C++を拡張したアニメーション記述言語とその処理系

1D−7

安倍広多　松浦敏雄　谷口健一

大阪大学 基礎工学部 情報工学科

1　まえがき

近年, ビットマップディスプレイを備えた汎用のワークステーションが急速に普及してきており, これらを用いたCAI[1]やプレゼンテーションシステム[2]として, 計算機の画面上での簡易アニメーションが注目され始めている. これらのアニメーションでは, アニメーションの実行時に, マウスやキーボードなどのユーザからの入力を受け付けることができ, それによって, 画面の表示内容を変更できるような対話的な使い方も要求される. また, 画面上でのオブジェクトの動作速度を指定できる機能なども要求される.

筆者の所属する研究室では, このような, ワークステーション上での対話型かつ実時間のアニメーションを行なう方法を提案し, そのためのシステムRASを試作している[3]. RASのシナリオでは, ある適当な時間毎に区切って, その間における個々のオブジェクトの動作(直線移動や, 回転などの予め用意してある基本動作で記述できるもの)をすべて指定する必要がある(従って, 各オブジェクトごと独立にそのオブジェクトの一連の動作を記述することはできない)等, 必ずしもその記述は容易ではなかった. また, RASのシナリオ記述言語は, シナリオを記述するために特別に用意した言語であり, 変数や基本的な制御構造等の計算機言語としての基本機能を備えてはいるものの, 汎用的な高級言語ほどの記述能力は有していなかった. そのため, アニメーション表示を伴う任意のアプリケーションプログラムをRASの枠組みの中で記述することは容易ではなかった.

本研究では, このような制約なしに各オブジェクト毎の動きを自然に記述でき, かつ複数のオブジェクトの並列動作などを容易に記述できるアニメーションの記述法を提案する. また, その言語処理系, 並びに, アニメーションの実行系の実現法を示す.

2　アニメーション記述言語

本システムは, 2次元のアニメーションを対象としている. アニメーションに登場する物体(以下アニメーションオブジェクト, または単にオブジェクトと呼ぶ)の定義, 動きはアニメーション記述言語で指定する.

アニメーション記述言語には以下の条件が要求される.

1. 個々のオブジェクトの動作記述が容易であること. — オブジェクトに対する等速直線運動や角速度一定の回転等の基本動作の指示が容易に行なえ, また, 特別な動作指定(加速しながら移動する等)をも記述できること.
2. 複数のオブジェクトの並行動作, 逐次動作の指示が容易であること.
3. オブジェクト同士が重なったときの前後関係を容易に記述できること.
4. オブジェクト同士の関係(衝突したことや, 距離が一定以内に接近したこと等)を検出できること.

オブジェクト指向言語を用いると, クラスの継承の機能によって, 特別な動作の指定等をサブクラスのメソッドとして記述することができ, 条件1.を容易に満たすことができる.

そこで, 本システムでは, オブジェクト指向言語であるC++を基に, 条件2.,3.を満たすために拡張した言語(以下PC++と呼ぶ)と, アニメーションのためのクラスライブラリを用意し, これらを用いてアニメーションを記述する.[1]

2.1　PC++

PC++はC++に並行動作を記述する機能を付加したものである.

PC++では, C++の複文({ ... })を記述できる場所に, スレッド文(@{ ... @})と呼ばれる特別な複文を記述することができる. 概念的にはスレッド文は@{から@}までの処理を新たなプロセス[2]で実行することを意味する. スレッド文の次の(C++の)文の実行は, スレッド文の実行が終了するまで待機させられる. また, スレッド文の中にさらにスレッド文を書くことができる. 複数のスレッド文を並行動作記述子"&"または"|"で連結することができ, 連結されたスレッド文は並列に実行される[3]. "&"では, 連結された2つのスレッド文が共に終了するまで待ち, 引き続く文はその後に実行される. 一方"|"では, 連結された2つのスレッド文のどちらかが終了するまで待つ.

2.2　アニメーションクラスライブラリ

全てのオブジェクトに共通する性質(表示, 移動, 座標系の回転など)を定義するため, AnimationObjectクラスを用意した. すべてのアニメーションオブジェクトは, AnimationObjectクラスの導出クラスのインスタンスである. また, AnimationObjectクラスの導出クラスとして, 任意に指定したビットマップを表示するオブジェクトを定義するBitmapObjectクラス, 文字列を表示するオブジェクトを定義するFontObjectクラスを予め用意している. ユーザはこれらのクラスを用いることで簡単にアニメーションオブジェクトを定義できる. この他に, AnimationObjectクラスは4節で述べるスケジューラスレッドやアニメーションスレッドの関数定義を含んでいる.

2.3　アニメーションの記述例

以下, 簡単なアニメーションの記述例を図1に示す. この例では、画面中央の太陽の回りを地球が, 地球の回りを月がまわるアニメーションを記述している。

6行目のMakeAnimationWorldでアニメーションのクラスを初期化する. 引数は, ウインドウの幅, 高さ, 及び, 1フレームの時間(秒)である. ここでは400×400のウインドウを開き, 1フレームの時間は0.05秒と指定している.

7–9行目では, BitmapObjectクラスのインスタンスとして, 太陽(Sun), 地球(Earth), 月(Moon)を定義している. それぞれのイメージは引数で与えたビットマップファイルに格納されている. 太陽は絶対座標系に, 地球は太陽の座標系の上に, 月は地球の座標系の上に載せる. 10–14行目のスレッド文では, (絶対)座標(200,200)

An Animation Description Language and its Implementation based on C++

Kota ABE, Toshio MATSUURA, Kenichi TANIGUCHI
Department of Information and Computer Sciences,
Faculty of Engineering Science, Osaka University
Toyonaka-shi, 560 Japan

[1] 現時点では条件4.については実現できていない.

[2] 実際にはスレッドと呼ばれる軽いプロセス[4]を生成し, それに実行を任せる.

[3] CPUは一つであるので, 一度には一つのスレッド文しか走行状態ではない. 文脈切替えしながら走ることになる.

に太陽を置き, 太陽の自身の座標系を 5 秒で 360 度回転させることを永遠に続ける. これによって, 太陽の座標系に載っている地球と月が太陽の回りを回ることになる.

14–18 行目のスレッド文では, 地球を (太陽の上の) 座標 (100,0) に置き, 地球の上の座標系を 2.5 秒で 360 度回転させることを永遠に続ける. これによって, 地球の座標系の上に載っている月が地球の回りを回ることになる.

18–20 行目のスレッド文では, 月を (地球の上の) 座標 (40,0) に置く.

```
1  #include "animation.h"
2  #include "Bitmap.h"
3  @include
4  main()
5  {
6    MakeAnimationWorld( 400 , 400 , 0.05 ); // 初期化
7    @export BitmapObject Sun  ("bitmaps/sun.xbm");
8    @export BitmapObject Earth("bitmaps/earth.xbm", &Sun);
9    @export BitmapObject Moon ("bitmaps/moon.xbm", &Earth);
10   @{          // 太陽を制御するスレッド文
11     Sun.put( Point( 200, 200 ) );
12     while(1)
13       Sun.rotate( Angle( 360 ) , Sec(5) );
14   @} | @{     // 地球を制御するスレッド文
15     Earth.put( Point( 100 , 0 ) );
16     while(1)
17       Earth.rotate( Angle( 360 ) , Sec(2.5) );
18   @} | @{     // 月を制御するスレッド文
19     Moon.put( Point( 40 , 0 ) , );
20   @}
21 }
```

図 1: アニメーションの記述例

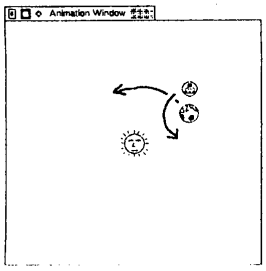

図 2: アニメーション実行例

3 言語処理系

PC++によるアニメーションの記述は図 3に示す過程を経て実行ファイルへと変換され, それを実行することによって, 4節で述べる機構が動作し, アニメーションを実行できる.

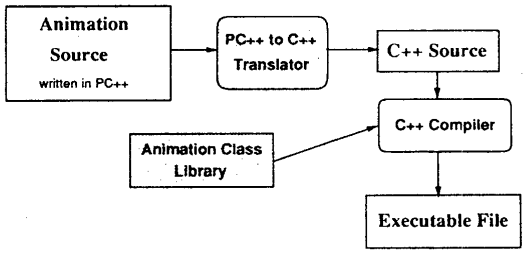

図 3: アニメーション記述から実行まで

4 アニメーションの実行

アニメーションは, 短い時間間隔で静止画像を逐次更新することによって実現する[4]. また, 本システムでは全ての並行動作はスレッドを用いて記述することにした.

1 つのオブジェクトが生成されると, AnimationObject クラスの構築子によって, そのオブジェクトの描画を担当するアニメーションスレッドと呼ばれるスレッドが 1 つ割り当てられる (図 4中の object1,2,3). オブジェクトへの動作の指示 (実際には C++のメソッドの呼び出し. n 秒間で座標 A から座標 B まで移動せよ等) は, その動作の開始時にアニメーションスレッドに伝えられる. スケジューラスレッドと呼ばれる特別なスレッドが, フレームの間隔毎に全てのアニメーションスレッドをスケジュールし, それぞれのオブジェクトの描画を要請する. スケジュールされたアニメーションスレッドは, 受け取った動作の指示に基づいて, 担当するオブジェクトの現在位置や表示イメージ等を計算し, 描画する.

スレッド文 (@{ ... @}) と並行動作記述子 ("&", "|") による並行動作は, PC++ to C++ Translator が, 並行動作記述子で連結されたスレッド文を並行に走らせるような C++のコードを出力することによって実現している. 図では 2 つのスレッド文が並行動作記述子で結ばれ, 並行に動作している (thread1,2).

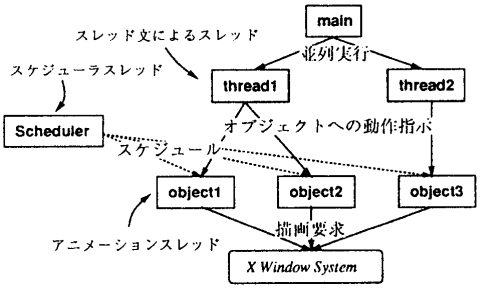

図 4: スレッド間の関係

5 あとがき

本研究では, 容易に並行動作を記述でき, アニメーションを記述するためのクラスライブラリを備えたアニメーション記述言語を作成した. オブジェクト指向言語である C++をベースとしたため, クラスの継承を用いてアニメーションオブジェクトの定義を容易に行なうことができ, また, 並行動作する複数のオブジェクトに対する指示を別々に記述できるため, 複数のオブジェクトが並行動作するようなアニメーションの記述が容易となった.

今後の課題としては, アニメーションの実行においてどの程度の大きさのイメージならば実用的な速度でアニメーションが可能なのかどうかを調べること等が挙げられる. さらに, 現時点では実現できていない, オブジェクト同士の関係 (2節の条件 4.) の記述を可能にすること等も挙げられる.

参考文献

[1] Sherwood, B.A. and Sherwood, J.N.: "CMU Tutor: An integrated programming environment for advanced-function workstations", *Proc. of the IBM Academic Information System University AEP Conference*, San Diego. (Apr. 1986).

[2] Shimojo, S., Fujikawa, T., Matsuura, T., Nishio, S., and Miyahara, H.: "A New Hyperobject System Harmony: Its Design and Implementation", *International Conference on Multimedia Information Systems*, McGraw-Hill, pp.243–257 (Jan. 1991).

[3] 松浦, 梶本, 谷口: "ワークステーション上での実時間アニメーションの方法とその評価", 信学論 (D-I), J75-D-I, 7, pp.469-478 (1992-07).

[4] "System Services Overview", Sun Microsystems, pp.71–106. (May 1988).

[4] この個々の静止画像をフレームと呼ぶ