Scheme言語における 3次元グラフィクス・インタフェースの実現

1Q−3

村上幸司　白崎昌俊
(株)沖テクノシステムズラボラトリ

1 はじめに

3次元グラフィクス・プログラミングでは、3次元物体及び光源に関して、定義すべき項目が非常に多いため、それらの項目の変更などが頻繁に行われる。また、物体の形状は、モデリング等で動的に変化するため、頂点や線分等のデータを動的かつ柔軟に操作する必要がある。

そこで、プログラムの変更が容易で、データを動的かつ柔軟に扱うことができるScheme言語環境で3次元グラフィクス・インタフェース機能を実現することによって、動的かつ柔軟性の高いグラフィクス・プログラミング環境の構築を実現した。以下、この実現方法について述べる。

なお、本システムはi860[*1]をメインCPUとして搭載したOKIstation7300上において実現した。

2 機能

本システムは、3次元グラフィクス処理機能をScheme言語環境で利用可能とするためのシステムである。ここでは、Scheme環境が持つ特性を活用し、図形の定義や動作、描画などの定義を対話的にかつ効率良くプログラミングすることができる。

2.1 Scheme言語環境下での3次元物体のリアルタイム描画機能

この機能は、Scheme言語環境とC言語環境との間でのデータの受渡しの環境を用意することによって実現可能となったものである。これにより、Scheme言語環境下で容易にリアルタイムの3次元グラフィクス処理を行うことが可能となった。

2.2 Scheme言語環境下での入力イベント処理機能

イベント処理とは、3次元グラフィクス・ライブラリが備えている、マウスからの座標入力等の各種入力機能である。この機能は、C言語環境からScheme言語環境へのデータ転送を実現することによって実現可能となったものである。これにより、Scheme言語環境下における対話的なアプリケーションの作成をより容易なものにすることが可能となった。

3 実現

3.1 インタフェース構成

インタフェース構成を図1に示す。以下、アプリケーション・プログラムに近いものから順に説明する。

3.1.1 ユーザ・プログラム

ユーザがScheme言語で記述するグラフィクス・アプリケーション・プログラム。

3.1.2 Scheme言語インタフェース

ユーザからの指示を直接受け取る部分。

3.1.3 Scheme ↔ C言語間Gate関数

C言語とScheme言語との間でデータの受渡しを行うための部分。Scheme側でのsys:graphicsが、C言語ではs_graphicsに対応する。

3.1.4 C言語インタフェース

Scheme言語の内部データとC言語のデータとの変換を行い、3次元グラフィクス・ライブラリ関数を呼び出す。

3.1.5 3次元グラフィクス・ライブラリ

C言語仕様の3次元グラフィクス処理関数ライブラリ。3次元グラフィクス処理を行う実体である。

3.2 実現方法

3.2.1 Scheme言語とC言語との間の言語接続

本システムでは3次元グラフィクス処理部として、C言語ライブラリを呼び出す方法を採用している。そのため、Scheme言語環境とC言語環境との間において、データや関数に対して整合性を取っている。これによって、Scheme言語環境の拡張が容易にできるようになった。

Implementation of 3D Graphics Interface for **Scheme** Language
Koji Murakami , Masatoshi Shirasaki
Oki Technosystems Laboratory, Inc.

3.2.2 Scheme 言語と C 言語の間の変数型の対応

Scheme 言語、C 言語の間の値の受渡しに際して問題となるのが、この変数型対応である。主に問題となるものは C 言語における配列や浮動小数点数等をどのように Scheme 言語に対応させるかである。

配列に対応させることができるものには、vector がある。変換マトリクスは、1 次元の配列として扱っている。

3.2.3 イベント処理

イベント処理で、3 次元グラフィクス・ライブラリで発生したイベントに対応して、値を Scheme 言語環境に転送する。このようなイベント値の受渡しには vector を用いている。

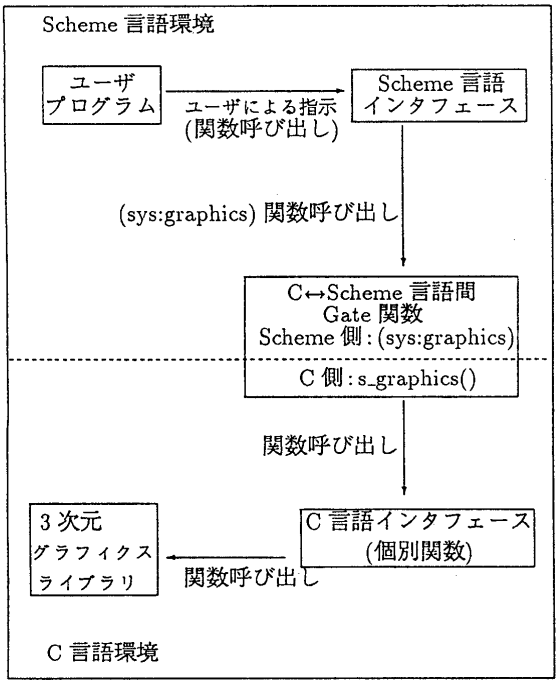

図 1 インタフェース構成及びデータの流れ

4 プログラム例

```
; パラメータの定義。n0,p0,p1,p2,p3 の 5 つの vector をグローバル定義している。
 1:(define n0 #(0.0 0.0 1.0))  ; 法線ベクトルの定義
 2:(define p0 (make-vector 3 0.0)) ; 座標の定義 (1)
 3:(define p1 (make-vector 3 0.0)) ; 座標の定義 (2)
 4:(define p2 (make-vector 3 0.0)) ; 座標の定義 (3)
 5:(define p3 (make_vector 3 0.0)) ; 座標の定義 (4)
   ;4 点を (0.0,0.0,0.0) に設定する。

   ; ユーザ定義による四角形定義関数
 6:(define (draw-plate n n2)    ;n2 = n * 2
   ;n = polygon 分割数のパラメータ。(n*n) 分割される。
 7:    (do ((i 0 (+ i 2)))
 8:        ((>= i n2))
       ;4 点の x 座標の決定
 9:     (vector-set! p0 0 (- (/ i n) 1.0))
10:     (vector-set! p1 0 (- (/ i n) 1.0))
11:     (vector-set! p2 0 (- (/ (+ i 2) n) 1.0))
12:     (vector-set! p3 0 (- (/ (+ i 2) n) 1.0))
13:     (do ((j 0 (+ j 2)))
14:         ((>= j n2))
          ;4 点の y 座標の決定
15:       (vector-set! p0 1 (- (/ j n) 1.0))
16:       (vector-set! p1 1 (- (/ (+ j 2) n) 1.0))
17:       (vector-set! p2 1 (- (/ (+ j 2) n) 1.0))
18:       (vector-set! p3 1 (- (/ j n) 1.0))
19:       (bgnpolygon)     ; polygon の定義開始
   ; 法線ベクトルの設定     座標値の設定
20:       (n3f n0)         (v3f p0)    ; (1)
21:       (n3f n0)         (v3f p1)    ; (2)
22:       (n3f n0)         (v3f p2)    ; (3)
23:       (n3f n0)         (v3f p3)    ; (4)
24:       (endpolygon))))  ; polygon の定義終了

   ; ユーザ定義による四角形表示関数
25:(define (set-action count)
   ;count = 実行回数。四角形を回転させつつ表示する
26:  (do ((i 0 (+ i 1)))
27:      ((>= i count))
28:    (cpack 0)   (clear) ; 黒で塗り潰す
29:    (rotate 5 #\z) ; z 軸について回転させる
30:    (rotate 5 #\y) ; y 軸について回転させる
31:    (draw-plate 10 20)
32:    (displaybuffer))) ; ディスプレイに表示する
```

5 おわりに

Scheme 言語環境において 3 次元グラフィクス処理を行うためのインタフェースを用意することで、高効率、対話的かつ柔軟性の高い開発環境を実現することができた。

今後の課題は、インタフェース部分の処理の最適化を行うことである。

参考文献

1. IEEE Standard for the Scheme Programming (1990).

2. 宇田川 他: Scheme 記述を用いた CG アニメーションシステム MARS の開発, 情報処理学会第 43 回全国大会, 1P-9.

3. 白崎 他: i860 の特長を生かした基本グラフィックスライブラリ BGL の開発, 情報処理学会第 43 回全国大会, 7U-3.

*1 i860 はインテル社の登録商標である