動的な自然現象オブジェクトの対話型モデリング手法 ― 打上げ花火開発支援システムを題材として ―

樁田 泰行† 西野 浩明† 凍田 和美‡ 宇津宮 孝一†

本論文では動的オブジェクトをモデリングするための手法として、アニメーショングラフを用いたモデリング手法について述べる。アニメーショングラフとは、動的なオブジェクトの動きや変化を階層化し視覚化したグラフのことで、そのグラフの構造によりオブジェクトのアニメーションを決定する。モデル作成者は、このアニメーショングラフに動きや変化を規定するパラメータや関数をもつアニメーションノードの編集や追加、削除などの操作をすることでモデリングを行う。本手法の実装事例として、仮想打上げ花火開発支援システムを取り上げ、そのモデリング作業の流れとともに、グラフやノードの効果について説明する。

Interactive Modeling Method for Dynamic Natural Objects

Yasuyuki Kabata† Hiroaki Nishino† Kazuyoshi Korida‡ Kouichi Utsumiya†

This paper describes a new modeling method for dynamic natural objects using animation graphs. This animation graph is a hierarchical tree, being made of graphical nodes with rules like physical lows. The animation graph has dynamic object animation data, and dynamic objects are displayed according to the data. Users can easily create a dynamic object model by adding some animation nodes to the animation graph or deleting other nodes from the graph. This proposed method is applied to a modeling system for fireworks design.

1. はじめに

近年の計算機技術の発達に伴って、仮想環境も著しい進化を遂げている。仮想環境は、テクスチャマッピングなどの処理を施したリアルなオブジェクトから成り、現実感を増している。しかしながら、仮想環境の多くは、このような静的なオブジェクトのみで構成され、動的なオブジェクトが環境に組み入れられる例はそれほど多くない。これは、もちろん計算機の能力が動的なオブジェクトをシミュレートするのに十分でないことも原因の一つとして考えられるが、同時に動的なオブジェクトを作り上げるツールの不足や作成ツールの利用しにくさなども原因として挙げられるだろう。そこで我々は、仮想環境に組込むための動的なオブジェクトを容易かつ直感的に作成できるモデリング手法を提案し、その手法を実際の例に適用して、有効性を検討する。今回、モデリングの対象となる動的なオブジェクトとは、雲や煙などの自然現象を指し、これらのオブジェクトは,パーティクルを利用した描画方法によって再生される。また、これらのオブジェクトは Dynamic Natural

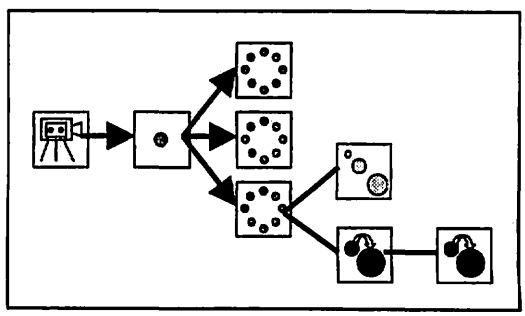

図 1: アニメーショングラフ

† 大分大学工学部
{ kabata, hn, utsumiya } @csis.oita-u.ac.jp
http://www.csis.oita-u.ac.jp/CS/cs1/cs1.html
‡ 大分県立芸術文化短期大学
korida@oita-pjc.ac.jp

Objects (以下 DNO)と呼ばれる。本手法では、これらのオブジェクトのアニメーションを図1に示すようなアニメーショングラフで動きを表現する。このグラフは、動きや変化のパターンを視覚化したノードで構成されている。 DNO のアニメーションは、このグラフを基に再生される。このDNOアニメーションを同時に見ながら、アニメーショングラフの編集を行うことで、直感的なモデリングを行うことができる。本手法の適用例として、我々は本手法を打上げ花火開発支援システムを取り上げる。仮想打上げ花火開発支援システムへ実装するために、グラフを構成するアニメーションノードを花火の動きや変化のパターンに沿って最適化し、モデリングの経験の乏しい初心者でも利用しやすい形で提供する。また、 本手法のモデリング概念を、この仮想打上げ花火開発支援システムを例に説明する。

2. 研究の背景

まず、DNOの描画方法を説明する。今回モデリングの対象となるDNOの一般的な描画法は、パーティクルシステム、もしくはテクスチャマッピングを利用した手法を用いる方法の2種類がある[1][2]。テクスチャマッピングを用いる方法は、一般的に家庭用ゲーム機等で用いられ、大雑把な形ではあるが、比較的容易にその表現(動きや変化)を手にすることが可能である。一方、パーティクルシステムは、粒子のシミュレーションなどに利用され、物理法則などをすべての粒子(点)に適応し、細かく精巧な動きをもつ煙の気流などのオブジェクトを作り出すことが可能である。しかしながら、このパーティクルを用いる手法では、物理法則を通して制御するため、利用者自身が頭に描く動きを作り出すことは難しい。我々は、仮想環境内にできる限り精巧なオブジェクトを配置することを考え、パーティクルシステムによるオブジェクトの描画法を利用する。したがって、この手法を用いてDNO のモデリングを行う場合は、いくつかの問題点を克服した上で実装しなければならない。問題点の第一は、DNO は、机やビルなどといったような静的なオブジェクトに比べてパラメータが多く、また時間毎に変化するパラメータの設定を行わなくてはならいため、パラメータの管理や把握が困難なことである。第二は、動きを決定する際に用いる物理法則などの関数を記述する必要があり、また、数式を利用し動きを間接的に制御するため、モデルの動きを制御するのは困難である。したがって、目的のアニメーションの作成までに膨大な時間を必要とする。第三の点は、パーティクルシステムのモデラの位置付けとして、応用レベルのアプリケーションであるため、3 次元 CG モデラの基本知識は当然のように必要となることである。したがって、本手法ではこのような問題点を解決するために、モデルの管理や編集を容易にするアニメーショングラフを用いて、グラフをモデリングインタフェースへ統合する。本研究の関連研究には、以下のものがある。Lintermann らは、植物を対話的にモデリングする手法を開発した[3]。 植物の形状は複雑であり、一般的にモデリングの難しいオブジェクトとされる。そこで、植物の生成過程を分類し、その生成過程をもつコンポーネントと呼ばれるものをモデリングの単位に利用する。コンポーネントは、生成過程を視覚化した画像である。利用者は、このコンポーネントを幾つかリンクさせることで、植物のモデリングを行う。船橋らは、編物のデザインを行うためのルールベースを用いた支援システムを開発した[4]。このシステムは、編物の初心者に自分の編上げたい編物の最終イメージを事前に見せることができる。利用者は、いくつかのルールをもった模様網目記号を碁盤目上に配置する。その碁盤目上に並んだ記号を基に編物の CG を作成する。このシステムは、単に最終画像を見せるだけでなく、計算機初心者でも、簡単に CG の作成を可能にするインタフェースの構築にある。以上の例は、今回提案する手法と同様に、オブジェクトの生成過程を分類し、パターンを用いてオブジェクトのモデリングを間接的に行う手法である。

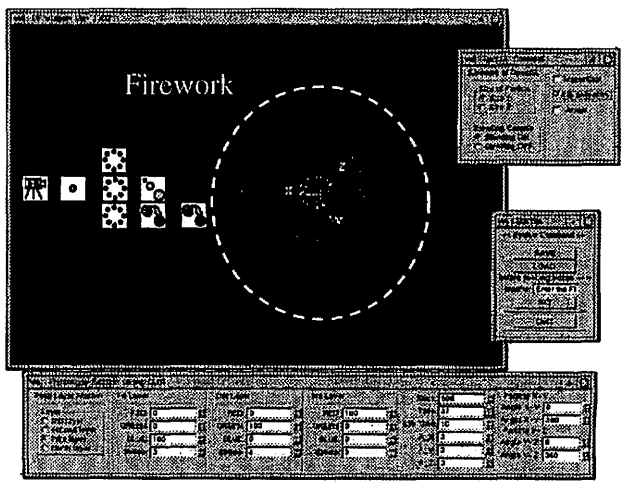

図 2:モデリングシステムのインタフェース

3. グラフを用いた対話型モデリング手法

パーティクルを用いたオブジェクトのモデリングは、いくつかの問題点をもっている。これらの問題点を解決するために、我々は、DNO のアニメーションをアニメーショングラフと呼ぶ階層化されたグラフ構造で表現する。同時にモデリングにもこのグラフを利用し、構成するノードの編集や追加を行うことで、DNO の間接的なモデリングを可能とする。 このノードの編集は、マウスやキーボードといった標準の入力装置で行う。アニメーショングラフの編集インタフェース画面の例を図2に示す。図 2 に示すように、グラフは DNO アニメーションの出力ビューワ上に統合され表示される。利用者は、このモデルのアニメーションとグラフを同時に見ながら編集を行う。また図2の周りのサブインタフェースは、ノードのパラメータ等の編集に利用する。以下の節では、まずアニメーショングラフとアニメーションノードの役割について述べる。

3.1 アニメーションノード

まず、アニメーショングラフを構成するノードについて説明する。 このアニメーションノードは、DNO の動きや変化をノード単位で保持する。この動きや変化を視覚化し、利用者に、そのノードの役割を視覚的に提示する。それぞれのノードにその役割を実行する関数をクラスに実装した。今回、このノードを打上げ花火開発支援システムに実装するために、アニメーションノードをシステムに沿って最適化したものを用意した。図 3 に示す正方形のイメージがそれぞれの役割を視覚化したノードである。大きく分類すると、生成ノードと制御ノードという2種類のノードがある。図3の(a)～(c)に示す生成ノードを用いることにより、グラフの進行時間がこのノードに到達すると、指定した数のパーティクルを生成する。今回、このパーティクルの生成法則を 3 種類作成した。(a)は一点、(b)は複数、そして(c)は放射上にパーティクルを生成する。生成されたオブジェクトに動きや変化をつけるノードが(d)～(k)に示す制御ノードである。制御ノードはさらに、属性制御ノードと軌道制御ノードに大きく分類される。ここでいう属性とは、色や形を言う。これらのノードは、生成ノードにリンクされることで、パーティクルに動きをつけることができる。図 3 の(d)は色の変更、(e)はパーティクルサイズの変更、(f)は残像効果、(g)はパーティクルを点滅させるためのノードである。軌道制御ノード(h)は直進、(I)波、(j)は螺旋、(k)はランダムといったような軌道をもつベクトルをパーティクルに与えるものである。このように、動きをオブジェクト化し、それをつなぎ合わせることで、間接的かつ直感的なアニメーションのモデリングを可能とする。

3.2 アニメーショングラフの役割とクラス構造

動作の単位であるアニメーションノードをリンクした階層的木構造をアニメーショングラフと呼ぶ。このアニメーショングラフを一見すれば、アニメーションの概要を把握できる。このグラフは、オブジェクトのアニメーション作成が大規模化するに従って重要になるであろう。また直感的にアニメーションの構成が理解可能となり、モデリングも容易となる。さらに、アニメーショングラフは、オブジェクトグラフで管理し保存すれば、以降のモデリングにおいて、再利用も容易に行える利点もある。このグラフを実現するためのクラス構造やデータ構造

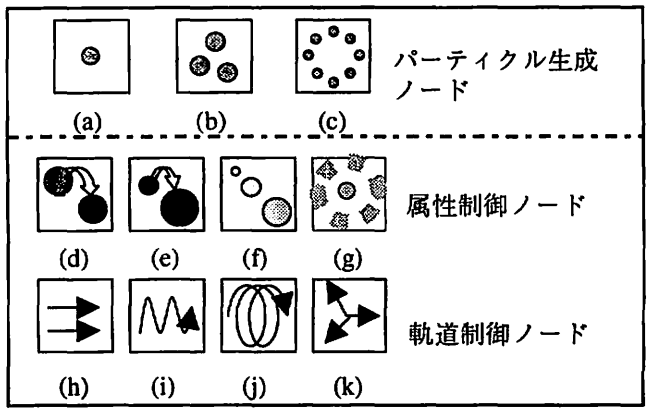

図 3：アニメーションノードの例

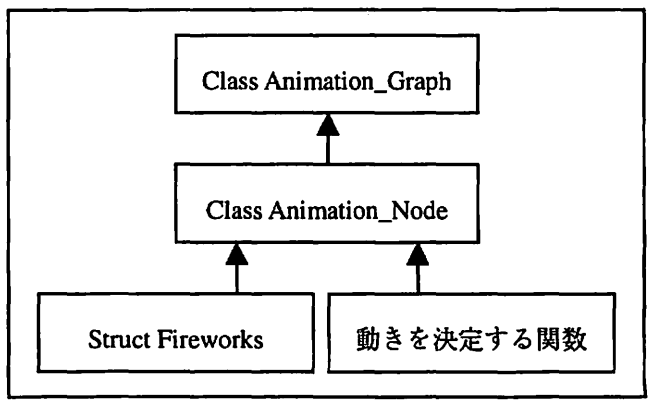

図 4:クラスの関係

について説明する。このグラフ構造を作成するに当たって、我々は、図4に示すような継承関係のクラスを作成した。クラスには、アニメーションノードクラスとアニメーショングラフクラスを用意した。一つの動的なオブジェクトのアニメーションは、一つのグラフクラスのインスタンスと複数のノードクラスのインスタンスから構成される。ノードクラスは、対象となるオブジェクトのデータもしくは動きや変化を決定する関数をもつ。今回、ノードとオブジェクトのデータ構造と関数とを切り離すことで、アプリケーションに応じたノードを作成することが可能となる。したがって、アプリケーションに応じてさまざまなグラフが作成可能である。また、この関数やデータ構造の作成はプログラマとアプリケーションのクライアントが協同で行う。今回は、打上げ花火開発システムに適用するため、花火のデータ構造とこれに必要な関数を用意した。

3.3 アニメーションカメラ制御ノード

本手法では、モデルを構成するアニメーションノードのほかにアニメーションの制御を行うためのアニメーション制御ノードを準備する。このノードを利用することで、アニメーションの設定と効率のよいモデリング環境を提供できる。図5に示すノードがアニメーション制御ノードで、図5(a)をモーションカメラ、(b)をスティルカメラと呼ぶ。これらのノードは、モデルのアニメーションを制御するために、2つの重要な役割をもっている。第一は、仮想環境におけるアニメーションの開始時間の設定で、仮想環境にオブジェクトを導入する場合に、再生時間の設定を行うものである。第二は、モデリング中のアニメーションの開始点や停止時間を視覚化することである。アニメーション開始位置の可視化の例を図6に示す。図 6(a)では、アニメーション開始記号が3連結のノードの先頭にあり、DNOはノードの最後までなぞり、終わると同時に先頭に戻って繰り返して再生される。また、(b)では、利用者が真中のノードを明示的にノードの最後と指定することで、最後のノードをなぞることなく、2つのノードのみを再生する。また、図 6(c)では、アニメーション制御ノードを移動することで、アニメーション開始位置を 2 番目のノードから開始している。このように、アニメーションの再生範囲をノード単位で決定することにより、ローカルな再生を可能とし、モデリングの効率を上げることができるので、大規模なグラフを構築する場合に特に有効である。以上のようなアニメーションの開始点の移動は、モデリングの作業に欠かすことのできない機能である。

4. モデル作成の枠組み

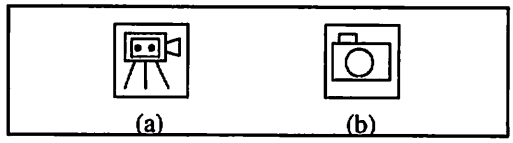

図5: アニメーション制御ノード

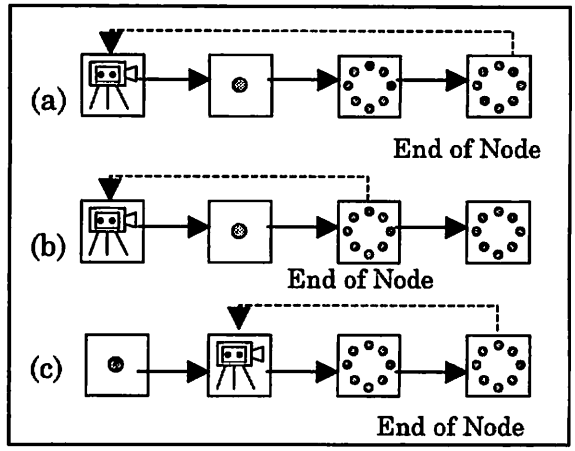

図6: カメラノードの利用例

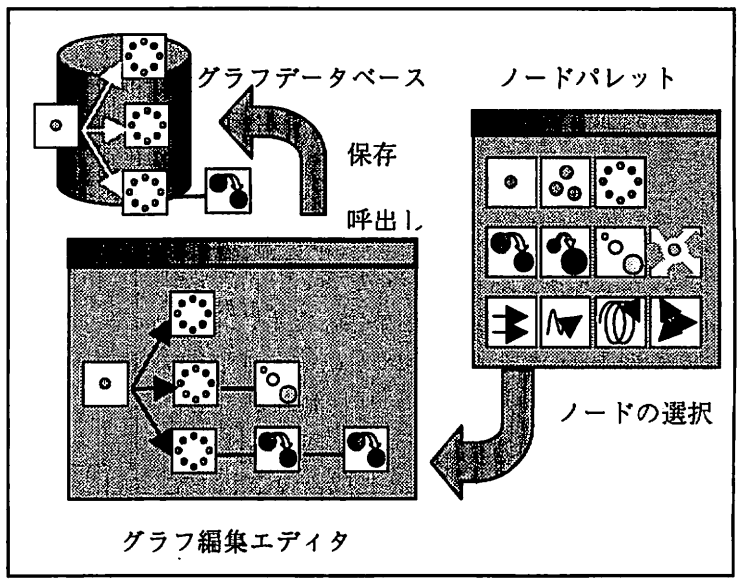

図7: グラフ編集の流れ

図7に、モデルの作成環境を示す。図で示すように、利用者がDNOを作成する場合、大きく 2 つの方法でアニメーショングラフを作成することができる。第一は、新規作成で自分のイメージに合うまでノードの追加や削除、編集を行う方法である。この方法は、ある程度このモデラに慣れた利用者向きかもしれない。第二の方法は、すでにあるアニメーショングラフを利用することによるアニメーションノードの編集を中心としたモデリング方法である。 利用者はDNOアニメーションのデータベースから、自分が制作したいオブジェクトのイメージに近いアニメーショングラフを呼出し、そのグラフを編集することでモデリングを行う。システムに慣れていない利用者にとって、前者の方法よりもモデリングが容易であろう。またグラフ構造の利点として、制作したアニメーショングラフ、もしくはいくつかのノードを部品としてデータベースに保存しておき、他のアニメーショングラフを作成する場合に再利用できることである。具体的なモデリング作業については、仮想打上げ花火開発支援システムを題材にして、次節で述べる。

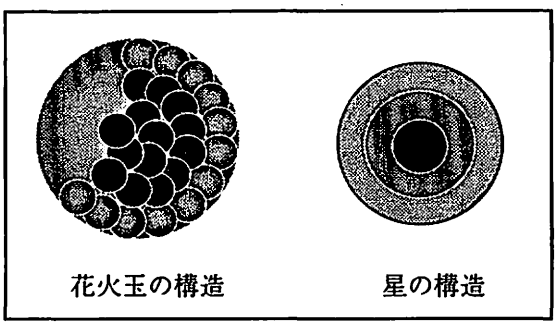

図 8 : 花火の構造

5. 打上げ花火開発支援システムへの実装

上述した手法を我々は、仮想打上げ花火開発支援システムに実装する。システムの利用者として、コンピュータやモデラ、CAD に慣れていない利用者を対象としている。打上げ花火もまた、雲や炎といったオブジェクトと同様に動的な自然現象オブジェクトに分類され、その描画方法にパーティクルシステムが用いられる。打上げ花火は見た目以上に複雑な動きと変化をもっており、点の設定だけでは、さまざまな種類を簡単に作成することができない。そこで、動きをパターン化し、専用のノードを作成し、それをモデリングに利用する。

5.1 打上げ花火システムへの適用

今回、打上げ花火にこの手法を実装したのは、花火が動的な自然現象オブジェクトであるという理由のほかに、使用する利用者(主に花火師)がコンピュータの初心者であるということ、また打上げ花火の構造が非常に階層的な作りになっていることが挙げられる。第 1 の理由であるが、パーティクルを利用したモデリングシステムは初心者にとって非常に利用しにくいものである。前述したように動的なオブジェクトは他の静的なオブジェクトに比べ、描画するためには非常に多くのパラメータを必要とする。したがって、現実の世界に近い形でモデリングの環境を準備する必要がある。図8に示すように、花火は階層的な構造をもつので、グラフ

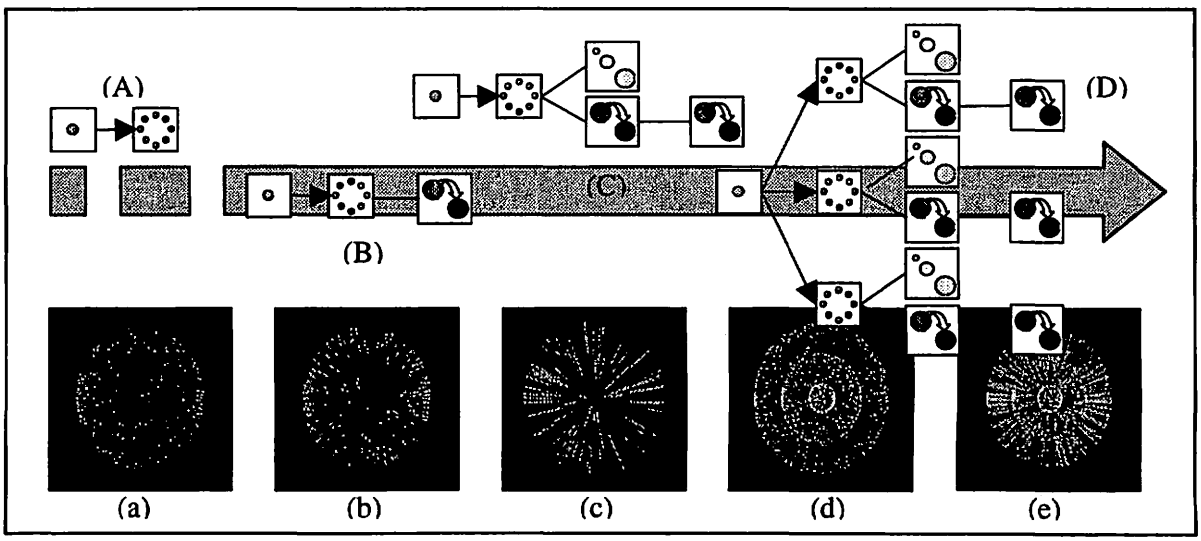

図 9:アニメーショングラフの作成過程

構造で表現しやすいという付随的な理由もある。

5.2 打上げ花火を例にしたモデリング過程

このモデリング手法の実装例として、図 9 に打ち上げ花火の製作過程を示す。図 9 では、利用者が新規作成から、3 層の打上げ花火のモデルを作成する流れを示したものである。まず、利用者はベースとなる生成を用意する。その図が(A)で、その状態での出力が(a)である。次にベースとなるグラフに動きや変化をつけるために、ツールパレットから、色を変化させるノードや残像を残すノードをリンクさせる。図(B)と(C)がそれに当たる。これによって生成される画像が(b)や(c)となる。このような変形方法は、実世界にあるルールを視覚化し、間接的にパーティクルのモデリングを行っている。さらに花火を階層化し、3 層構造を作るために、同じグラフを部品として再利用することで、簡単に3層構造の花火を作成することができる。

5.3 実装環境

図10に本手法を実装した仮想打上げ花火開発支援システムの環境を示す。このシステムを、Silicon Graphics 社製 WS の O2 (IRIX6.3) と ONYX2 (IRIX6.4) そして、Visual Workstation 320 (Windows NT) 上に実装する。このアプリケーションに、プログラミング言語としてCとC++を利用し、グラフィックスライブラリに OpenGL[5]と OpenGL glut[6]、インタフェースライブラリに GLUI[7]を利用した。このシステムでは、各計算機上で作成したデータを Onyx2 上のデータベースに転送し、大型スクリーンによる仮想環境での打上げ花火を模擬できる。

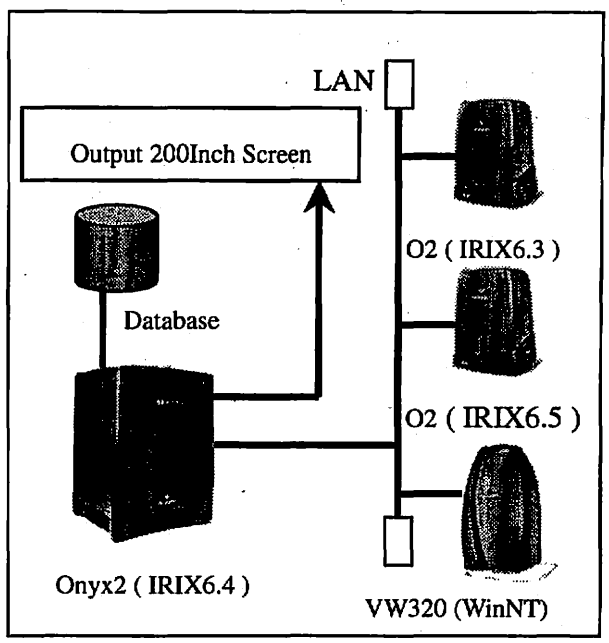

図 10 : 打上げ花火システム実装環境

6. おわりに

本論文では、動的なオブジェクトのモデリングを目的としたオブジェクトのグラフによる管理方法とアニメーショングラフを用いたモデリング手法について述べ、この提案手法を打上げ花火開発支援システムに適用して、その有効性を検討した。本手法を用いることにより、複雑な動きをもつオブジェクトをノードの組合わせアニメーショングラフを作成することで容易にモデリングを行うことが可能となった。

今後の課題として、本手法は仮想打上げ花火を題材にしたノードを作成してきたが、他のオブジェクトの場合について、一般的なノードの作成方法を検討する必要がある。また花火では、ノード間の干渉などを気にする必要はなかったが、衝突などの処理についてもノードレベルで実行できないか検討していきたい。

参考文献

[1] James.D.Foley, et al: Computer Graphics Principls and Practice, Addison-Wesley, 1998
[2] 安居院, 中嶋: コンピュータ グラフィックス, 昭晃堂, 1992.
[3] Lintermann.B, Deussen.O : Interactive Modeling of Plants, IEEE Computer Graphics and Application, Vol.19, No.1, pp. 56 – 65, 1999
[4] 船橋, 岩月, 武藤, 山田, 伊藤：ルールベースを用いた編物デザイン支援システム, 情報処理学会論文誌, Vol.39, No.8, pp.2547-2552, 1998
[5] Neider,W. David,T. and Woo,M: OpenGL Programming Guide, Addison-Wesley, 1993
[6] Mark.J.Kilgard: OpenGL Programming for X Window System, Addison-Wesley, 1997.
[7] Rademacher,P: A GLUT-Based User Interface Library Ver2.0, 1999. http://www.cs.unc.edu/~rademach/