4Y-3

リアルタイム3DCGプログラミングにおける ビジュアルプロファイラの有意性に関する研究

竹内亮太† 渡辺大地††

†東京工科大学大学院メディア学研究科 ††東京工科大学メディア学部

1 はじめに

近年リアルタイム3DCGを用いたコンテンツを制作する上で、クラスライブラリを用いたプログラミングを行う機会が増えている。しかし、ライブラリを用いて3DCGシーンを構築する場合、ライブラリが提供するメソッドが実際にどのような挙動を示すかが分からず、期待通りの動作が得られないことがよくある。またプログラムの規模が大きくなると、シーン内の3次元モデルに対してどのような操作を発行したかが煩雑になり、不具合の原因を発見するのが困難になる。

このように3DCGシーンを構築するプログラミングにおいては、3次元モデルに対する制御命令の動作を履歴として記録し、確認出来る仕組みが存在すれば有用であると考えられる。一般的なプログラミングにおいて動作を解析する場合には、デバッガなどのツールを使用することが多い。しかしデバッガツールの多くは、プログラムが今まさに実行する処理を制御するものであるため、一定の処理が完了した状態での処理結果と、その時点までに実行した処理の内容を確認したいというニーズに対応出来る物ではない。ライブラリ側で処理の履歴を記録する機構を用意すれば、ユーザープログラムが実行した処理のUndo/Redoが実現出来るため、処理の流れと実際の表示を対応づけた可視化が可能となる。プログラム上の複数の箇所から3次元モデルを制御している場合でも、履歴情報から操作の発行元と内容が明確になるため、3DCGシーン構築及びデバッグの効率的な支援が可能になると考えられる。

そこで本研究では、3次元モデルに対する処理の履歴を3DCGシーン上で可視化するシステムを開発し、既存の3DCGクラスライブラリでの開発を支援する環境を提案する。なお本研究においては、処理の履歴を監視して記録する機構をプロファイラと呼称しており、一般的なホットスポットを検出するツールとは異なる物であることに注意されたい。

Research on the visual profiler in real-time 3DCG programming

† Ryota TAKEUCHI

†† Taichi WATANABE

Graduate School of Media Science, Tokyo University of Technology (†)

Faculty of Media Science, Tokyo University of Technology (††)

2 既存のライブラリへのプロファイラの導入

本研究ではプロファイリングの対象を、3次元モデルに対する座標変換操作を行うメソッドとする。3次元座標変換操作とは、3次元座標ベクトルと4×4正方行列を用いて表す、拡大縮小・任意軸回転・平行移動の諸操作を指し、既存の3DCGライブラリが一般的に実現している機能である。本研究では実装例として、OpenGLベースの3DCGクラスライブラリであるFK System[1]を使用し、3次元座標変換操作を行うメソッドに対して履歴情報を記録する機能を付加した。

プロファイラを導入するに当たっては、オブジェクト指向プログラミングにおける継承関係を利用する。ライブラリのユーザーが利用するインターフェースは変更せずに、基底クラス→実装クラス→プロファイラクラス→インターフェースというように、インターフェースの前に追加する機能のクラスを挟んだ継承関係を構築する。このように継承関係を設計に用いることで、既存のライブラリのコードに対する変更を最小限に抑えると共に、利用者側は既存のライブラリと同じ利用方法のまま、プロファイラを利用することが可能となる。以下の図1は、プロファイラを導入する際のクラスの継承関係を示す。左側が、既存のライブラリにおける3次元モデルを表すクラス図で、右側はプロファイラを導入したクラス図である。

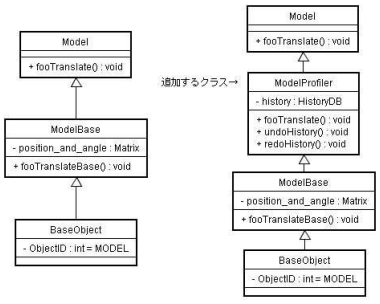

図 1: プロファイラを導入したクラスの継承関係

3 コマンドヒストリーによるUndo/Redo機構

取得する履歴情報は、実行したメソッドのIDと引数を配列に蓄積していくことで記録する。3次元座標変換操作のメソッドは、3次元モデルの状態に対して

変化量を与えるタイプと、座標や方向ベクトルを直接代入するタイプに大別することが出来る。このため、代入操作を行うメソッドの場合は、引数と共に書き換える前の値も保存する必要がある。以下の表 1は、実際に記録するメソッドの処理内容と引数の例を示す。

表 1: 履歴情報として記録するメソッドの例

メソッドと引数	処理内容
Scale(スケール)	スケール代入
Rotate(座標, 軸, 回転角)	任意軸回転
Angle(オイラー角)	姿勢の直接代入
Translate(移動ベクトル)	平行移動
MoveTo(位置ベクトル)	座標の直接代入

こうして記録した履歴をコマンドヒストリーと呼称する。コマンドヒストリーには実行した処理内容が全て記録されているため、この情報を利用することで逆操作を定義することが出来る [2]。平行移動や回転の場合は、移動ベクトルや回転角の符号を反転させることで逆操作となる。スケールや方向ベクトルを代入する操作の場合は、あらかじめ保存してある代入前の値を復元することで逆操作とする。順操作は記録したコマンド通りの処理をそのまま行うだけで実現できる。以上の操作を定義することにより、コマンドベースで履歴を自由に巻き戻したり進め直したりすることが可能となる。

コマンドヒストリーには Undo/Redo に必要な情報以外に、メソッドのコール元、実行ステップ数などをデバッグ情報として付加することが出来る。デバッグ情報を付加する必要がない場合は、ユーザーは既存のコードに対してほとんど手を加えずに、Undo/Redo 機構を利用することが出来る。

4 ヒストリーブラウザ

実装した Undo/Redo 機構は、ヒストリーブラウザと呼称するインタフェースを介して制御することが出来る。ブラウザはライブラリのコードとして埋め込まれており、実行中にユーザーが任意のタイミングで呼び出すことが出来る。ブラウザを起動した時点でユーザープログラムの実行を中断し、ターゲットとした 3 次元モデルの履歴を操作するモードに移行する。

ブラウザには、ユーザープログラムからターゲットの 3 次元モデルに対して発行した操作の履歴を表示し、任意のステップにおける表示状態をユーザープログラムウィンドウ上で確認することが出来る。以下の図 2は、実際にユーザープログラム上でヒストリーブラウザを使用している様子を示す。

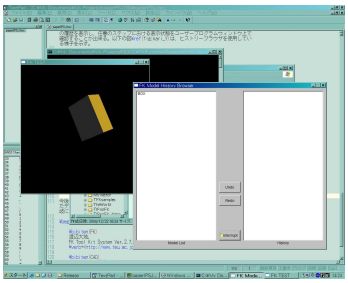

図 2: ヒストリーブラウザーのスナップショット

現状では、履歴の記録は 1 次元の連続的な配列データとして処理しているため、履歴を巻き戻した状態でブラウザを終了した場合は、巻き戻したステップから先の情報は破棄している。将来的にはツリー構造を持った履歴情報に対応することで、異なるパターンの操作履歴を並列的に記録して、複雑な操作のテストが行えるであろう。

履歴情報はテキストドキュメントとしても出力可能であり、デバッグ情報として利用出来るのはもちろん、今後本手法を発展させていく上でも役立つ機会が多いと思われる。

5 まとめ

本研究では、既存の 3DCG クラスライブラリに対してプロファイラ機構を導入し、3 次元モデルの座標変換操作におけるコマンドベースの Undo/Redo と、ユーザープログラム上から履歴を操作可能なヒストリーブラウザを実装した。その結果、プログラムが実行したメソッドと実際の表示の対応をインタラクティブに確認出来るようになった。

本手法は、既存のデバッガツールで行うような実行ステップを制御するシステムとは異なり、プログラムが一定の処理を完了した段階で、実行した処理結果を振り返って確認することが出来るため、ユーザーが意図する処理の単位で実行結果を確認することが可能となり、開発支援に非常に有用であると言える。

今後の課題としては、座標変換操作以外のメソッドへの対応や、トランスレータを用いたソースに対するデバッグ情報の自動的な付加、履歴を巻き戻した時点から新たな操作を行った場合の分岐に対応することによる、並列的な履歴管理の実現などが考えられる。

参考文献

[1] 渡辺大地, FK Tool Kit System Ver.2.7.7, <http://www.teu.ac.jp/aqua/~earth/FK/>.

[2] 鳥谷浩志, 千代倉弘明, ３次元ＣＡＤの基礎と応用, 共立出版, 1991.