ヒューマンインタフェースシンポジウム2000論文集, pp. 407-410, 2000

オブジェクト指向に基づく仮想空間構築手法に関する研究

小牧 大輔[*1]　社領 一将[*1]　石井 裕剛[*1]　吉川 榮和[*1]

A Study on Object-oriented Construction Method for Virtual Environment

Daisuke Komaki[*1]　Kazumasa Sharyo[*1]　Hirotake Ishii[*1] and Hidekazu Yoshikawa[*1]

Abstract – **The goal of this study is to develop a new framework for constructing virtual environments, in which the interaction not only between a virtual human and virtual objects but also among plural virtual objects can be simulated, and the virtual objects can be reused easily without reconstructing them. To achieve this goal, a new method for constructing virtual environments has been proposed by applying the object-oriented concepts to the definition of virtual objects. This method allows convenient construction of virtual objects by combining the components of various functions which are independently developed in advance. In this paper, how to apply the object-oriented concepts to the definition of virtual objects and an example of the construction of virtual objects in accordance with the proposed method are described.**

Keywords : **Virtual Human, Virtual Environment, Object Interaction, Object-oriented.**

1. はじめに

仮想現実感 (Virtual Reality:VR) の技術が発展するに伴って、人と物、および物と物が相互に作用 (インタラクション) できる 3 次元仮想環境の必要性が、様々な分野で高まりつつある [1]。また、Johnson らは 3 次元コンピュータグラフィックス (CG) を用いたアニメーションで、自然な動作が可能な仮想人間を構築し、複雑な機器の操作や修理などの教育に、仮想人間が実際にお手本を見せながら教える方法が、効果的であることを確認している [2]。

しかし、従来の仮想空間の構築方法 [3] では、構築対象が複雑で大規模になったときに、仮想物体の運動や仮想物体間の関係、および仮想人間の動作を効率的に扱うことが困難であり、構築労力が膨大になる問題が存在する。

同様に、従来の大規模コンピュータソフトウェアは、構築に時間と労力が必要であり、開発には困難を伴っていた。しかし、C++や JAVA などのオブジェクト指向型プログラミング言語の普及と共に、既存のソフトウェアコンポーネントを組み合わせる方法の開発が発達し、現在ではソフトウェアを構築する労力を大幅に削減し、さらに大規模なソフトウェアシステムの開発も容易に行えるようになっている。

本研究では、仮想人間の動作と、その動作に応じた仮想物体の動きを同時に合成でき、仮想物体同士のインタラクションも考慮した仮想空間を効率的に構築するための新しい手段として、仮想物体や仮想人間の性質・機能等を定義する手法にオブジェクト指向の考え方を導入する。

本報告では、まず仮想物体の定義へオブジェクト指向を適用する方法について述べ、次に仮想物体の機能・性質をコンポーネント化したクラスを管理する方法について述べる。その後に、それらの手法に基づいて構築した仮想物体同士のインタラクションも考慮した仮想空間のシミュレーション手法について述べ、最後に、仮想人間の動作を合成する手法について言及する。

2. オブジェクト指向に基づいた仮想物体の定義方法

ここでは、仮想物体の機能・性質の定義手法へ、オブジェクト指向の概念を適用する方法について述べる。

2.1 仮想物体定義へのオブジェクト指向の適用

オブジェクト指向とは、近年ソフトウェアプログラミングの分野などで頻繁に使用されている考え方である。オブジェクト指向では、共通の属性をコンポーネント化することにより、共通の処理を行うプログラムなどの再利用を容易にしている。

本研究では、仮想物体の機能・性質の定義手法にオブジェクト指向の概念を導入することで、仮想物体の形状や動き、その仮想物体に対する仮想人間の動作を合成するための情報など、仮想物体の機能・性質をコンポーネント化して、様々な仮想物体に再利用できるようにすることを目的とする。その際、オブジェクト指向の性質として、クラス化、継承、多態性、カプセル化を利用する。以下では、オブジェクト指向の各性質の役割と、仮想物体の機能・性質の定義手法への適用方法について述べる。

2.1.1 クラス化の適用

クラス化とは、性質などの共通の属性をコンポーネント化することであり、一般的には再利用性の向上、変更への耐性の向上などのメリットがある。本研究では、仮

*1: 京都大学大学院 エネルギー科学研究科
*1: Graduate School of Energy Science, Kyoto University

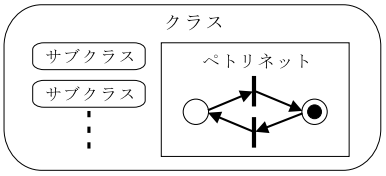

図 1　クラスの構成
Fig. 1　Configuration of Class.

想物体の形状や動き方、その仮想物体に対する仮想人間の動作を合成するための情報などの特徴をクラス化する。また、仮想物体の動きをシミュレートするときに、仮想物体の機能・性質を構成するクラスのコピー (インスタンス) がコンピュータ上に作成され、その機能・性質を実現する。インスタンスは複数の内部状態を持ち、その内部状態は、外部からのイベントの発生により遷移する。そして、その内部状態に応じて、インスタンスが提供する機能・性質が変化するものとする。

本研究で提案する仮想物体の定義手法では、クラスはその構成要素として次の 2 つを持つことができる。

- 1 つのペトリネット
- 複数のメンバクラス

ペトリネットは、その属するクラスの内部状態とイベント発生による内部状態の遷移を定義するためのものであり、詳細は 3.章で述べる。メンバクラスは、クラスに機能・性質を付加するものであり、複数個定義することが可能である。

本研究で用いるクラスの例としては、形状を定め衝突するという機能を持つ「平面クラス」や「円筒クラス」、動くという機能を持つ「動作クラス」などがある。

2.1.2　継承の適用

仮想物体の形状を平面と捉えて衝突を計算する「平面クラス」と、円筒形と捉えて衝突を計算する「円筒クラス」は、衝突を計算する具体的な処理方法は異なるが、同じ衝突する機能を扱うという共通性を持っている。そこで、図 2に示すように、この共通な機能を持った「形状クラス」という、より汎用的なクラスでグループ化することができる。

継承とは、このような共通な機能・性質でグループ化された汎用的なクラスから、より専門的なクラスを生成する際に用いられる機能で、継承元のクラスをスーパークラス、継承先のクラスをサブクラスと呼ぶことにする。一般に、この継承という性質により、同様の機能・性質を持ったクラスを定義する際の手間の軽減や、機能・性質の扱いを統一することによる保守性の向上、さらに次に述べる多態性の実現とそれによるメリットがある。

なお、継承には、多重継承と単一継承が存在する。本研究では、多重継承を利用した時に継承関係などが複雑になる現象を回避するために、単一継承のみを認める。

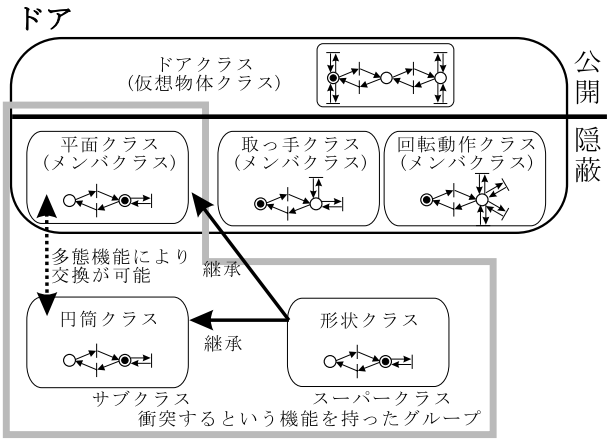

図 2　仮想物体におけるクラスの役割
Fig. 2　Roles of Class in Virtual Object.

2.1.3　多態性の適用

多態性とは、実際のクラスの性質・機能等を意識せずにクラスの処理を呼び出すための機能である。例えば、「形状クラス」を継承している「平面クラス」や「円筒クラス」は、「形状クラス」から衝突するという処理を継承しているとする。この時、実際には「平面クラス」と「円筒クラス」のときでは異なる計算処理で衝突を判定するが、その違いを意識せずに同じ衝突として処理することが可能となる。

2.1.4　カプセル化の適用

カプセル化とは、クラスの本質的な特徴に寄与しないクラスの詳細を隠蔽することであり、隠蔽することによって、内部データに対する外部からの直接のアクセスを禁止し、データ構造の一貫性を保証し、保守性を向上させることができる。本研究では、2.1.1項で述べたように、クラスの本質的な特徴である機能・性質は、クラスの内部状態とイベント発生によって定義されている。よって、他のクラスからは (1) 内部状態の把握、(2) イベント発生の感知、(3) イベント発生の提示、という 3 つの処理のみ利用することを可能とし、それ以外の情報をすべて隠蔽する (図 2参照)。

2.2　クラスによる仮想物体の構成例

以上で述べたオブジェクト指向の各性質を、ドアの定義に適用した例を図 2に示す。この場合、ドアはドアの機能を統括するドアクラスという仮想物体クラスと、衝突や動きなど、様々な機能を実現する複数のメンバクラスにより、構成される。また、平面クラスが形状クラスから継承されているように、それぞれのクラスはより汎用的なクラスより継承されて定義される。クラスの機能・性質は、内部状態とイベント発生により定義されるが、これは 1 つのペトリネットで表現する。ペトリネットによるクラスの機能定義手法と、クラスから作られるインスタンスの機能管理手法は 3.章で述べる。

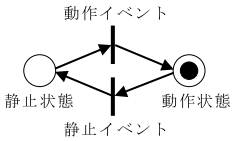

図 3　動作インスタンスのペトリネット
Fig. 3　The Petri-net for Motion Instance.

2.3　オブジェクト指向を導入する利点

以上のように、オブジェクト指向の各性質を利用して、仮想物体を定義すると、次のことが可能となる。

- 形状などの仮想物体の機能・性質を 1 つのクラスとしてコンポーネント化できる。
- 各機能・性質を実現するクラスを、より汎用的なクラスから継承することで、機能をグループ化することができる。
- 多態性の機能を利用することで、同じクラスから継承して作成した、クラスを相互に自由に入れ替えることができる。

これらが可能になれば、仮想物体を作成する際に、仮想物体の雛型となる仮想物体クラスに必要な機能を持ったメンバクラスを付け加えるという手法で、仮想物体を作成することが可能となる。この開発手法の具体例については、4. 3節で説明する。

3.　ペトリネットを利用したクラスの状態の管理

2. 1. 1項で述べたように、クラスの機能・性質を、内部状態とイベント発生の組み合わせにより定義する。本研究では、この内部状態とイベント発生による内部状態の遷移をペトリネット [4] を用いて表現する。具体的には、クラスの内部状態をペトリネットのプレースに、クラスのイベントをペトリネットのトランジションにそれぞれ対応させる。内部状態の遷移はプレース間を移動するトークンにより表現される。

例えば、「動作クラス」は、仮想物体が静止している状態を実現する「静止状態」と、動いている状態を実現する「動作状態」を内部状態に持つ。さらに、「動作イベント」と「静止イベント」という 2 つのイベントを持ち、「動作イベント」の発生により「静止状態」から「動作状態」へ、「静止イベント」の発生により「動作状態」から「静止状態」へ遷移すると定義できる。「動作クラス」から作られる「動作インスタンス」は、これに対応する図 3に示すペトリネットをシミュレーションすることにより管理される。

4.　物の動きの合成とインタラクション

ここでは、仮想物体の定義方法を具体的な例を用いて説明する。

4.1　仮想物体の構成と処理の流れ

図 2に示したように、1 つの仮想物体は、1 つの仮想物体クラスで構成され、その仮想物体クラスが、複数のメンバクラスを持つ。

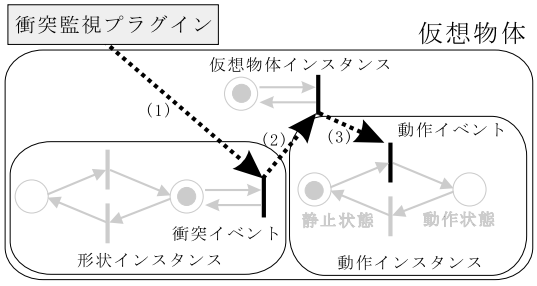

図 4　衝突時の処理の流れ
Fig. 4　Process Flow of Collision Event.

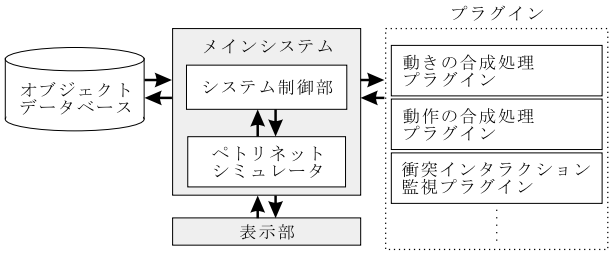

図 5　システム構成の概略
Fig. 5　Overview of System Configuration.

例えば、「他の仮想物体との衝突によって動きだす」という性質の仮想物体を定義したいとする。この場合、「仮想物体クラス」に、衝突機能を実現する「形状クラス」と、動く機能を実現する「動作クラス」(3.章参照) という 2 つのメンバクラスを加える。「形状クラス」は、「静止状態」と「衝突イベント」を持っており、4. 2節で述べる衝突インタラクション監視プラグインによって、衝突イベントが発生することで、「仮想物体クラス」に衝突したことを知らせる機能を持つ。

ここで、「形状クラス」が衝突イベントが発生したことを認識した際、「動作クラス」の動作イベントを発生させるという処理を、「仮想物体クラス」に定義する。そうすると、衝突が発生したときには図 4の (1)(2)(3) の順序で処理が行われ、「動作クラス」から作られる「動作インスタンス」は静止状態から動作状態に遷移し、仮想物体が動き出すという現象を実現できる。

4.2　インタラクションの監視

仮想物体と仮想物体の間には、衝突などのインタラクションが働く場合がある。本研究では、このようなインタラクションは、システムプログラムにプラグインとして実装される専用プロセス (図 5参照) が、仮想空間内のインスタンスの位置や状態を常に監視して、必要なときにインスタンスのイベントを発生させることで実現する。

例として、仮想物体間の衝突というインタラクションを実現する場合について述べる。この場合、メンバクラスとして実装されている「形状クラス」から作られる「形状インスタンス」と、「形状クラス」を継承している

すべてのクラスから作られるインスタンスの位置を、衝突インタラクション監視プラグインが監視し、衝突する状態になったと判断できる時点で、図 4の (1) に示すように、衝突する双方のインスタンスの衝突イベントを発生させることになる。衝突に伴って起こる現象は、衝突イベントが発生した時に実行される処理によって規定される。

4.3 仮想物体の定義例

ここでは、ドアや窓など、人が操作して開閉する物を対象に、仮想物体の定義例を述べる。人は、ドアと窓の両方に対して「開ける」と「閉める」という動作をする。本研究では、5.章で述べるように、仮想人間の動作を合成するために必要な情報 (動作情報) は、仮想物体から受け取る。そのため、ドアと窓の仮想物体クラスである、ドアクラスと窓クラスは、「開ける」と「閉める」という動作情報を人に提供するという、同じ機能を持っているとしてグループ化でき、そのスーパークラスとして「開閉クラス」を定義できる。

ここで、「開閉クラス」のメンバクラスとして、「持つ」という動作に必要な情報を仮想人間に提供する機能を有する「取っ手クラス」と、衝突などの形状からくる機能を有する「形状クラス」、および仮想物体の動く機能を有する「動作クラス」を定義すると、取っ手を持ってドアを開けたり閉めたりする機能が実現できる。

この「開閉クラス」から継承して「窓クラス」を作成するには、「形状クラス」を継承した「平面クラス」と、「動作クラス」を継承した「平行動作クラス」を、「形状クラス」と「動作クラス」の代わりに入れるだけでよい。また、「ドアクラス」を作成するには、「平行動作クラス」を「回転動作クラス」に入れ替えるだけでよい。

このように、仮想物体の機能の定義にオブジェクト指向の考え方を導入することで、仮想物体の各機能をコンポーネント化することでき、各機能のコンポーネントを組み合わせることで、仮想空間内の仮想物体を定義することが可能となる。今回用いた各機能のクラスは、他の仮想物体の定義にも用いることができるので、再利用性が高く、効率的な仮想空間の構築を支援できるものと考えられる。

5. 人の動作の合成手法

ここでは、仮想物体に対する仮想人間の動作の合成方法について述べる。

5.1 動作の合成手法

オブジェクト指向の考え方に従えば、仮想人間の機能である動作は、仮想人間にクラスとして追加できるようにする方法が、一番適切である。しかし、この方法を用いると、新たに定義した物に対して適切に動作をするためには、仮想人間に何らかの動作を合成するために必要な情報 (動作情報) を追加することが不可欠になり、定義が煩雑になる。

そこで、本研究では、仮想物体に仮想人間の動作情報をもたせる手法 [5] を採用する。具体的には、仮想物体を構成する仮想物体クラスか、そのサブクラスに動作情報をもたせる。例えば、「取っ手クラス」には「持つ」という動作情報を持たせておく。そうすることで、新たに「取っ手クラス」を追加した時に、仮想人間は新たに定義し直すことなく、「持つ」という動作することが可能になる。

5.2 仮想人間の動作合成処理例

「ドアを開ける」という動作を例に、仮想人間の動作をどのように実現するかを説明する。本研究で作成するシステムでは、ユーザが「ドア」「開く」というように、動作対象の仮想物体名と動作名を指令して、仮想人間に動作を命令する。そこで、仮想人間が「ドアクラス」(4. 3節を参照) から作られる「ドアインスタンス」に対して、「開く」という動作を合成するために必要な情報を要求する。それに対して、ドアインスタンスは、ノブを握る動作に必要な、ノブの位置やノブの太さ、ドアを開く動作に必要な、手の通るべき軌跡等、具体的に動作を合成するために必要な情報を返す。仮想人間は、その情報を利用し、プラグインの形でシステムに実装されている動作合成プログラムを用いて動作を合成する。

6. ま と め

本研究では、仮想人間の動作と、その動作に応じた仮想物体の動きを同時に合成でき、仮想物体同士のインタラクションも考慮した仮想空間を効率的に構築するための新しい手段として、仮想物体や仮想人間の性質・機能等を定義する手法にオブジェクト指向の考え方を導入した。

今後の課題としては、この仮想物体定義手法を利用した、仮想空間シミュレーションシステムの開発や、GUI による仮想物体定義支援システム、仮想空間構築支援システムの構築が挙げられる。

参 考 文 献

[1] N.I.Badler: Virtual Humans for Animation, Ergonomics, and Simulation, IEEE Workshop on Non-Rigid and Articulated Motion,(1997).

[2] W.L.Johnson, and J.Rickel: Steve: An Animated Pedagogical Agent for Procedural Training in Virtual Environments, ACM Press,Vol.8,No.1-4,pp.16-21,(1997).

[3] 石井他: 人工現実感技術を用いた機器補修の訓練環境構築支援システムの開発, 日本バーチャルリアリティ学会論文誌,Vol.4,No.1,pp.303-312,(1999).

[4] 村田: ペトリネットの解析と応用, 近代科学社,(1992).

[5] 市口他: 仮想空間におけるアフォーダンスを利用した人体モーションの合成, ヒューマンインタフェースシンポジウム'99 論文集,pp.211-216,(1999).

[6] 社領他: 仮想空間内における人の動作の融合手法に関する基礎研究, ヒューマンインタフェース学会研究報告集,Vol.2,No.2,pp.63-68,(2000).