解 説

拡張可能なロボット用統合GUI環境 Choreonoid

Choreonoid: Integrated Extensible GUI Environment for Robots

中 岡 慎一郎* *産業技術総合研究所

Shin'ichiro Nakaoka* *National Institute of Advanced Industrial Science and Technology (AIST)

1. はじめに

ロボットの研究開発や運用をしていると，対象ロボットシステムのモデルやデータ，プログラムを扱うためのグラフィカルユーザインタフェース（GUI）が欲しくなる場面がある．その代表的な例は，ロボットの動作シミュレーションであろう．シミュレーション結果を把握するためには，結果を3DCGによるアニメーションやグラフ表示等によって可視化することが欠かせない．さらに，モデルの位置／姿勢の変更といった各種設定について，グラフィックと連動したインタラクティブなインタフェースで行うことができれば，多くのユーザにとってシミュレーションをより円滑に進めることが可能となる．実際に，OpenHRP [1] や Webots [2], Gazebo [3] のように，そのようなGUIを備えたロボット用シミュレータが開発され広く利用されている．

本稿では，そのようなGUIを効率的に開発するためのソフトウェアフレームワークとして筆者が中心となって開発しているChoreonoid [4] を紹介する．これを用いることで，実際に**図1**に示すようなGUIを備えた本格的なロボットシミュレータも実現できており，ほかにも様々な機能をユーザが追加していくことが可能なフレームワークとなっている．

そもそも，ロボットシステムを扱うGUIを新たに開発しようとすると，その開発コストが問題となる．一般的にGUIの開発はそれに特有のスキルと多くの手間を必要とする作業である．さらにロボットが対象になると，3DCGによるモデルの描画／ドラッグや，多くのデータやインタフェースの統合が必要となることも多く，その開発にはCGやフレームワーク設計に関する高度なプログラミングスキルが求められる．それらのスキルや手間はロボットシステム本体の開発に追加して必要となってくる部分であり，ロボットの開発者や研究者にとって大きな負担となってしまう．

このことが原因で，対象ロボットシステムを扱う十分な機能を備えたGUIの利用を諦めてしまうことも多い．例えば，システムの設定や操作のほとんどをプログラムコードによる記述や文字列コマンドの入力で対応し，GUIについては実行結果の確認のみを既存のシミュレータやビューアを用いて行うという状況もよく見られる．それで十分な場合もあるだろうが，一方で研究開発の進展につながる試行錯誤が十分にできなかったり，開発した技術の利用を研究室の外部に広めていくことが困難であったりすることも，実際の状況としては多いと思われる．

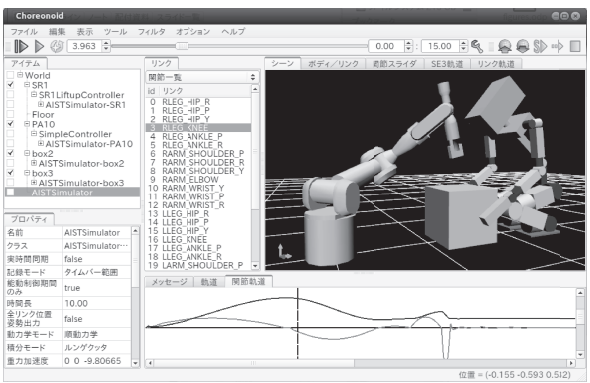


図1 Choreonoidによるシミュレーションの例

この状況を改善するために考慮すべきことは，やはりソフトウェアの再利用である．よく設計されたフレームワークの上で，共有可能な機能は共有し，個々の状況で必要となる機能はその機能のみの実装に集中できるようにすれば，開発のコストは大いに低減できるはずである．さらに，追加した機能の間の連携がしやすければ，利用事例の増加に伴って開発効率がさらに向上していくことが期待できる．そのような環境に実装された研究開発の成果は，普及もしやすくなるだろう．以上をGUIの領域でも進めていき，これまでにない「拡張可能なロボット用統合GUI環境」となるべく開発を進めているのが，Choreonoidなのである．

2. 設計方針

まずChoreonoidの特徴と類似システムとの相違を設計方針の観点から明らかにしたい．以下の五つの要求に応えることが，Choreonoidの設計における主な目標である．

原 2013年1月7日
キーワード：Robot Software Platform, Graphical User Interface

*〒305–8568 茨城県つくば市梅園 1–1–1
*Tsukuba–shi, Ibaraki

（1）GUI の全域をカバーすること

Choreonoid は GUI も対象に含むフレームワークであり，この点で近年利用の盛んな OpenRTM [5] や ROS [6] ノードシステムといったミドルウェアとは異なる．さらに，GUI の全域をカバーし，ユーザの機能追加において GUI としてできることはほぼすべてできるようにすることを目標としている．このレベルの領域を扱うことは，今までのロボット用フレームワークではなかったことである．なお，Choreonoid と上記ミドルウェアは決して排他的な関係にあるわけではなく，連携させることで互いに機能を補完しあえる関係となることを目指している．

（2）ロボットモデルを扱う基本機能を提供すること

Choreonoid では，ロボットモデルを扱う基本機能として，内部計算処理から GUI 上でのモデルの操作まで統合的にカバーしたものを提供する．これにより，単なる GUI フレームワークではなく，ロボットを対象とした GUI システムの開発・運用において実際に有用なフレームワークとなることを目指している．

（3）多様な機能を追加可能とすること

従来のロボット用 GUI システムの中にもユーザによる機能拡張が可能なものは存在した．例えば，物理エンジンを入れ替えることが可能なシミュレータや，表示する内容をカスタマイズできるモデルビューアといったものがこれに相当する．しかし，それらの拡張性は対象とする機能に限定されたものであり，想定外の機能，例えばインタラクティブな振り付け機能といったものを追加することは不可能であった．これに対して，Choreonoid は特定の機能を想定しているわけではなく，ユーザの必要に応じて多様な機能を追加できることを目指しており，この点で従来の拡張可能なロボット用 GUI システムとは異なるものである．

（4）追加機能を統合可能であること

多様な機能を追加可能であっても，それらが単に別々に利用可能であるだけであれば，それぞれの機能を異なる複数のツールとして実装した場合と大きく変わらないことになる．これに対して，追加した機能をうまく統合することができれば，インタフェースもよりシンプルとなり，複数の機能を連携させたより複雑な処理も可能となる．これを実現するにあたっては，（2）に示した基本機能の共有も大きな役割を果たすが，Choreonoid はそれだけでなく，統合を支援する汎用的な仕組みもいくつか導入し，追加機能の高レベルな統合化を目指している．

（5）最大限の実行効率を得られるようにすること

ロボット用 GUI システムの多くは，必ずしも実行効率を重視したものではない．例えば OpenHRP [1] や Gazebo [3] は，計算処理を行うサーバ群とフロントエンドの GUI をネットワーク接続で連携させる分散オブジェクト型の構造を採用している．また，ユーザによる機能拡張のほとんどを，Python や Java といった非ネイティブコード型の言語で行うシステムも多い．そのような構造は状況に応じて利点のあるものであるが，実行効率の面では不利である．これに対して Choreonoid は C++言語によって記述される単一プロセス構造を基本としており，実装される機能は最大限の実行効率を得ることが可能である．これによって，重い計算処理と GUI を高いフレームレートで連動させなければならないような機能も実現しやすい．一方で，プロセスの分散化やスクリプト言語の利用については，必要に応じて部分的に導入する方針をとっている．

3. 機 能 と 応 用

Choreonoid の機能の構成を**図 2** に示す．この図に示されるように，Choreonoid の構造は大きくフレームワーク部とプラグイン部に分けられる．フレームワーク部は GUI システム全体を統合管理するための機能と各種ライブラリからなり，より特化した機能はフレームワーク上で動作するプラグインとして実装される．図 2 のプラグイン部に含まれるのは主な既存プラグインであり，ユーザによる追加機能もプラグインとして実装されここに追加されることになる．プラグインはフレームワークの機能だけでなく，他のプラグインの機能も直接利用することが可能であり，この場合プラグインの間に依存関係が生じる．図 2 ではプラグインを含むすべてのモジュール間の依存関係をモジュール間の上下関係で表している．

以下では図 2 に示される機能のいくつかを紹介し，Choreonoid を用いて実際にどのようなことが実現できるのかを示す．なお，それらの機能の多くはプラグインとして実装されており，ユーザもそれらに劣らない機能をプラグインとして実現可能であると言える．

3.1 汎用インタフェース

図 3 に示すのは，フレームワークの Base モジュールが提

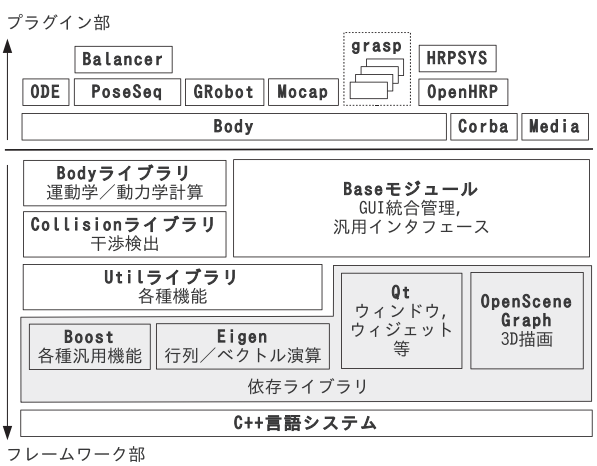


図 2 フレームワークと主なプラグインの構成

タイムバー：時刻・アニメーションを管理

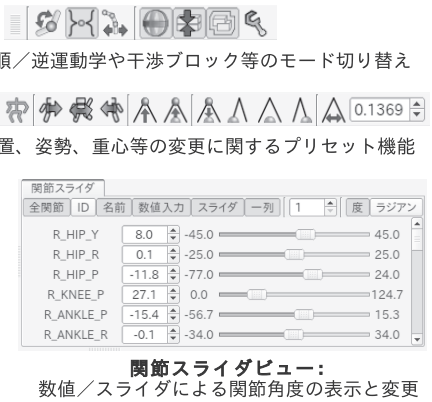
アイテムツリービュー：
操作対象モデル／データを管理

プロパティビュー：
アイテムの属性を管理

シーンビュー：
3DCGによる表示・入力

グラフビュー：
軌道データのグラフ表示

図 3 汎用インタフェース



運動学バー：順／逆運動学や干渉ブロック等のモード切り替え

姿勢バー：位置、姿勢、重心等の変更に関するプリセット機能

リンクビュー：
対象部位の指定

関節スライダビュー：
数値／スライダによる関節角度の表示と変更

図 4 ロボットモデル操作用基本インタフェース

供する汎用インタフェースのいくつかである．インタフェースの主な単位は「ツールバー」と「ビュー」であり，これらは「メインウィンドウ」内で統合的に管理される．ツールバーやビューの配置は作業内容に応じてユーザが自由に変更することが可能である．また，プラグインによって独自のツールバーやビューを追加することも可能である．

このなかでも特に重要なのが，「アイテムツリービュー」である．ここではGUI上でユーザが明示的に扱うモデルやデータが「アイテム」として抽象化されツリー構造を用いて統合的に管理される．機能の追加に必要であれば，アイテムの型もプラグインによって追加・拡張していくことが可能である．アイテムツリービューでは，アイテムの位置や親子関係，チェック状態，選択状態をユーザが自由に変更することが可能である．それらの情報は，複数のアイテムが対象となる機能において，対象アイテムやアイテム間の対応関係を決めるために参照される．この仕組みは，多様なデータを追加し統合的に扱うにあたって有効に機能する．

3.2 ロボットモデルを扱う基本機能

Body プラグインによって，ロボットや環境のモデルに対応する「Body アイテム」や，モデルの動作軌道を格納する「BodyMotion アイテム」が利用可能となる．また，これらをユーザが直接操作できるようにするため，Body プラグインは**図 4** に示すようなインタフェースも提供する．さらに，図 3 に示したシーンビューに対しても，Body アイテムのモデルを描画したり，ビュー上でマウスのドラッグ操作によりモデルの姿勢を変えるための機能が追加される．これらは Choreonod 上でロボットモデルを扱う際に基本となる機能であり，これらによってロボットモデルの読み込み，表示，位置／姿勢の変更，干渉検出，動作軌道データの再生といったことが可能となる．したがって，Body プラグインは 2 章の要求（2）を満たすにあたってベースとなる重要なプラグインである．なお，Body プラグインの中でも GUI から独立して利用可能な部分は実際には「Body ライブラリ」というライブラリに分離されており，そちらは一般のプログラムからも利用可能である．

3.3 シミュレーション機能

Body プラグインはシミュレーション機能も含んでおり，これによって図 1 にも示したように Choreonoid をシミュレータとして用いることが可能である．シミュレーション機能は OpenHRP3 [7] のシステムと物理計算エンジン [8] を Choreonoid 上で再構築したものであり，OpenHRP3 と同様のシミュレーションが可能である．さらに実行効率は OpenHRP3 よりも向上しており，シミュレーションの内容や実行環境にもよるが，2 足歩行ロボットの歩行シミュレーションで 2〜5 倍程度の高速化を確認している．OpenHRP3 は分散オブジェクト型のシステムであったが，Choreonoid 上への再構築にあたってはフレームワークの構造に倣いシングルプロセス・マルチスレッドのよりシンプルな構造とした．このことが効率向上の主要な要因となっており，2 章で示した設計方針（5）の利点を示す例となっている．

接続可能な制御プログラム（コントローラ）の形態はプラグインによって追加できるようになっており，OpenHRP プラグインによって OpenHRP3 用のコントローラと接続することも可能となっている．また，物理計算エンジンもプラグインによって追加可能であり，Open Dynamics Engine [9] を利用するための ODE プラグインも提供されている．このように拡張性も高いものとなっており，今後 RT コンポーネントや ROS ノードと直接接続するための機能の開発や，他の物理エンジンのテストも行っていきたい．

3.4 振り付け機能

PoseSeq プラグインにより，Choreonoid を動作振り付けツールとして使えるようになる．この利用例を**図 5** に示す．このプラグインは新しいアイテムとビューを一つずつ追加する．アイテムについては，ロボットの動きの元にな

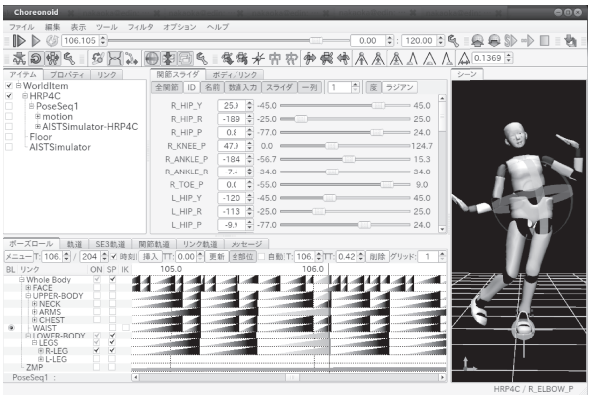


図 5 振り付け機能の利用例

る複数の姿勢（キーポーズ）を格納する「PoseSeq アイテム」が追加される．キーポーズはその姿勢をとるべきタイミングとセットで与えるようになっており，キーポーズ間の動きはそれらを補間した軌道として生成される．そして，このデータの表示と編集を行うビューとして，図 5 の左下に位置する「ポーズロールビュー」も追加される．これは横軸を時間軸，縦軸を身体部位として，キーポーズに対応する箇所を模式的に表したものであり，これを用いてキーポーズの追加，削除，および姿勢やタイミングの変更を行うことができる．

これは CG のキーフレームアニメーションと同様の仕組みであるが，2 足歩行ロボットが対象である場合，この仕組みだけでロボットが転倒しないバランスのとれた動きを作ることは難しい．この問題は Balancer プラグインによって解決される．これは PoseSeq プラグインの補助機能として動作するもので，キーポーズの姿勢とそれを補間した動きをバランスのとれたものに自動補正する．この補正はユーザがキーポーズを入力・変更する度に瞬間的に機能するので，ユーザは通常のキーポーズ編集と同様の感覚でバランスのとれた動きを作成できるようになる [10]．

以上の振り付け機能は十分実用的なものであり，これを用いることで，実際に 2 足歩行ヒューマノイドロボット HRP-4C がプロの振り付けでダンスと歌唱を行うデモも実現している [11]．そのような機能が，フレームワーク部や Body プラグインの部分を除けば，基本的には PoseSeq アイテム，ポーズロールビュー，およびバランス補正の 3 点のみの追加実装で実現していることになる．これは，Choreonoid では高度な機能をその機能の本質に集中して効率的に実装可能であることを示している．

3.5 その他の応用例

上で紹介したもの以外にも，Choreonoid 上で動作する応用機能として以下のような様々なものが開発されている．

実ロボット同期機能: Choreonoid 上のロボットモデルの動きをオンラインでロボット実機に転送し，両者の動きを同期させる機能．現在のところ，ホビーロボット G-Robots に対応した GRobot プラグインと，HRP-4C 等のロボットに対応した HRPSYS プラグインが開発されている．

メディア再生機能: Media プラグインによって，音声や動画のデータを読み込む Media アイテムと動画を表示するメディアビューが追加され，タイムバーと連動した音声や動画の再生を行うことが可能となる．

動作適用機能: モーションキャプチャで取得した人の動作軌道をロボットに適用する機能．インタラクションメッシュによる動作適用手法を拡張する手法 [12] によって実現しており，Mocap プラグインとして実装されている．

把持計画機能: アームの先端にハンドが取り付けられた構成のロボットに対して，把持計画，軌道計画，作業計画など，種々の計画問題を解くためのプラグイン一式が，辻，原田によって “graspPlugin” として開発されている．この機能については本特集で別途開発者による解説 [13] があるので，ぜひそちらも参照いただきたい．

これらの例からも，実際に Choreonoid を用いて多様な機能を実現できていることが分かる．もちろん，これらの機能は他の機能と連携させることが可能である．例えば，3.4 節で触れた HRP-4C のダンスデモを実行する際には，振り付け機能とメディア再生機能，実ロボット同期機能を連携させ，振り付けした動作を音楽と同期させながら実ロボットで実行するという操作をすべて Choreonoid で行った．また，上に挙げた動作適用機能や把持計画機能については，最先端の研究課題としてそれらの機能が開発された例であり，研究を進める試行錯誤の場としても Choreonoid が有効なことを示すものである．

4. アイテムツリーを介した機能の統合

多様な機能を統合するにあたって重要な役割を果たす仕組みの一つが，アイテムツリービュー上で管理されるアイテムツリーである．ここではその利用例について簡単に紹介する．

図 6 の (A) では，まず床のモデルである Floor と HRP-4C ロボットのモデル 2 体を World アイテムの小アイテムとして配置することで，これらが同一の仮想世界に属することを示している．干渉検出やシミュレーションは，この仮想世界ごとに行われることになるので，World アイテムを追加することで複数の仮想世界を同時に扱うことも可能である．Motion1〜Motion3 は動作軌道データを格納する BodyMotion アイテムであり，これらをロボットモデルの小アイテムとして配置することで，動作データとモデルの対応付けを表現している．ここでは 1 体めの HRP-4C に二つの動作データが関連付けられているが，Motion2 を選択状態とすることでそちらが処理対象であることも表現している．このようにして，モデルと関連したデータを扱う

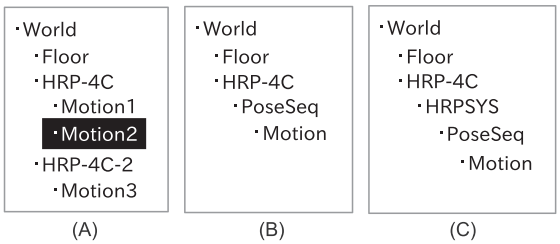


図 6 アイテムツリーの利用例

機能は，ツリーの状態のみからデータとモデルの対応関係や対象データの決定を行うことが可能となる．他の例として，シミュレーションにおけるモデルとコントローラ，動作結果データの関係も同様に表現可能である．

(B) では，キーポーズ列を格納する PoseSeq アイテムをやはり HRP-4C モデルと対応付けている．また，PoseSeq アイテムは自前で BodyMotion アイテム “Motion” を生成するようになっており，ここにキーポーズを補間した動作軌道が格納される．この場合，Motion も同時に HRP-4C モデルと対応づけられることになる．すると，BodyMotion アイテムを対象としたアニメーション，軌道表示，ファイル保存といった既存機能のすべてが，新たに追加したキーポーズデータの編集結果に対しても，追加のインタフェースや操作を必要とせずそのまま利用できることになる．

(C) ではさらに，動作データを実ロボットに転送するための HRPSYS アイテムを挿入している．HRPSYS アイテムは PoseSeq プラグインとは独立した HRPSYS プラグインにおいて実装されたものであり，BodyMotion アイテムを対象として新たに追加されたものである．この場合でも，このようにツリーを構成することで，キーポーズの編集結果をそのまま実ロボットに同期させることが可能となる．

以上のように，新たなデータ型や機能をアイテムとして追加し，アイテムツリー上で他のアイテムとの適切な位置関係をもたせるだけで，追加した機能と既存機能とのスムーズな連携が可能となるのである．

5. プラグインの記述

プラグインの実際のプログラムコードの例を**図 7** に示す．このプラグインは二つのボタンを追加し，それぞれのボタンを押すことで対象ロボットモデルのすべての関節角を一定値増加／減少させるというものである．このような簡潔な記述で，「対象モデル／データを特定しユーザの操作によってその内容を更新し，結果をすべての関連インタフェースに反映させる」ということが実現できている．

5〜15 行めではプラグインの初期化用関数である initialize 関数をオーバーライドし，ボタンの追加を行っている．まずボタンを格納するツールバーを 6 行めで作成し，これに対して 7 行め，10 行めでそれぞれ “Increment”，“Decrement”

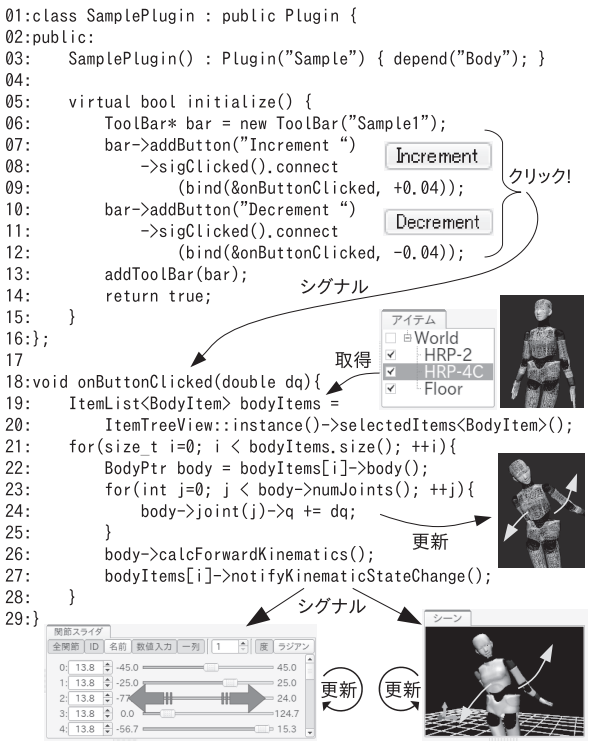


図 7 プラグインの記述例．なお，ヘッダファイルのインクルードや名前空間の指定等は省略している

というラベルのついたボタンを追加している．ボタンはそれが押されたときに “sigClicked” という「シグナル」を発行するようになっている．ここでは 8〜9 行め，11〜12 行めで，それぞれこのシグナルに “onButtonClicked” という関数を結びつける記述を行っている．

この記述により，ボタンが押されると 18〜28 行めに記述された処理が実行される．まず 19〜20 行めにおいて，対象となるロボットモデルをアイテムツリービューから取得する．ここではユーザが選択状態にしているアイテムを返す “selectedItems” 関数に，ロボットモデルに対応する Body アイテム型をテンプレート引数として与えて呼び出すことで，利用中の複数のアイテムの中から対象アイテムのみを抽出したリストを得ている．図の例では HRP-4C モデルのアイテムが対象となっている．

取得した Body アイテムのリストに対して 21 行めからループを回し，モデル本体に対応する Body 型のオブジェクトを取り出して処理を行っている．処理の本体は 23〜26 行めで，モデルのすべての関節に対して，関節角度に対応する変数 “q” を引数 “dq” の分だけ加算し，変更後の関節角度からリンク位置を更新する順運動学計算を行っている．なお，ここでの引数 dq はもともと sigClicked シグナルと関数 onButtonClicked を結びつける際に追加で与えたパラメータ “+0.04” もしくは “−0.04” に対応している．

最後に，27 行めで更新後の Body アイテムに対して関数

“notifyKinematicStateChange” を呼んでいる．これにより，Body アイテムがもつ “sigKinematicStateChange” というシグナルが発行され，このシグナルに結び付けられている関数があればそれらがすべて実行される．実際に，干渉検出といったモデルの状態と関連する内部処理や，モデルの状態を表示するシーンビューや関節スライダビューといったインタフェースの更関処理は，それぞれこのシグナルとの結びつけがあらかじめ行われており，シグナル発行時にモデルの最新の状態を反映する処理が行われることになる．

このコードをアレンジするだけでも様々なことが可能である．例えば，ボタンクリックの代わりにタイマが発行する周期的なシグナルを処理関数と結びつけるようにし，処理関数において Kinect から取得した関節角で変数 q を更新するようにすれば，ユーザの姿勢と連動したロボットモデルのアニメーションも実現できるだろう．

以上のように，プラグインの開発においては，シグナルを自前の関数と結びつけたり，自前の関数でシグナルを発行したり，さらには自前のシグナルを定義することで，既存の機能や今後追加される機能との連携を簡潔に記述することが可能となっている．

6. おわりに

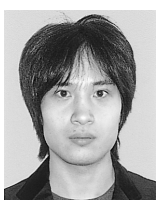
本稿では Choreonoid について，「拡張可能なロボット用統合 GUI 環境」としてどのようなことを実現可能かを紹介してきた．本稿ではその設計要素のすべてを紹介することはできなかったが，Choreonoid が今までにない領域をカバーしつつも十分実用的なフレームワークとなっていることを感じていただければ幸いである．

Choreonoid は 2011 年 11 月より一般公開されており，オープンソースソフトウェアとしてソースも含めて誰もが自由に利用可能となっている [14]．また，Choreonoid はマルチプラットフォーム設計としており，すでに Windows, Linux, OS X をサポートしているため，ユーザの利用 OS に合わせて比較的手軽に導入することが可能である．公開後一般ユーザによる利用事例やプラグイン開発事例も増えつつある．この流れを発展させ多くのユーザが利用する標準プラットフォームとなることを目指して，今後も Choreonoid の更なる開発を進めていきたく思っている．

謝　辞　Choreonoid のベースとなる部分の開発は次世代ロボット知能化技術開発プロジェクト（平成 19〜23 年度）の一環として実施したものである．ここに感謝の意を表する．

参考文献

[1] 金広，藤原，梶田，横井，金子，比留川，中村，山根：“ヒューマノイドロボットソフトウェアプラットフォーム OpenHRP”，日本ロボット学会誌，vol.21, no.7, pp.785–793, 2003.
[2] O. Michel: “Cyberbotics Ltd. Webots: Professional mobile robot simulation,” Inernational Journal of Advanced Robotic Systems, vol.1, no.1, pp.39–42, 2004.
[3] Gazebo, http://gazebosim.org/
[4] S. Nakaoka: “Choreonoid: Extensible virtual robot environment built on an integrated GUI framework,” Proceedings of the 2012 IEEE/SICE International Symposium on System Integration (SII2012), 2012.
[5] 安藤：“初心者のための RT ミドルウェア入門—OpenRTM-aist-1.0 とその使い方—”，日本ロボット学会誌，vol.28, no.5, pp.550–555, 2010.
[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T.B. Foote, J. Leibs, R. Wheeler and A.Y. Ng: “ROS: an open-source robot operating system,” Proceedings of the IEEE International Conference on Robotics and Automation Workshop on Open Source Robotics, 2009.
[7] 中岡，山野辺，比留川，山根，川角：“分散コンポーネント型ロボットシミュレータ OpenHRP3”，日本ロボット学会誌，vol.26, no.5, pp.399–406, 2008.
[8] S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita and H. Hirukawa: “Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms,” Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.3641–3647, 2007.
[9] Open Dynamics Engine, http://www.ode.org/.
[10] S. Nakaoka, S. Kajita and K. Yokoi: “Intuitive and flexible user interface for creating whole body motions of biped humanoid robots,” Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.1675–1682, 2010.
[11] 中岡，三浦，森澤，金広，金子，梶田，横井：“ヒューマノイドロボットのコンテンツ技術化に向けて—クリエイターによる多様な表現の創出が可能な二足歩行ヒューマノイドロボットの実現—”，シンセシオロジー，vol.4, no.2, pp.80–91, 2011. http://www.aist.go.jp/aist_e/research_results/publications/synthesiology_e/vol4_no2/vol04_02_p87_p98.pdf
[12] S. Nakaoka and T. Komura: “Interaction mesh based motion adaptation for biped humanoid robots,” Proceedings of the 12th IEEE-RAS International Conference on Humanoid Robots, pp.625–631, 2012.
[13] 辻，原田：“graspPlugin for Choreonoid”，日本ロボット学会誌，vol.31, no.3, pp.232–235, 2013.
[14] Choreonoid Official Site, http://choreonoid.org/



中岡慎一郎（Shin’ichiro Nakaoka）
2001 年東京工業大学理学部情報科学科卒業．2006 年東京大学大学院情報理工学系研究科コンピュータ科学専攻博士課程修了．博士（情報理工学）．2006 年 4 月より産業技術総合研究所知能システム研究部門研究員，現在に至る．2011 年 11 月より 1 年間英国エジンバラ大学客員研究員．ヒューマノイドロボットによる動き提示，ロボットソフトウェアプラットフォーム等の研究に従事．2008 年度日本ロボット学会論文賞，IEEE/SICE SII2012 Best Paper Award（Robotics）受賞．　（日本ロボット学会正会員）