I_002

Java3Dによるリンク連鎖機構の3次元モデリングツールの開発
Development of 3D Modeling Tool for Kinematic Chain with Java3D

古賀 雅伸 † 森宗 翔吾 †
Masanobu Koga Shogo Morimune

1. はじめに

制御系開発のプロセスは制御対象のモデリング、制御対象の設計、シミュレーション、実装の順序で行う。しかし、シミュレーション結果の大半はグラフや数値によって表現されるため、制御対象がどのような動きをしているのかが把握しにくい。このシミュレーション結果が不明確なまま実装を行うと、機材の破損やそれに伴う費用などが生じてしまう。

そこで本研究では、Java3DによるCGで制御対象を生成し、時間ごとにおける制御対象の各部の位置情報を元に描画したCGを動かし、アニメーションを行うことでシミュレーション結果を表現するツール、JAMAST (Java3DModelingAndSimulationTool) の開発を行うことを目的としている。

また本研究では、リンク連鎖機構を主な制御対象としている。

2. Java3Dによる3次元モデリング

ツールの開発にはプラットフォーム非依存であるJava言語を使用し、3次元描画の実現にはJava3D[1][2]を使用する。

3次元描画に関する既存の3Dグラフィックス APIとしてOpenGLやDirectXが存在する。OpenGLにおけるモデリング・レンダリング方法は、三角形や四角形のポリゴンを繰り返し生成し、それらを組み合わせることでモデルの構築を行うもので、ポリゴンの生成にはポリゴンの頂点となる座標をメソッドの引数として正確に入力する必要がある。複雑な形状や曲面も複数のポリゴンで分割することにより、表現が可能となっている。したがって、この既存の3Dグラフィックス APIではポリゴンの頂点の座標を正しく定めることが重要となっている。

一方、Java3Dにおけるモデリングの特徴として、Javaのプラットフォーム非依存性だけでなく、モデル間の構成にシーングラフを採用していることがあげられる。シーングラフとは物体や光源、視点といった3次元グラフィックスに必要なデータを木構造で表現したデータベースである。これにより、この木構造の作成が物体の描画に繋がるため、リンク連鎖機構を取り扱いやすい。Java3Dで定義されているいくつかの基本的なクラスのメソッドを使用することで、モデルやシーングラフを作成および操作し、表示とレンダリングの制御を行っている。またJava3Dは、OpenGLやDirectXより上に位置づけられており、こちらから見えないところでOpenDLなどの内容を使用している。

シーングラフにおいて、シーングラフの変化が表示される絵に影響するので、各物体の頂点の移動後の座標の計算をせずに、変化させる部分の節を移動させるのみで済む。シーングラフにおいて上位のノードの影響を下位のノードが受け継ぐことから、光源や視点などの不可視の存在のオブジェクトを描画する際、シーングラフのルートからそのオブジェクトに至るパス上にある各ノードで行われたグラフィックス状態の変更を全て組み合わせることがレンダリングへとつながる。

そして、本研究で開発したJAMASTでは、モデルのパラメータやモデル同士の構成をXMLで表現と保存を行うため、Casterを使用してXML文書の構造をそのままJavaのクラスにマッピングしている。マッピングされたクラスでモデルや座標や回転に関する値を設定するメソッドを作り、そのメソッドを入力ダイアログを通じて、各パラメータの設定を行う。レンダリングも同様に、入力ダイアログを通じて前述にあげたJava3Dのレンダリングを実現する。

3. JAMAST

JAMASTには以下に示す新規性、有用性、信頼性があると考えられる。

まず新規性において、図1に示すようにモデル描画画面の隣で、Java3Dのシーングラフを元にモデル間の構成をツリーで表現しているだけでなく、ツリーにはモデルのパラメータが表記されているので、モデル間の構成とモデルの形状と内容を同時に確認できることがこのツールの新規性にあたるといえる。

次に有用性は、Java3Dで構成されているので、モデルの頂点について考慮する必要がなくなり、コード数も減少できることから、特にモデルの移動お呼びか移転時の効率の上昇が期待できる。

最後に信頼性に関しては、モデリングデータをXMLによって保存しているので、各モデルのパラメータが理解しやすく、モデル同士の構成がXMLとツールのツリーは酷似しているので、データの操作や管理上高い信頼性を持つといえる。

JAMASTのその他の詳細は、以下に記述する。

3.1 構成

図1に示すのがJAMASTの実行画面で、画面左が現在のモデリング状態を表す「キャンバス」で、画面左がそのモデリングのシーングラフを表す「ツリー」である。

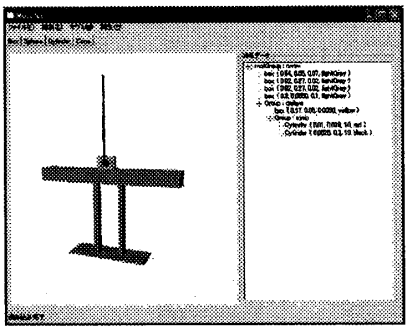

図 1: JAMAST 実行画面

†九州工業大学

3.2 モデリング

本研究で開発しているJAMASTはプリミティブと称する使用頻度の多い基本図形、直方体、円柱、円錐、球体を生成し、接続することでモデリングを行う。

プリミティブの生成方法は、ツリーに形状や位置、回転に関するパラメータを直接入力して生成するものと、ツールバーで、デフォルトの値で設定された各パラメータから生成するものがある。プリミティブの編集は、ツリーをクリックすることで種類以外変更できる。

キャンバスで生成した制御対象のモデルに、実時間ごとにおける制御対象各部位置および回転情報のデータを読み込むことで、制御対象の動きをCGで再現する。

3.3 プリミティブの重複防止

プリミティブの生成において、生成を指定した座標次第では、すでに生成しているプリミティブ同士と重複してしまうことがある。いずれプリミティブ同士を接続して制御対象のモデリングを行うが、制御対象の各部を正確にモデリングに対して、プリミティブの重複は妨げとなる。

意図的にプリミティブ同士の重複を行うとき以外で、重複が生じると描画座標を編集しなおさなくてはいけない上、編集後に別のプリミティブと重複することも考えられるため、重複しない座標を計算しつつ編集を行うことは非常に効率が悪い。

そこで、本ツールでは新しくプリミティブを生成すると同時に、現在描画しているプリミティブと重複しないか判定を繰り返し、重複しない座標を自動算出し、そこへ生成することができる。

3.4 コネクタを用いたプリミティブの接続

本ツールでは、プリミティブを接続していくことで、リンク連鎖機構をはじめとした制御対象のモデリングを行っている。ここでモデリングしたCGを用いてシミュレーションを行うため、制御対象の構造を正確に再現する必要がある。

しかし、平面上で立体空間を表現することにおいて、正面から見た場合では奥行きの距離感がつかみにくいことや、プリミティブの回転や視点の位置によって死角が生じるなど、物体同士を接続することが困難になる状況が生まれる。この状況下で、プリミティブ同士の表面を正確に接続するためにプリミティブの座標の計算することは非常に困難である。

そこで、本ツールではプリミティブの接続を行うコネクタを用いる。コネクタを使用する状態に移行すると、描画画面上のプリミティブに種類ごとに指定された位置にコネクタが生成される。コネクタをクリックすると、プリミティブ同士が接続される。コネクタによる接続の様子は図2に示す。

また、コネクタはユーザの必要に応じて、生成と消去の切り替えを指示できる。

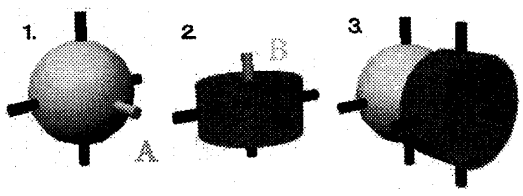

図2: コネクタ処理概要

1. 接続の基点となるコネクタAを選択する
2. コネクタAのところへ移動させるコネクタBを選択する
3. コネクタBの角度をコネクタAと平行にした後、コネクタAのもとへコネクタBをプリミティブごと移動させる。

3.5 XMLデータ保存

モデルのデータ保存にはXMLを使用している。本研究では、プログラム内からのXMLデータの読み込み、書き込み、編集を行うことのできるこのツールのスキーマ専用のJavaAPIを作成しており、このAPIを使用することによりJavaプログラム内で作成したXMLデータの読みとり、保存が可能となる。

3.6 アニメーション

JAMAST実行画面のメニュー「再生」で、図3に示すアニメーションの実行画面が表示される。「参照」ボタンでアニメーションのためのデータファイルを読み込み、再生ボタン(左端)で制御対象モデルのシミュレーションを、モデルのアニメーションから確認できる。

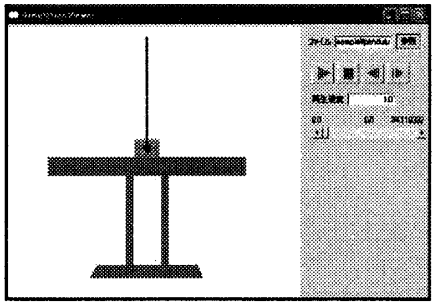

図3: モデルのアニメーション実行画面

4. 終わりに

コネクタを利用することで、コネクタが配置されているところに接続したい場合、詳しい値を計算および入力する手間が省ける。また、プリミティブの重複を許可しない場合、重複しない座標へ自動で移動できるので、生成後に手動で座標を入力する必要がなくなる。

したがって、この二つの機能を併用することで、制御対象のモデリングがスムーズに行うことができるようになるといえる。

今後の課題として、保存したモデリングデータを現在の描画画面に追加させることを目標としている。この機能を実装させることにより、複雑な形状のモデルがプリミティブ同様に生成しやすくなることで、モデリングの幅が広がることや効率の向上させることが望める。

参考文献

[1] えんどう やすゆき, 平鍋健児「Java3D プログラミング バイブル」(ナツメ社 2003)

[2] Henry Sowizral, Kevin Rushforth, Michael Deering（翻訳：竹内里佳)「The Java3D API 仕様」（株式会社 アスキー 1999)

[3] 古賀美希「Java3Dによるリンク連鎖機構のモデリング及びアニメーションツールの作成」(2004年)