付けます。
11. [**Unity**] **SlipStream.cs**スクリプトコンポーネントをSlipStreamオブジェクトに追加します。
12. [**Unity**] **SlipStream.cs**コンポーネント下に、サーバーアプリケーションで定義されたStreaming Unity3D IPアドレスを入力します。
13. [**Unity**] サンプルキャラクターをUnityにインポートします。(Asset Import Package Characters)
14. [**Unity**] Project panel内のimported assetsフォルダから、キャラクターを sceneに読み込みます。(Assets Standard Assets Characters ThirdPersonCharacters Models Ethan)
15. [**Unity**] キャラクターを選択し、inspector panelを使用して Animator classを無効にします。
16. [**Unity**] Add Componentをクリックし、**LivePoseAnimator.cs**スクリプトをキャラクターに追加します。
17. [**Unity**] LivePoseAnimatorコンポーネントのプロパティを入力します：
    1. Desintation Avatar下で、動かしたいアバターを選択します(Ethan EthanAvatar)。
    2. Slip Stream Object下で、**SlipStream.cs**を添付した空のオブジェクトを選択します。
    3. Actor下で、インポートしたいMotiveのスケルトン名を選択します。
18. [**Unity**] Objects下の設定 (SlipStreamおよびEthan) を再度確認し、プロジェクトを実行します。すべての設定が正しい場合、Motiveのトラッキングデータを使用してEthanが動きだします。

Unity3Dでリアルタイムストリーミングを行う



NatNet SDKを使用する

NatNetデータをアプリケーションに取り込むためには、コードサンプルを使用することが最速の方法です。通常、次の方法を推奨しています。

1. 使用しているアプリケーションの開発／インターフェイス要件を特定します（マネージド、ネイティブなど）。
2. サンプルフォルダに収納されている対応するNatNetサンプルアプリケーションのNatNetサンプルコードを、使用しているアプリケーションに適用させます。
3. 情報を追加する際は、次ページのAPIリファレンスを使用してください。

Visual Studioソリューションファイル ¥Samples¥NatNetSamples.sln を開き、すべてのNatNetサンプルプロジェクトをビルドします。

アプリケーションをゼロから作る場合、各アプリケーション独自の要件については次のセクションを参照してください。

NatNetデータを受信するためにNative Clientをビルドする

NatNetサーバーアプリケーション（Motiveなど）からデータを受信するために、NatNetクライアントアプリケーション／ライブラリをビルドするためのステップは次の通りです。

1. 使用しているアプリケーションのコードにSampleClientサンプルを適用します。
2. NatNetClient.hおよびNatNetTypes.hを含めます
3. NatNetLib.lib（動的）またはNatNetLibStatic.lib（静的)のどちらかをリンクします
4. [オプション]動的をリンクした場合、NATNETLIB_IMPORTSを定義し、使用しているアプリケーションと併せてNatNetLib.dllを配布します。

注意：NatLetLibを静的にビルドする場合、必ずws2_32.libをリンクしてください。

NatNetデータを送信するためにNative Serverをビルドする

NatNetフォーマットのデータをNatNetクライアントアプリケーションに送信／転送するために、NatNetサーバーアプリケーション／ライブラリをビルドするステップは次の通りです。

1. SimpleServer（SampleServer.cpp)を、使用しているアプリケーションのコードに適用します。
2. NatNetServer.hおよびNatNetTypes.hを含めます。
3. NatNetLib.lib（動的）またはNatNetLibStatic.lib（静的)のどちらかをリンクします
4. [オプション]動的をリンクした場合、NATNETLIB_IMPORTSを定義し、使用しているアプリケーションと併せてNatNetLib.dllを配布します。

注意：NatLetLibを静的にビルドする場合、必ずws2_32.libにリンクしてください。

NatNetデータを受信するためにマネージド.NETクライアントをビルドする

マネージドNatNetクライアントアプリケーションをビルドするためのステップは次の通りです。

1. VB.NET/C# プロジェクトの参照設定にNatNetML.dll .NETアセンブリを加えます。
2. これにより、使用しているコードでインテリセンスライブラリコメントと併せて NatNetMLネームスペースが使用できるようになります。

注意： .NETアプリケーションを配布する際は、必ずNatNetML.dllも併せて配布してください。

API リファレンス

NatNET APIは次のオブジェクトで構成されています：

NatNetClient MotiveなどのNatNetサーバーと通信するためのクラス。

NatNetServer NatNetサーバーとNatNetフォーマットデータパケットを導入するためのクラス。

NatNet Data Types NatNetパケット内にエンコードされた、構造体をカプセル化するためのデータ。

NatNet Assembly .NETコンポーネントで呼び出すことが可能なマネージド（.NET)クラスライブラリ。NatNetアセンブリがネイティブNatNetライブラリをラップし、.NET対応環境（VB.NET、 C#、LabVIEW、MATLABなど)で使用する NatNetClient およびNatNet Data Typesを提供。

NatNet Data Types

NatNetサーバーアプリケーションは、次のタイプのモーションキャプチャデータをストリーミングします。

NatNet Data Types

Data Type	記述
MarkerSet Data	特定されているマーカーとマーカーポジション（X、Y、Z）に関するリストです。ordered, padded, point cloud solved, model filled（オクルージョンの部分）が含まれます。「all」と命名された特別なMarkerSetが含まれています。ラベル付けされたすべてのマーカーのリストです。
RigidBody Data	一意のID、ポジション、姿勢データ、また剛体を定義するために使用された一連のマーカーに関するセグメント。マーカーデータは、model-solved positionsです。
Skeleton Data	剛体の階層リストです。マーカーデータは、model-solved positionsです。
Labeled Markers	ordered, padded, point cloud solved, model filled（オクルージョンの部分）のラベル付きマーカーデータ(ラベル付きマーカーはMarkerSetとは関連づけられていません)。
Unlabeled Markers (Other markers)	スケルトンあるいは剛体ソルバによるラベル付けが行われていない、フレーム内のすべてのマーカーのポイントクラウドソルブ3Dポジション。
Force Plate Data	フォースプレートチャネルデータ（Fx、Fy、Fz、Mx、My、Mz)。フォースプレートデータには、モーキャプフレーム1つに対して複数のサンプルを含んでいます。これはフォースプレートの取得レートにより変わります。
Timestamp data	フレームのタイミングに関する情報。以下を含みます： Frame ID Frame Timestamp SMPTE Timecode（タイムコードが存在する場合）

NatNetクライアントは、**DataSetDescriptions構造体**を使用して、事前にサーバーアプリケーションが出力しているデータオブジェクトの内容を確認することができます。NatNetクライアントは**FrameOfMocapData**を使用してサーバーから実際のデータを受信します。どちらのパケットも、DataHandlerコールバックを通じてクライアントに配信されます。

Dataset Descriptions: このパケットには、モーションキャプチャデータセット（マーカーセット、スケルトン、剛体）の記述が含まれています。また、このデータセット向けにモーションキャプチャデータのフレームが生成されます。

Frame of Mocap Data: このパケットには、Dataset Descriptionsに記述されているデータセットすべてのためのシングルフレームのモーションキャプチャデータが含まれています。

前述のデータセット（マーカーセット、スケルトン、剛体）に加えて、FrameOfMocapDataには、1フレーム単位の追加のートラッキングデータが含まれています。この追加データの情報は事前に知ることができない為、DataSetDescriptions構造体には記述されていません。

Labeled Markers: 定義済みのマーカーセットまたは剛体に関連づけられていないラベル付きマーカー。このデータタイプは、マーカーセットまたは剛体がトラッキングアプリケーションで明示的に定義されていないにも関わらず、ラベル付きマーカーが生成されている場合に使用されます。

Other Markers: 対象フレームにおいて、ラベル付けされていないすべての3Dポイント。

データが構造体間で重複している可能性があります。たとえば対象とする**FrameOfMocapData**において、**LabeledMarkers**及び**RigidBody Data**(sRigidBodyData構造体)で同じマーカーが含まれている場合があります。このような場合、必要に応じてクライアントコード内で修正するためにマーカーIDを参照します。

SampleClientサンプルは、データセットの記述（**Dataset Descriptions**）およびデータ（**Frame of Mocap Da**）を取得し、そのデータを解釈する方法を示しています。

各タイプの最新のデータ記述については、***NatNetTypes.h***ヘッダファイルまたは ***NatNetML.dll***アセンブリを参照してください。

座標系の規約

NatNetデータストリームでは、剛体の姿勢データはクォータニオンとなります。クォータニオンは回転軸順序に非依存ですが、右手系左手系に依存します。クォータニオンをオイラー角に分解する際は、変換させる座標系の規約を検討することが大切です。オイラー角の座標系規約では、次の点を考慮する必要があります。

- 回転軸の順序
- 左手座標系または右手座標系
- 絶対軸（Global）または相対軸（Local）

例としては、OptiTrack Motiveソフトウェアが使用する座標系規約は、次のような 「Motive」座標系となります。

X（ピッチ）、Y（ヨー）、Z（ロール）順序、右手座標系(RHS)、相対軸(Local)

NatNet SDKにはクォータニオンからオイラー角への変換ルーチンが含まれています。 それぞれの導入方法の詳細や使用例は、WinFormsサンプルまたはSampleClient3Dを参照してください。

遠隔コマンドおよびコントロール

NatNetは、NatNetサーバー(Motive) およびNatNetクライアント（使用しているアプリケーション)間でコマンドとリクエストを送受信するためのコマンド／リクエストメカニズムを備えています。コマンド／リクエストの例として、録画の開始／中止、撮影中のテイク名の設定、現在のフレームレートに関するクエリなどがあります。

Motiveに対応しているコマンド／リクエスト

コマンド	説明	パラメータ	戻り値：
UnitsToMillimeters	単位をミリメートル(mm)にリクエスト	なし	float
FrameRate	現在のシステムトラッキングフレームレートをリクエスト	なし	float
StartRecording	録画開始	なし	なし
StopRecording	録画中止	なし	なし
LiveMode	Liveモードに変更	なし	なし
EditMode	Editモードに変更	なし	なし
CurrentMode	現在のモードをリクエスト	なし	int
TimelinePlay	テイクの再生を開始	なし	なし
TimelineStop	テイクの再生を中止	なし	なし
SetPlaybackTakeName	再生テイクの設定	テイク名	なし
SetRecordTakeName	録画テイクの名前の設定	テイク名	なし
SetCurrentSession	現在のセッションの設定	セッション名	なし
SetPlaybackStartFrame	開始フレームの設定	フレーム番号	なし
SetPlaybackStopFrame	停止フレームの設定	フレーム番号	なし

SetPlaybackCurrentFrame	現在のフレームの設定	フレーム番号	なし
CurrentTakeLength	現在のテイクの長さをリクエスト	なし	int
AnalogSamplesPerMocapFrame	モーションキャプチャのフレームあたりのアナログサンプル数をリクエスト	なし	int

- これらのコマンドのリストおよび使用方法の全貌についてはWinFormsの例を参照してください。
- さらに詳しくはAPIレファレンスの *SendMessage(…)* および *SendMessageAndWait(…)* 関数を参照してください。

タイムコード

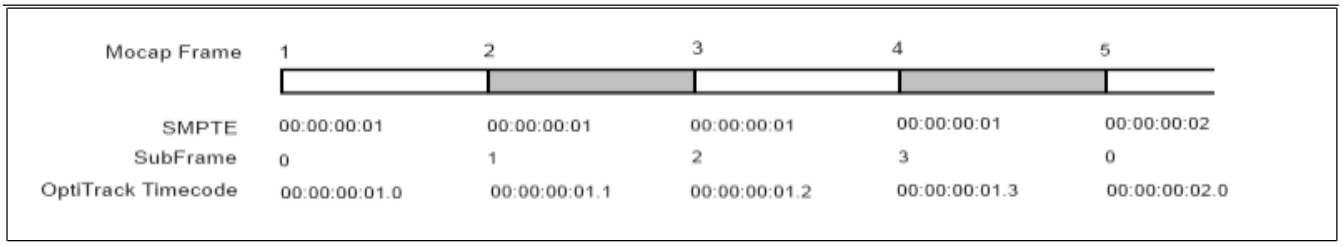
対応するシステムでは、NatNetデータのフレームすべてにOptiTrackタイムコードスタンプが含まれます。OptiTrackタイムコードスタンプは一般的なTVスタジオで使われるSMPTEタイムコードスタンプの拡張フォームです。

注意：SMPTEタイムコードに対応するためには、OptiTrack eSyncハブが必要です。

モーションキャプチャのフレームレートは通常のSMPTEタイムコードフレームレートを超えるため、「subframe」値をタイムコードスタンプの最後に追加する必要があります。「subframe」値は0を基準値として、フレームの間のn番目となります：

典型的なOptiTrackタイムコードの表示

(120 fps mocap data, 30-fps no-drop SMPTE timecode source)



上記の例のように、一般的な120fpsモーションキャプチャセッションを、 30 fps、no-drop SMPTEタイムコードのスタジオ-同期ソースに同期した場合、モーションキャプチャフレームとスタジオフレームは**4 :1**の割合となります。追加されたモーションキャプチャフレームは、OptiTrackタイムコードのOptiTrack SubFrame欄に表示されます。

OptiTrackタイムコードの汎用的なフォームは：

HH:MM:SS:FF.Y	時：分：秒：フレーム.サブフレーム

OptiTrackタイムコードが2つの符号なし整数の形式でNatNetクライアントに送信されます。

unsigned int Timecode	OptiTrackエンコードSMPTEタイムコード
unsigned int TimecodeSubframe	OptiTrackエンコードサブフレームデータ

注意：FrameOfMocapData構造体では、タイムコードデータのフレームにタイムコード情報に関する2つのエントリーが含まれています。Timecodeパラメータはライブストリーミングの際とファイル再生（編集)の際では異なって解釈されます。
ライブモードでは、Mocapハードウェアの設定にSMPTEタイムコードが存在する場合のみTimecodeパラメータが有効となります。これは、通常eSync およびタイムコードジェネレーターを使用しているケースです。Timecodeパラメータの値が存在する場合、その値は正確にフォーマットされたSMPTEタイムコードの値となります。

編集モードでは、**Timecode**パラメータはSMPTEタイムコードフォーマットに変換された現在のフレーム番号となります。

NatNet SDKは、Timecodeパラメータを、文字列で表示するフォーマットにデコードするためのヘルパールーチンを提供します。また、整数のタイムコードフレームの間に存在する場合があるサブフレームのデコードも行います。

Latencyは、キャプチャを行うコンピューターが各フレームに与えるタイムスタンプです。カメラ情報が有効になっている際には、Motiveのカメラプレビュービューポートにも表示されます。ライブの際もファイルを再生する際も同様です。

タイムコードの値は、直接使用するのではなく、NatNetタイムコードユーティリティ関数を使用してデコードする必要があります。

`bool DecodeTimecode(unsigned int inTimecode, unsigned int inTimecodeSubframe, int* hour, int* minute, int* second, int* frame, int* subframe);`	OptiTrackタイムコードデータを各タイムコード値にデコードするためのヘルパー関数
`bool TimecodeStringify(unsigned int inTimecode, unsigned int inTimecodeSubframe, char *Buffer, int BufferSize);`	OptiTrackタイムコードデータをユーザーフレンドリーな数列「hh:mm:ss:ff:yy」にデコードするためのヘルパー関数

NatNetヘルパー関数を使用してタイムコードをデコードする方法の例です。（*SampleClient.cpp*より）

```
// decode timecode to values
int hour, minute, second, frame, subframe;
bool bValid = pClient->DecodeTimecode(data->Timecode, data->TimecodeSubframe, &hour, &minute,
                                                    &second, &frame, &subframe);

// decode timecode to friendly string char s
zTimecode[128] = "";
pClient->TimecodeStringify(data->Timecode, data->TimecodeSubframe, szTimecode, 128); printf("Timecod
e :%s¥n", szTimecode);
```

NatNetClient Class

説明

NatNetClientは、MotiveなどのNatNetサーバーアプリケーションを接続するための完全なC++クラスです。

コンストラクタ&デストラクタ用ドキュメンテーション

NatNetClient::NatNetClient ()

NatNet Clientの新しい（マルチキャスト）インスタンスを作成する。

NatNetClient::NatNetClient (int iConnectionType)

指定した接続プロトコルを使用してNatNet Clientの新しいインスタンスを作成する。

パラメータ：

iConnectionType　接続の種類（0 = マルチキャスト、 1 = ユニキャスト）

NatNetClient::~NatNetClient ()

デストラクタ

NatNetClient::Uninitialize()

サーバーから切断する。

メンバー関数ドキュメンテーション

int NatNetClient::GetDataDescriptions (sDataDescriptions ** pDataDescriptions)

サーバーアプリから、現在ストリーミングされているデータオブジェクトの記述をリクエストします。リクエストへのレスポンスがあるかタイムアウトするまで、この呼び出しはブロックされます。

パラメータ：

pDataDescriptions データ記述の配列

戻り値：

成功：データオブジェクトの数それ以外：0

sFrameOfMocapData * NatNetClient::GetLastFrameOfData ()

直近に受信したモーキャプデータのフレームを検索。

戻り値：

モーキャプデータのフレーム

int NatNetClient::GetServerDescription (sServerDescription * pServerDescription)

クライアントが接続されている現在のNatNetサーバーの記述をリクエスト。リクエストへのレスポンスがあるかタイムアウトするまで、この呼び出しはブロックされます。

パラメータ：

pServerDescription　　NatNetサーバーの記述

戻り値：

成功：データオブジェクトの数それ以外：0

int NatNetClient::Initialize (char * szLocalAddress, char * sz ServerAddress)
int NatNetClient::Initialize (char * szLocalAddress, char * sz ServerAddress, int HostCommandPort)
int NatNetClient::Initialize (char * szLocalAddress, char * szServerAddress, int HostCommandPort, int HostDataPort)

クライアントソケットを初期化し、指定したアドレスでNatNetサーバーを接続するように試みます。

パラメータ：

szLocalAddress	クライアントのIPアドレス
szServerAddress	サーバーのIPアドレス
HostCommandPort	サーバーのコマンドポート(デフォルト = 1510)
HostDataPort	サーバーのデータポート(デフォルト = 1511)

戻り値：

成功した場合は0、それ以外はエラーコード

void NatNetClient::SetMulticastAddress (char * szMulticast)

接続するNatNetサーバーマルチキャストグループ／アドレスを設定します。SetMulticastAddress()は必ずInitiaize(...)の前に呼ぶ必要があります。

パラメータ：

szMulticast　　マルチキャストアドレス

void NatNetClient::NatNetVersion (unsigned char Version[4])

クライアントが使用しているNatNetライブラリのバージョンを取得します。

パラメータ：

Version　　バージョンの種類(form: major.minor.build.revision)

void NatNetClient::SendMessage (char * szCommand)

サーバーおよび戻り値にメッセージを送信します。レスポンスはインバンド方式となります。

パラメータ：

szCommand	アプリケーションが定義するメッセージ列

int NatNetClient::SendMessageAndWait (char * szCommand, int tries, int timeout, void ** Response, int * pnBytes)

アプリケーションが定義したメッセージをNatNetサーバーに送信し、レスポンスを待ちます。

パラメータ：

szCommand	アプリケーション定義のメッセージ
tries	メッセージの送信を試みた回数
timeout	タイムアウトになる前にレスポンスを待つ時間（ミリ秒で表示）
Response	アプリケーション定義のレスポンス
pnBytes	レスポンスのバイト数

戻り値：

成功した場合は0、それ以外はエラーコード

int NatNetClient::SendMessageAndWait (char * szCommand, void ** Response, int * pnBytes)

アプリケーションが定義したメッセージをNatNetサーバーに送信し、レスポンスを待ちます。

パラメータ

szCommand	アプリケーション定義のメッセージ
Response	アプリケーション定義のレスポンス
pnBytes	レスポンスのバイト数

戻り値：

成功した場合は0、それ以外はエラーコード

int NatNetClient::SetDataCallback(void (*CallbackFunction)(sFrameOfMocapData *FrameOfData, void* pUserData), void* pUserData /*=NULL*/)

NatNetフレームデリバリ用のデータコールバック関数を設定。この関数は、NatNetがインバンドデータ（データフレームなど）を受信した際に呼び出されます。

パラメータ

CallbackFunction	コールバック関数
pUserData	ユーザーが定義するデータ

戻り値：

成功した場合は0、それ以外はエラーコード

void NatNetClient::SetVerbosityLevel (int iLevel)

NatNet内部メッセージのメッセージレポートレベルを設定します。

パラメータ

iLevel	冗長レベル (NatNetTypes.h内のVerbosity levelを参照)

int NatNetClient::Uninitialize ()

現在のNatNetサーバーから切断。

戻り値：

成功した場合は0、それ以外はエラーコード

付録A：ビットストリーム構文

最新のビットストリーム構文を提供するために、NatNet SDKは、テストが容易な作業用デパッケージ化サンプルを有しています。これにより、NatNetクライアントライブラリを使用せずに直接NatNet Packetsをデコードします。

注意：パッケージを直接デコードすることは推奨していません。ビットストリームパケット構文は変更される場合があり、最新のNatNet Packetsライブラリ向けにリビルドするためのアプリケーションが必要です。NatNetライブラリが使用できない場合にのみ、NatNet Packetsを直接デコードします。

NatNetクライアントライブラリを使用すると、クライアントアプリケーションを将来のビットストリーム構文の変更から保護することが可能です。

Direct Depacketization Clientをビルドする（NatNetなし）

NatNetクライアントライブラリが使用できない場合（Unixなどの未対応プラットフォーム等）にNatNetデータストリームを受信したい際は、NatNet直接をデパケット化するためのテンプレートとしてPacketClientサンプルを使用できます。

1. 使用しているアプリケーションのコードにPacketClientサンプルを適用します。
2. NatNetビットストリーム構文にリビジョンがある際には、コードを定期的に更新してください。

オプティトラック・ジャパンのWebサイトでは、ユーザーの皆さまに向け、
「サポート」ページを用意しております。
NatNet SDKの最新版のダウンロード等が可能です。

https://www.optitrack.co.jp/support.html

お問い合わせ

オプティトラック・ジャパン株式会社

電話　　：03-5413-4882
受付時間：平日10:00～17:00
メール：support-desk@optitrack.co.jp
Skype　：OptiTrackJapan_SupportDesk
受付時間：平日10:00～17:00（要予約）