EFFECTIVE CONSTRUCTION OF
PHYSICALLY BASED VIRTUAL ENVIRONMENTS

BY

ZHANG XIAOKE

A Thesis Presented to
The Hong Kong University of Science and Technology
in Partial Fulfillment
of the Requirements for
the Degree of Master of Philosophy
in Industrial Engineering and Engineering Management

Hong Kong, December 1998

Copyright © by Zhang Xiaoke 1998
AUTHORISATION

I hereby declare that I am the author of this thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for academic purposes.

[Signature]

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of research and education.

[Signature]
EFFECTIVE CONSTRUCTION OF
PHYSICALLY BASED VIRTUAL ENVIRONMENTS

BY

ZHANG XIAOKE

Approved:

[Signature]

DR. CHUAN-JUN SU  SUPERVISOR
Department of Industrial Engineering and Engineering Management

[Signature]

PROF. MITCHELL TSENG  HEAD
Department of Industrial Engineering and Engineering Management

[Signature]

DR. VINCENT DUFFY  MEMBER
Department of Industrial Engineering and Engineering Management

[Signature]

DR. BENJAMIN YEN  MEMBER
Department of Industrial Engineering and Engineering Management

Department of Industrial Engineering and Engineering Management
December 1998
ACKNOWLEDGEMENTS

I wish to express my grateful appreciation to my supervisor Dr. Chuan-Jun Su, for his genius supervision. Prof. Mitchell Tseng's valuable feedback is highly appreciated. Without their support I would not have finished this thesis at all.

I also wish to thank Ford Motor Company for the constant support during the project period.

I enjoyed the cooperation with Mr. Chua Fung Chun, Mr. Kan Ho Yin and Mr. Pong Yip Lam very much, and want to express my thanks for their programming work. Special thanks are extended to Mr. Denil Chan for his technical aids.

Ms. Breuer Vera spent a lot of time correcting my thesis, her help and kindness is unforgettable.
# Table of Contents

Title Page................................................................................................................................. 1

Authorisation Page.................................................................................................................... ii

Signature Page........................................................................................................................... iii

Acknowledgments...................................................................................................................... iv

Table of Contents..................................................................................................................... v

List of Figures ........................................................................................................................... vii

1. Introduction .......................................................................................................................... 1
   1.1 Problem Formulation ........................................................................................................ 1
   1.2 Thesis Objective ............................................................................................................... 3
   1.3 Motivation ........................................................................................................................ 3
   1.4 Key Issues of Physically Based Virtual Environment Development ......................... 4
   1.5 Approaches to the Objective .......................................................................................... 5
   1.6 Organization of this Thesis .............................................................................................. 6

2. Literature Review .................................................................................................................... 7
   2.1 Effective construction of Virtual Environments ............................................................ 7
   2.2 Physically Based Modeling ............................................................................................. 8
   2.3 Sense of Reality ............................................................................................................... 13
   2.4 Interaction ....................................................................................................................... 16
   2.5 Summary .......................................................................................................................... 19

3. Framework for Effective Construction of Physically Based Virtual Environments ... 21
   3.1 Introduction-simulation based design and VR technology ............................................. 21
   3.2 Strategies of the Framework ........................................................................................... 23
   3.3 Motivation of Integrating Process Description and Behavior Modeling ........................ 24
   3.4 Architecture of IVECS .................................................................................................... 31
   3.5 Benefits of Integrated Virtual Environments for Complex Systems ............................ 38
List of Figures

Figure 1-1. Human machine integration ......................................................... 2
Figure 3-1 Mapping of process description and behavior modeling ................... 24
Figure 3-2 Example of a process flow diagram .............................................. 29
Figure 3-4 Immersive Process Detailing ....................................................... 36
Figure 3-5 Communication between different levels and entities ...................... 38
Figure 4-1 Structure of object behavior modeling .......................................... 53
Figure 4-2 Rigid object simulation control flow ............................................ 56
Figure 5-1 Virtual Environment for VPPG .................................................... 60
Figure 5-2 Paradigm of Ping-Pong behavior modeling .................................... 63
Figure 5-3 Integration of visual model and physical simulation ....................... 69
Figure 5-4 Snapshot of the physical simulation ............................................ 70
EFFECTIVE CONSTRUCTION OF
PHYSICALLY BASED VIRTUAL ENVIRONMENTS

BY

Zhang Xiaoke

A thesis presented to The Hong Kong University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Philosophy in Industrial Engineering and Engineering Management

Hong Kong

December 1998

ABSTRACT

Virtual Reality (VR) technology has potentially wide applications in areas where it is more economical and safer for participants to experience virtual worlds rather than corresponding real worlds. However, VR applications are very limited due to the high cost of Virtual Environment (VE) development and comparatively low quality of the VE system. This thesis emphasizes the efficient construction of VEs in order to reduce the cost and lead time of VEs development. In addition, the thesis focuses on improving the quality of VE by enhancing the sense of realism (correct behavior) via Physically Based Behavior Modeling. Dynamical behavior modeling is of critical importance to enhance the sense of realism in VEs. A hierarchical structure for dynamical behavior modeling is proposed
based on characteristics analysis of simulation entities and behavior levels in a VE. By comparing existing behavior modeling tools or methodologies, an Object-Oriented (O-O) approach is adopted. The flexibility, reusability problems of VE development are then partially solved. Physically based behavior modeling as an instance of dynamical behavior modeling is specially discussed.

It is necessary and valuable to integrate VR technology with complex System Engineering to make VR really useful. However, system complexity and immature VR technology limit the integration. Therefore, a framework for effective construction of VEs for complex processes or systems is needed. Integrated VE for Complex System (IVECS) is presented as a framework to solve the integration problem.

This framework handles from high-level process description to low-level behavior modeling. IDEF3 (Integrated DEFinition language) is used as process modeling tool and O-O dynamical behavior modeling is integrated in the IVECS. Behavior lib and property lib are used to reduce the VE construction time.

Though future work is still needed, it is hopeful that IVECS is able to reduce the high cost of VEs construction and development, and at the same time improves the quality of the VE by Physically Based Behavior Modeling.

In the end, as an implementation example, a Virtual Ping Pong Game is developed following the IVECS framework.

**Key Words:** Physically Based Modeling, Behavior modeling, Virtual Environment construction, System Engineering, Human Computer Interface, Object Oriented, Complex System
1. Introduction

1.1 Problem Formulation

Virtual world is a computer generated, three-dimensional, interactive, multi-sensory, synthetic, and immersive Virtual Environment (VE) for user's reach. Virtual Reality (VR) means the immersion of human in a computer generated VE. In essence it is an advanced interface to human machine interaction. VR technology has wide applications in areas where presence is essential and where it is more economical and safer for participants to experience virtual worlds rather than corresponding real worlds. Applications include engineering & architecture, education & training, military simulations, scientific visualization, financial analysis, and medical applications.

However, VR has not been widely accepted in industries. There are two main reasons: 1) VR technology is not mature enough. It is difficult and time consuming to develop a VE. Moreover, the technique itself is still under development and can not fulfill the requirement of "reality" and "ease of use"; 2) There is a psychological distance to consider VR as a tool rather than as a luxury toy. The use of expensive VR peripheral established a fiction-like image and mass media has reported virtual worlds as computer generated alternatives to reality. VE users tend to have a high expectation before they really get into the virtual world. The gap between the real world and computer generated world is still a great obstacle to VR's popularization. As a result, there is comparatively little literature on really successful VR applications in industry. To bridge the gap, the first and most important step is improving the visual representation. In this thesis, to have a realistic visual representation means to have a
physical simulation under computer graphics. It is difficult to make graphics behave physically correct, because there is no relation between computer graphics and physical properties. In order to make a VE applicable to engineering design and analysis, integration of visual models and physical simulation is essential. As described in Figure 1-1, a user interacts with machine through visual models (computer generated 3D models) in the VE while a physical simulation is running in the back end to increase the sense of realism. The importance of this integration has been well recognized by researchers in the field of Physically Based Modeling (PBM). However, PBM still remains as a research issue due to its complexity and technological limitations of hardware and software. We are lacking of a general approach for physically correct behavior modeling. This results in a big gap between VEs and physical environments.

Figure 1-1. Human machine integration

To VE developers, reducing the lead-time and increasing the reusability are also very important. Considering a VE as a special product, quality and cost are equally important. The realistic visual representation and user friendly interface are quality consideration, while effective VE construction is more cost related.
1.2 Thesis Objective

In order to effectively construct physically based VEs, our research objectives are: 1) Provide a process description based methodology to effectively generate VEs; 2) Enhance the sense of realism of a VE by using Physically Based Behavior Modeling. Therefore we can bridge the gap between a VE and a physical environment for engineering design and analysis. The ultimate goal is to develop a framework for effective construction of VEs for engineering design and analysis including physical experiments without the cost of physical setup and investment.

1.3 Motivation

The construction of physical environments for engineering design and analysis is time-consuming, labor-intensive, and sometimes dangerous. New approaches are needed to reduce experimental time, cost, and improve the degree of safety in order to raise the competitiveness of the company.

With the advent of VR technology, conducting experiments without setting up physical environments becomes possible. More and more people are realizing the advantages of performing virtual tests in a VE instead of investing precious time and money on physical tests. For example, software such as PAM-CRASH (PC based) is commercially available for analyzing automobile collision. The limitation of those techniques compared with VR technology is the incapability of providing 3D environment for immersed navigation, 3D visualization (although some software does provide 3D visualization without enough interaction) and the most important sensory interaction. If we can conduct physical experiments in three-dimensional, interactive and immersive VEs (it is not applicable up to date), the benefits are more than those
PC based software can provide. Though benefits of virtual experiments for engineering design and analysis are obvious, VR application is greatly limited by long lead-time and high cost for creating a VE. Two major technical limitations of VR keep it from being widely adopted in industrial applications: First, it is tedious and difficult to create or modify a VE for a complex system; Second, even with those fancy and expensive devices such as Dataglove and Head Mounted Display (HMD), the computer generated world is still far away from the real world. Reducing lead-time for creating a physically based VE (means enhancing the realism) that allows engineers to perform design and/or experiments is therefore the main research goal of this thesis.

Conducting physical experiments in a VE requires a tremendous amount of efforts. Multi-disciplinary knowledge is required such as material validation, structure analysis, air dynamic analysis as well as computer graphics modeling and VE construction. In addition, most advanced numerical techniques such as Finite Element (FE) method are needed to provide physically correct modeling capabilities. Contemporary hardware can not conduct a FE analysis of complex structure in real-time, while a car crash test needs to model details of dynamic behavior under different loading conditions. As our first attempt to investigate general approaches for effective construction of VEs to conduct physical experiments, a Virtual Ping-Pong Game (VPPG) has been developed.

1.4 Key Issues of Physically Based Virtual Environment Development

From system developer's perspective, to develop a physically based VE, the following issues must be considered:

- Formal representation of virtual environment
Application specification (functional requirements)

High-level abstraction system (process) modeling

Mapping from abstract system information to concrete scenario representation

Object dynamic behavior modeling

physically-based modeling

- fast collision detection and handling to increase the sense of realism;
- mathematical representation and (numerical) solution of physical phenomena;
- data visualization (transformation between data and graphic)

real-time simulation and interaction;

- response of user input (3D manipulation);
- integration of numerical simulation and graphic rendering /data communication in real-time

User interface selection and modeling

1.5 Approaches to the Objective

Our approaches to the objective follow the key issues mentioned above.

First, we begin from application specification. We propose to use IDEF3 as the process description and simulation tool. The system scenarios are captured by IDEF3. Then we convert text based process description to VR representation.

The complete description of the object behavior is missed in the process level. Therefore, there is a gap between VE and the real environment. In order to narrow the gap, we present a hierarchic object dynamic behavior modeling approach, where physically based modeling is the major concern. By adopting Object Oriented (O-O) concept, we consider the object behavior as one integral of an object entity. The
object entity includes the object geometry, behavior and property. Physical law is one of the behavior knowledge presented or triggered by transition condition.

By integrating process modeling and simulation with object behavior modeling together, we developed a framework of efficient construction of physically based VE for engineering design and analysis. Templates of object entities and standard libraries are used to increase the reusability and reduce the lead-time.

1.6 Organization of this Thesis

After the introduction in the first chapter, a literature review will be given in the second chapter. A framework for effective construction of physically based VEs will be developed in the third chapter. Within which high-level process description will be incorporated with multilevel object behavior modeling to construct the VE of a complex system. Standard libraries of object geometry, property and behavior are used to reduce redundant developing work. Following the framework, a hierarchical approach for object dynamic behavior modeling will be presented. The fifth chapter will introduce the VPPG as an implementation example of the framework. The last chapter will summarize the thesis and make recommendations to the future work.
2. Literature Review

2.1 Effective construction of Virtual Environments

A Virtual Environment has been defined as the three dimensional computer generated model or database which can be interactively experienced and manipulated by the user through visual, auditory, and tactile channel. Despite the great potential of VE technology, there are two major problems that must be resolved before VE technology become popular and useful:

➢ There is a need for software infrastructure and tools for effective constructing VEs;

➢ There is a need to develop Object Behavior Modeling techniques to enhance the sense of reality in a VE.

Few literatures can be found in effective VE construction, most of VE construction related references are talking about application specific VE construction. In the Automatic Generation of VR Scripts (AGVRS) from IDEF3 Behavior Description project, KBSI has a first documented attempt to integrate system behavior models with VR scripted simulation models. Input to the conversion algorithm are IDEF3 process flow diagram which contain information about 1) actions, 2) actors who perform the actions, 3) places where the action is performed, and 4) constraints. Output from the conversion algorithm is the API script for ENVISION/ERGO, a COTS VR system from Deneb Robotics, Inc. AGVRS approach resolved rapid VE generation problem, however, the low level, physically based object behavior is unavailable from the high level, process description method. Recently, some VRML
authoring tools can generate static VE easily, but the behavior modeling is always the challenge task. To bring life to VEs, object dynamic behavior modeling is an important issue. Object Behavior Modeling in computer graphics has been studied for a length of time. Among OBM, Physically Based Modeling is the relatively new approach and it can provide a convenient and straightforward way of specifying behavior for a VE.

2.2 Physically Based Modeling

Physically Based Modeling is of crucial importance for enhancing the sense of realism, at the same time it is a challenge in VR. It is worth to have a review on related applications and research. Quite a few applications make use of physical simulation, such as visualization of static spatial environments, the most common applications of VR technology so far. Some examples include the driving simulator in Iowa [Cremer, J. 1996], where the user's vehicle interacts with other independently simulated road vehicles. At the department of defense's NPSNET [Macedonia, M. R., Brutzman, D. P. et al. 1995], "units" of military vehicles engage in simulated combat on static terrain. Each simulated unit communicates its status to each other unit, since the environment is fixed, the communication requirements are bounded by the number of simulation entities. Though these systems may use some actual physical simulations, only a few objects in the environment are actually changing, the computation demand is not too large. Other typical examples of physical simulation are concerned with the physics of every object interaction, such as impenetrability and collisions; they have been used to evaluate the ergonomics of environments like kitchens, automobiles, or workspaces. In these systems, simulation is typically limited to objects being directly manipulated, and the computations are simplified. At the
same time, many virtual visualization systems have been built to allow the user to interact with complex physical simulation, but the user is simply exploring precomputed data, without being able to interactively change the conditions and observe the result of their modification. The degree of "interaction" is limited. Even though, as scientific visualization it is still successful. In fact, scientific visualization is a very important area in PBM. Therefore, we will review it in the next section separately. The most ambitious application of PBM is in full-scale interactive environmental simulation of planned environments from the viewpoint of an immersed human observer such as fire walkthrough and fire simulation. There are two documented related projects, one is from Naval research laboratory [David L. Tate, 1997], another is from UC Berkeley [Richard Bukowski, Carlo Sequin, 1997].

2.2.1 Scientific visualization

Scientific visualization [McCormick et al. 1987] is the use of computer graphics to create visual images which aid in the understanding of complex, often massive numerical representations of scientific concepts or results. Such numerical representations, or data sets, may be the output of numerical simulations as in computational fluid dynamics or molecular modeling. The numerical representation can also be recorded data as in geological or astronomical applications, or constructed shapes as in using computers to visualize topological arguments. These simulation are often containing high-dimensional data in a three dimensional volume. VR technology provides a unambiguous display of these structures through a rich set of spatial and depth cues. Scientific visualization is oriented towards the informative display of abstract quantities and concepts, as opposed to attempting to realistically represent objects in the real world. Thus the graphic demands of scientific
visualization can be oriented towards accurate, as opposed to photo-realistic representations.

There is a relatively rich set of applications on scientific visualization, which have appeared in spite of the immature state of VR technology. These include the Virtual Windtunnel [Bryson and Levit, 1991] and Virtual Spacetime [Bryson, 1992]. Molecular modeling [Brooks et al. 1990], scanning tunneling microscope display and control [Taylor et al., 1993], and medical visualization systems [Bajura et al. 1992] have been developed at the University of North Carolina at Chapel Hill. A system to investigate cosmic structure formation has been implemented at the National Center for Super-computing Applications [Song and Norman 1993]. The Electronic Visualization Laboratory at the University of Illinois Chicago implemented several scientific visualization applications.

2.2.2 Real-time performance

Traditionally, scientific visualization has been based on static or animated two-dimensional images. These images have required a significant investment in time and expertise to be produced. The response time of the displayed image to the numerical simulation result is generally long. We call a system with sufficient performance to respond quickly and interactively with the user a "real-time" visualization system. This "real-time" is distinct from the usage in the phrase "real-time operating system", where demands are far more stringent than real-time visualization.

There are two aspects of real-time performance, which are real-time interaction and real-time visualization. Real-time interaction concerns about interactive response. If the interactive response is too slow, the user will experience difficulty in interaction. Real-time visualization concerns about how fast images are updated to the user, which
is particularly relevant to viewing control for time-varying data sets. In order to achieve real-time visualization, which is determined by animation rate, VR systems need to separate the computation and visualization processes, so that they run asynchronously. In this case, there may be several parameters determining the animation rate, i.e. the simulation rate (the rate at which data for visualization are computed), and graphics animation rate (the rate at which pictures are updated to the user). Generally, graphic animation rate might be the bottleneck, but in numerical simulation, it is also possible that the simulation rate is lower than graphic rendering rate, e.g. simulating the environmental parameters such as light, temperature, air flow for a single frame needs a great amount of time to get the simulation converged.

2.2.3 Collision detection

Objects in the real world cannot occupy the same place, but computer graphics that represent different objects can penetrate each other. When several objects are moving in a VE, there is a chance for collision and penetration. Fast and precise collision detection is therefore required to prevent interpenetrating for the sake of physical reality. Typically, collision detection is a geometric intersection problem, while collision response is a dynamical problem which involves predicting physical behavior of objects. Collision detection is one of the major research topics across computer graphics, robotics, and VR.

A series of methods or algorithms of collision detection have been published. Of the various representation schemes on which collision detection methods are based, Brep scheme which curved surfaces are represented by polygons with specified tolerance is the most commonly seen. The most primitive method for collision detection of polyhedral objects works by performing static intersection tests at discrete time
instants[Kamat, V. V, 1993][Vaneccek, G., 94][Kitamura et al., 1994]. The time interval between tests is assumed small enough so that the collision is less likely to be missed. Another Brep based method for collision detection keeps track of the distance between each pair of objects in the virtual world. If the distance between a pair of objects goes below the pre-defined threshold, a collision is then declared. A noteworthy use of this idea for collision detection of rigid, convex objects is [Lin et al., 1994], where coherence of objects between time instance and the property of convex polyhedral is used to detect collisions among objects in roughly constant time per object pair. The difficulty of this method is that the process to compute the distance between each pair of objects with complex shape is time-consuming.

To localize contact regions between two objects, a data structure called Brep-Index [Vaneccek, 1991] for quick spatial access of a polyhedron was proposed. A back-face culling or velocity-based collision detection method that removes roughly half of the faces of objects from being checked for detailed interference was described by Vaneccek [Vaneccek, G., 1994].

Fang [Fang, 1995] proposed a technique called Hybrid Volume Partition to detect accurate collision in the assembly of two polyhedral solids. In this method, the number of smaller boxes used will greatly affect the performance of the box-to-box partition testing time.

C. J. Su [C.J. Su, et al. 1996] presented an efficient and precise collision detection scheme for CSG represented objects in a VE. The objects are assumed to be rigid solids and would not be deformed during and after collisions. In summary, this method has the following advantages:

- Using natural hierarchical CSG tree structure, and adaptive and dynamic selection strategy for near optimal bounding volumes, it realizes effective and efficient
intersection or contact region localization; i.e., an intersection or contact can be localized into two leaf nodes of each object after limited steps of simple bounding volume intersection test.

- This CSG/Brep representation facilitates the combination of the advantages of the faster, less accurate bounding volumes with accurate distance-assisted collision detection and precise polygon-based intersection test.

2.3 Sense of Reality

People are realizing the gap between a VE and the real world. In essence the gap is caused by poor sense of realism. Therefore, sense of realism is of critical importance in a VE. Generally, realism is concerned with the fidelity of rendering of "reality" into a particular representation. A user can perceive this representation through his senses and formed a mental model of the "reality" provided by the VE. The sense of realism comes from objective behavior modeling and subjective user impression. Two categories of research related with "reality" are carried out, one is behavior modeling and simulating in physical domain, and the other is measuring the sense of presence or immersion in psychological and mental domain. Most of the current VR application research interests focus on the physical domain. As an example, Peter Astheimer [Peter Astheimer, 1994] summarized the state-of-the-art in providing realism in a VE. Techniques increasing visual and auditory presentation were discussed. Object behavior simulation and interaction techniques were also introduced.

Though the measure of realism in psychological domain will not be the research objective of this thesis, it is necessary to get a global understanding in this field since VR is quite human oriented.
The question of the "measure" for reality is a fundamental starting point for theories and measures of VE. Different definitions of "reality" have been proposed by Newton, Bernstein and Gibson. Factors affecting immersion include isolation from the physical environment, natural modeling of object behavior, and perception of self-involvement. Bob G [Bob, G., 1998] conducted some experiments to measure presence in a VE. Pavel Zaborik et al. gave a view of presence based on existential philosophy and ecological psychology where notions of subjective presence and objective presence no longer exist. Presence is then tied to one's successfully supported action in the environment. John M. Flash et al. [John M., Flash and John G. Holden, 1998] reviewed in basic assumptions to measure the experience in VEs and compare the experience of reality.

Because of inconsistent definitions of reality and multiple measurements based on different assumptions, the existence of ambiguity and arguments is obvious. It is helpful to adopt Rory Stuart's [Rory Stuart, 1995] definition:

Immersion: The presentation of sensory cues that convey, perceptually to users that they are surrounded by a computer-generated environment. It is an objectively quantifiable phenomenon that can be described by measures such as visual field of view, auditory localization cues, directions from which force reflection can be displayed, and the range of orientations supported by tracking.

Presence: the subjective sense of users that they are physically in a VE. Although this is a subjective measure, it's also somewhat quantifiable.

Engagement: The subjective sense of users' being deeply involved and occupied with what they are doing. This involves paying attention to the immediate experience and not being distracted by thoughts of other things.
No matter it is in the physical domain or psychological domain, real-time performance contributes to the sense of realism to a great extent. Therefore we give a review on the strategies for real-time performance in the following section.

2.3.1 Strategies for real-time performance

There are no simple, general rules for meeting the performance requirements that can be applied across applications. E.g. real-time performance sometimes pre-computation of derived quantities is required for computational speed, and sometimes computation on the fly is forced by data management considerations. The following issues should be considered.

Computation

The computation demands of a real-time interactive visualization system are determined by the particular visualization technique used in that system. A visualization based on simply color-mapping to an existing geometry requires very little computation. A technique based on particle integration has more computational demands depending on number of points and time steps. Isosurface computation has computational demands depending on the number of points in the data set. Therefore it will be dependent on the size of the data set.

Data Management

One data management compromise is the observation that one is often interested in data within some small sub-volume of the entire data set only. The researcher may be allowed to load the data for that sub-volume only and perform interactive visualization for that sub-volume.
Data compression may be used to reduce the amount of data that needs to be stored. In this strategy, one decompresses the data as needed. Therefore, increased computational overhead is accepted for the sake of increased data capacity.

Another approach to data management is to arrange the data on the disk in such a way that it can be loaded "as needed" at interactive rates.

**Network Overhead in Distributed systems**

A strategy which addresses both computational and data management requirements is to distribute the visualization system across several hardware platforms. In this way resources may be allocated to make the best use of their capabilities. Distributed systems introduce the consideration of the network connecting the system components. There are three concerns: network bandwidth, network latencies, and synchronization of the various components.

**Graphics**

Computer graphics is at the heart of the visualization process. In broad terms, scientific visualization is the mapping of abstract quantities into graphical representations. Vertex processing time and pixel scan conversion time affect graphics rendering performance.

2.4 Interaction

An interaction technique is a way of using physical input devices to perform a generic human-computer dialogue [Foley, 1990]. VR is superior to other forms of human-computer interaction since it provides a real-time environment integrating several new communication modalities. By providing a rich and real-time sensorial interaction, VR makes the user feel immersed in the simulation or application he or she is
running. Actually, there are three levels of interaction: interaction, real-time interaction, physically based interaction.

Sense of immersion or perception of realism is also greatly affected by successfully, physically based interaction. To provide sufficient realism, the simulation must include physical constraints such as object rigidity, mass and weight, friction, dynamics, surface characteristics, and so on. Adding physical property to objects requires both powerful computing hardware and specialized input/output devices.

There are three important concerns for an easy to use and physically based interaction paradigm as described in the following.

2.4.1 Constraining interaction

While VR is 3D, many tasks are 2D. There is a Degree of Freedom (DOF) problem to accomplish a given task. For task like sketching or accurate placement of objects, haptic constraints are highly desirable. It is also well known that friction against a surface greatly enhances control. Tactile feedback is of crucial importance for a grasping and assembly task.

2.4.2 Direct manipulation

Simulation based design allows the designer to work with the environment in a new way, which is called direct manipulation. Direct manipulation is beneficial to the design process. The designer can now link forms, uncouple them again, re-scale them, color them, distort them, and so on, by means of direct manipulation. This is all accompanied by visual, auditory and tactile feedback. This new capability is provided by simulation.
2.4.3 3D tracking paradigms

To perform a task, whether an accurate 3D tracking is available is crucial to the usability of the user interface. In immersive VR systems, position tracking associated with the HMD allows visual and auditory displays to be updated in response to the user's head motion and orientation. Usually a sensor determines the position and orientation relative to a source. Either the source or the sensor may be fixed. The underlying architecture of position tracking systems can be active or passive. In principle, any technology that permits the quick determination of the user's position and orientation can be the basis of a position-tracking system. Several commonly used tracking technologies are electromagnetic, ultrasonic, mechanical, optical etc.

From a perceptual point of view, problems with position tracking can lead to a number of difficulties for the user. e.g. Lag can contribute to overall system latency, which has been suggested as one reason of cyber-sickness. Poor accuracy and resolution can make fine motions required by some control tasks difficult or impossible. Some important terms are introduced:

**Accuracy/resolution:** For position trackers, resolution refers to the smallest movement that can be detected by the system, while accuracy describes how close the reported position is to the actual position.

**Responsiveness:** Responsiveness can be conveyed in terms of sample rate, data rate, update rate and lag [Meyer, et al., 1992]. Meyer et al. defined sample rate as the rate at which sensors are checked for data, data rate as the number of computed positions per second, update rate as the number of new position reported by the tracking system to the host computer per second, and lag as the delay between the movement of a remotely sensed object and the report of its new position. Sample rate often exceeds data rate because sensors are checked more often than the position is calculated; if the
tracking system processes raw data to reduce jitter or predict motion, the update rate might not directly reflect the measured position.

Tracking errors can be crucial to human-computer interaction. Their impact can be presented by Fitts's Law. According to Fitts's Law, when reaching for an object, the time (t) to reach the object is described as: \( t = a + b \log(2D/W) \)

where \( D \) is the distance from the initial position of the hand to the object, and \( W \) is the size of the object.

Lag affects tracking of a target: \( b \) is linearly proportional to the speed of the object tracked. Lag affects interaction by increasing \( b \) in Fitts's Law.

2.4.4 Models of tracker motion

Position tracking permits users to experience the sense of presence in the VE. It uses six coordinates to describe the position and orientation. A model of motion takes an input history of position, velocity and acceleration and predicts a future position. e.g. ballistic motion with constant acceleration \( a \):

\[ x(t) = x(0) + v(0)t + (a-t^2)/2 \]. Human motion is difficult to model in general. The most frequently used human motion is hand motion, which is capable of fairly high and unpredictable acceleration. It is very difficult to predict in the model of hand motion. Head motion is comparatively easier to model.

2.5 Summary

Physically Based Modeling has been intensively studied in computer graphics to improve the realism of visual models. Scientific visualization is among the most widely used and at the same time is an important application of PBM. However, to effectively develop a physically based VE still remains a research issue due to its
complexity and the limitations of hardware and software. We are lacking of a general approach for dynamic behavior modeling (PBM is one instance of dynamic behavior modeling), which causes a gap between VEs and real environments. In addition to dynamic behavior modeling, an intuitive, user-friendly interface for Human Computer Interaction is also crucial to increase the sense of realism and improve the quality of VEs. Very little literature in PBM has performed physical simulation of human interaction, and literatures related with interaction or Human Computer Interface discussing PBM in detail have seldom been found. In this thesis, human interaction is considered as one kind of dynamic behavior, and physical simulation is therefore introduced to the modeling of response of human interaction. Though eventually the visual user interface will be a VR enhanced immersive environment, it is currently a challenge to apply VR technology to Complex System Engineering. VR applications failed to gain its popularization in the engineering discipline due to high cost and long development time. Little research has been done to fulfill the requirements of rapid generation of VEs for engineering design and analysis. A standard approach for dynamic behavior modeling, and a framework for effective construction of physically based VEs are therefore highly demanded.
3. Framework for Effective Construction of Physically Based Virtual Environments

3.1 Introduction-simulation based design and VR technology

Simulation has been used in the design of manufacturing systems for a long time: determining the layout, selecting resources and buffer sizes, creating routings, optimizing scheduling, and defining the number and type of operators needed. The emerging field of VR and in particular, the use of VR for virtual prototyping and Simulation Based Design (SBD) has created a unique opportunity for addressing technical challenges in complex System Engineering [SE]. Although researchers have been aware of the opportunity to apply VR and SBD for SE for several years [Fan Dai, 1998], little has been done at the research level and the application level to realize a happy marriage of these two areas of knowledge.

Current VR applications focus on simple processes or a single activity, as will be described in the next chapter in reference to Object Behavior Modeling. It is necessary and valuable to develop VEs for complex processes and systems to make VR really useful. e.g. Visualization technology makes complex processes and systems manageable because a dynamic scene can reveal not only a list of events (activities), but also information about those events and their relationships one to another. Spatial, temporal and logical relationships are visualized within the appropriate context to enable users to better manage complex processes. However, the complexity of problem itself limits the application of VR in this field. There are several technical and pragmatic barriers that hinder the integration of VR with SE.

- Semantic barriers and communication inefficiencies: A central problem with the integration of VR with SE is the semantic gap between different "role types"
involved in the VP-enabled SE life cycle processes. Domain experts/end users articulate needs and requirements, systems engineers/knowledge engineers refine and flow down requirements and perform conceptual/detailed design and design explorations, software/hardware engineers and programmers perform implementation design, prototyping, and code refinement. The semantic gap between the background knowledge that each of these role types brings to the table imposes significant communication requirements and inefficiencies owing to communication losses. Technology integration is a key approach to address this problem.

- **Disparate tools and technologies:** Another problem with SE and VR integration is the diversity of methods, tools, and technologies that are available. The integration lacks of mechanisms that facilitate re-use of data/knowledge across multiple technologies (e.g., data models developed in CASE tool are not readable by VR modeling tools), interoperation of applications (e.g., a HLA simulation tool cannot communicate with a shop flow control simulator), and consistency maintenance (e.g., changes made to a functional model reflecting changes in system requirements cannot be automatically propagated to implied changes in the product design model in the VE).

- **Lack of methods and standards:** A significant problem (particularly from a pragmatic perspective) is the absence of structured, standard methods that describe/prescribe how to effectively combine the techniques and tools from SE and VR.

Due to these issues, VR is much too expensive and time consuming for a complex system. It is of paramount importance to develop a framework for effective construction of VEs to model a complex process or system.
This framework needs to handle from high-level process description to low-level behavior modeling. These two important components for process simulation have to be integrated to fulfill the functional requirements. Very limited literature on this integration is available. As the first attempt of combining the process description and behavior modeling, a framework for effective construction of physically based VEIs is proposed. This framework follows a top down, hierarchic approach for process description and behavior modeling. In addition, uses a hybrid coordination method for objects and events in a system or a process. In the next section, we will explain our approaches.

3.2 Strategies of the Framework

The basic idea of the framework is: We use IDEF3 to capture the system scenarios, which consist of an ordered sequence of activities - description of a process flow in terms of its temporal, casual, and logical relations. For the low-level, physically based behavior modeling, we use the OBM based on physical simulation. This framework enables developers to build a VE that integrates system and enterprise level abstract qualitative processes with fine-grained quantitative process information required for high fidelity engineering analysis and evaluation. To validate this framework, integration of OBM and process modeling, to be more specific, coordination between entities and different levels must be addressed.

There is an implementation gap between high level process description and detailed modeling of behavior. The process description is incomplete and possibly inaccurate and not amenable to prediction. The Object Behavior Modeling (detailed modeling of activities) is in a comparatively low level, with more information about how the activity is carried out, instead of global information about the system coordination.
Our proposed approaches to bridge the gap are:

Add an interface between process description and modeling of detailed activities;

Integrate entity behavior modeling into the process description.

As shown by Figure 3-1, we have the sequence of activities in the process level. Inside each activity, there are information about objects and behaviors. According to that information, we can go to a lower level - object entity behavior level. The detailed mapping will be discussed when introducing the architecture of the framework.

![Figure 3-1 Mapping of process description and behavior modeling](image)

3.3 Motivation of Integrating Process Description and Behavior Modeling

Modeling the high-level behavior is difficult without consideration of the whole system/process. Knowledge Based Systems Inc. (KBSI) has proposed a VR-Enhanced Integrated Process Design Environment (VR-IPDE), an architecture utilizing IDEF3 as process modeling tool and automatically generating VR scripts, which is capable of describing high-level system behavior. However, it is limited to high-level simulation, and low-level, physically based behavior is unavailable from VR-IPDE. Considering
combination of these two approaches (process description and behavior modeling) might produce a satisfactory result. While the benefits of incorporating OBM to a process simulation are easy to understand, the relationship between these two is symbiotic. The behavior knowledge base adds the flexibility and capability to handle local disturbances in the process; the process description will provide an organization mechanism between entities.

A framework stratification into higher-level and lower-level components is a reflection of differences in information needs. We use the same example of a virtual agent in a VE as Professor Thalmann [Thalmann, 1994]. One common low-level behavior is responsible for causing an agent to locomote from one position to another. Locomotion decisions require very detailed information - e.g. foot positions, distances, and angles. Differences in information needs provide one reason to separate higher-level and lower-level components, but there are other differences as well. One is level of commitment for future action. In locomotion, while each footstep brings the agent closer to its current goal, no commitment is made to steps that have yet to be taken. In contrast, lower-level information makes commitments to act in specified ways in the future. E.g. when the agent decides to walk to the door, he has already committed to opening the door once he has gotten there. While such commitments are tempered by possible changes in the environment, as long as a commitment remains active, it can affect the agent's decision-making and behavior. What professor Thalmann addressed is the virtual human's behavior, here we want to broaden the behavior scope to general systems.

While both the higher-level and lower-level components of the system are environmentally responsive, another difference motivating the separation of these two levels is the "granularity" if their reactivity, in both the form and time-scale of the
environmental feedback they depend on. Low-level behaviors are tightly coupled to the environment; higher-level components only receive feedback from the world entities, when the lower-level systems choose to relinquish control.

3.3.1 Process description and behavior modeling

Process description is focus on the top-level behavior of the system. Behavior of a system can be decomposed into several levels. The importance of hierarchic approach for behavior simulation has already been recognized. e.g. Nadia Magnenat Thalmann [Nadia Magnenat Thalmann, 1997] stated the reason that separating low-level motor control from high-level planning decisions: a symbolic planner is inappropriate for making decisions on some detailed activity such as foot placement, and likewise local potential field calculations are inappropriate for making long-range plans. Before we can integrate Object Behavior Modeling and process simulation, we should discuss their characteristics.

In industrial applications, object behavior is generally governed by rules such as production rules, operation rules. Physical law is among those rules. There is a basic conflict between rule-based and O-O programming as suggested by Kim W. Tracy. The rule based paradigm depends on a separation between the rules (which dictate some of the behavior), the facts and data of the problem. In fact, this separation is one of the strengths of the rule-based paradigm because it allows the system to use the same rule-base to solve many different problems. In the O-O paradigm, an object behavior and data both must be specified as part of the objects class definition. However, rule-based programming can be integrated naturally into the O-O paradigm by transition conditions. The O-O paradigm can also be extended to explicitly support
a specific type of behavior, embodied in rules. Therefore the conflict between rule-based and O-O programming is not a serious problem.

Though we can model object behavior from high-level to numerical detail, this high-level behavior is still limited to individual entity level.

Although O-O approaches provide modularity and conceptual organization, in large-scale applications they can result in complicated property and method variants, generating hundreds of object classes and forming a complex inheritance network. Traditional O-O methodologies expect all of the interactions, between objects and across all possibilities to be specified at least conceptually beforehand. While in human involved complex systems, interactions can be context dependent and unpredictable.

Existing research either focuses on low-level physically based Object Behavior Modeling or on high level process animation. Very little work has been done to satisfy both high-level and low-level components. OBM can represent how objects change their states over time in different levels, from top level decision making (of the entity) to the modification of object's state, but does not have the capability of modeling the high-level system behavior (e.g. OBM generally lack long-range planing capabilities). In addition, the coordination between objects such as back and forth messages could be very complicated and need to be predefined. How to organize individual objects (entities) to form a complex system is beyond the domain of OBM. Therefore, there is a need for high level system oriented organization mechanism.

On the other side, process simulation is capable of organizing a complex system in high level, such as IDEF3 based discrete event simulation. Like KBSI, we also propose to use the Air Force IDEF3 Process Flow and object description method [KBSI 92] as the higher-level, qualitative process description mechanism. Available
process or system modeling tools are two-dimensional icon based such as IDEF3 or Arena, they fail to illustrate the detailed physically based behavior, without mention of 3D simulation. The following section will introduce the IDEF3 method.

3.3.3 IDEF3 process description method

The notion of a scenario or story is used as the basic organizing structure for IDEF3 Process Descriptions. The process-centered strategy organizes process knowledge with a focus on processes and their temporal, causal, and logical relations within a scenario. IDEF3 Process Flow Diagrams are the primary means for capturing, managing, and displaying process-centered knowledge. These schematics provide a graphical medium that helps domain experts and analysts from different application areas to exchange knowledge about processes. This includes knowledge about events and activities, the objects that participate in those occurrences, and the constraining relations that govern the behavior of an occurrence. The example shown in Figure 3-2 depicts a process flow diagram of a scenario entitled Order Material. In this example, the owner of a business used IDEF3 to document the material ordering process to assist new workers in their training and to enforce company's purchasing standards. In particular, the owner wanted to record how Purchase Requests are processed for the benefit of new employees.
Figure 3-2 Example of a process flow diagram

The processes in the owner's description are represented in the schematic as labeled boxes numbered 1 through 10. Each box represents a distinguishable packet of information about an event, decision, act, or process. That is, boxes represent types of happenings. Such happenings are referred to by the neutral term Units of Behavior (UOBs). Each UOB represents a real-world activity. The information recorded about a UOB includes (1) a name (often verb-based) that indicates what action the UOB represents, (2) the names of the objects that participate in the process and their properties, and (3) the relations between objects. Arrows (called links) connecting the boxes in Figure 3-2 indicate the precedence relationships (or more generally constraints) between the activities being described. Thus, an instance of the UOB at the source of a link has to be completed before an instance of the UOB at the end of the same link starts. In Figure 3-2, the UOB labeled Request bids has to be completed before the start of the UOB Evaluate bids. The small box containing the "XOR" denotes a junction. A junction is a point in the process where a process splits into
multiple paths, or where multiple paths merge. Junctions represent constraints (or the effects of constraints) of the *activation logic* for the process. For example, the first junction in the above figure indicates that only one path will be taken in an activation of the described process.

The IDEF3 method allows users to capture descriptions at varying levels of abstraction by providing a mechanism called *decomposition*. A *decomposition* provides a means of organizing a more detailed description of a UOB. The decomposition schematic follows the same syntactic rules as those for a scenario and is created using the same IDEF3 elements. A UOB can have any number of different decompositions, all on the same level. The use of more than one decomposition for the same UOB is for the purpose of representing different points of view or providing greater details of the processing relating to the UOB. The UOB *Request Material* in Figure 3-2 has been *decomposed* into UOBs 7 to 10. The numbers in the lower-left corner of UOB boxes 7 to 10 include a reference to UOB 1 (the first digit) and the decomposition (decomposition 1 of UOB 1). This is illustrative of the IDEF3 numbering scheme, which allows explicit traceability between Levels of Detail in the description. The process description depicted in Figure 3-2 shows the material ordering process from a particular point of view—that of the business owner. It is possible to conceive this process from other views e.g., that of the Account Manager. Each view to be described would be presented in a separate decomposition with a unique label and number.

**Strengths of IDEF3**

As described above, IDEF3 is a powerful tool for high-level process description. It provides structured method for knowledge acquisition from domain experts. IDEF3 is also a good communication platform for people involved including customer of the
product (if the process is used in a product manufacturing), operator (actor) of the
process, domain experts, manager and decision-maker, system designer and VE
developer. Above all, IDEF3 captures description of a system in terms of its high-
level dynamics.

**Limitation of IDEF3**

Though IDEF3 is a powerful description and even process simulation tool, it is not a
suitable tool for LOD behavior modeling required by VEs. Limitations of IDEF3 are:
Continuous activities can not be captured;
A set of scenarios cannot be coalesced into a simulation model;
Completely defined models, as opposed to descriptions, cannot be represented;
Dynamic behavior can not be described, as a result, conditional state-based decision
making is not supported;
Temporal relations among concurrent UOBs cannot be captured (junction types are
inadequate).

**3.4 Architecture of IVECS**

By combing process description and behavior modeling, we developed a framework
for effective construction of physically based VEs. This framework provides an
integrated process/system modeling environment in which system-level, qualitative
process descriptions are integrated with the detailed, quantitative process/activity
descriptions. The abstract, qualitative process description is captured by IDEF3
process modeling tool and used to guide the overall design of VE.

From a functional view, this framework can be divided into three subsystems: data
transducers, VR authoring tools, and computation system. The operation system is
involved with all transactions. The arrows indicate the direction of data flow. The
transducing system (includes the user, sensors and display) convert human natural behavior into digital information, and further digital information into visual presentation. The user (actor) of the system interprets the VE perceptually and performs activities according to the process description and his own perceptual interpretation. Sensors convert human actions into binary data, mapping the physical action to the VE with position tracking, voice recognition, gesture interfaces, keyboards and joysticks. The visual display converts the digital model expressed as display stream instruction into subjective sensory information perceived as sensation by the user (actor).

The VR authoring tools coordinate display and computational hardware, software, and VE models. This subsystem provides techniques for navigation, manipulation, construction, editing, and other forms of user interaction. Object models (entities) encapsulate their appearance and behavior in the VE. During run time, entities' states undergo constant change due to parallel transactions, search-by-sort processes, and function evaluations.

As illustrated in the following Figure 3-3, the framework can be divided into several levels.

The application specification level: This is the very beginning of a VE design. In this level, the goal of system being modeled must be specified and the goal can be decomposed into a sequence of activities. Generally it is accomplished on paper or just in the system developer's mind. But for a complex system/process, it is beneficial to make use of simulation software. Here we suggest using IDEF3 for process modeling. After the process modeling, the objects, actor(s), UOB and relations between objects and actor(s) are specified. This part needs no programming effort.
Figure 3-3 Architecture of IVECS
IDEF3 provides a communication platform for people involved in the design of the VE, including system users, designers, domain experts, and developers.

VE construction level: In this level, we suggest making use of standard libraries for graphics (sphere, cube, cylinder etc.), commonly used property (such as material, color, and texture, and perhaps mass and size), basic behavior (such as moving along x, glowing red). Behavior library can be increased during development. Each time a new behavior is modeled, it can be added to the library as a template. Using the information in process level about each activity, object behavior models can easily be assembled. This part still does not need programming. However, the behavior modeling is not completed yet without underlying simulation and coordination.

Interaction level: Human being is involved in the VE system, he will either act as a special object called avatar (a representative of the user) that will be modeled in the VE or as an observer that is invisible in the VE (That is the reason why there is a dashed arrow that connects actor with graphic library). Human behavior is partially determined by the system in application specification level (such as perform a certain kind of activity), but some low-level behavior is still a result of Human-Computer-Interaction, and human behavior such as hand motion is unpredictable. Users are interacting at this level with visual models of object entities in the system. Commonly used representation of humans is a hand or tool that is used to operate on the virtual machine. In our prototype system, as it will be described in the next chapter, the virtual paddle is controlled by the user and is an active object that represents the human. Therefore, the hand or tool can be the actor directly.

Reasoning and simulation bottom level: User input, local behavior knowledge base of objects and global system knowledge are integrated in this level to simulate a system
behavior. Visual model (through graphic engine), reasoning and simulating (through simulation engine), and Input-Output drivers are integrated to form a system.

3.4.1 User interface design and modeling

In the framework, user interface plays an important role and is an integral of the whole architecture. To integrate human into the computer-generated world, hardware and software need to be especially tightly coupled. In order to encode the interface into the environment, developer can customize existing tools according to task requirements, or create a new one. From the process description point of view, the user interface behaves as an Immersive Process Detailer (IPD). We will adopt the IPD technique developed by KBSI.

There are two meanings of interface here. VE itself can be considered as a Human Computer Interface for user interaction. Traditional human-computer interaction is described in conversational terms rather than behavior terms. As Walker commented on conversational versus exploring metaphors: "Conversation is the wrong model for dealing with a computer... When you are interacting with a computer, you are not communicating with another person. You are exploring another world." The development of VP technology for interacting and manipulating symbols may represent the next step in the evolution of tools. Using VP technology, we will be able to enter into and directly interact with the world of symbols instead of looking passively at the symbols from the outside. To use the VE as a tool, or make the tool useful, it is necessary to integrate the capability of machine (computer) with the capability of humans. Another meaning of interface is the interface of VE, more specific to indicate the input and output devices and style.
In the context of VEs the interface techniques are generally using a Head-Mounted Display (HMD) for viewing stereoscopic images, a spatial tracker for locating the position of the head and hand, a equipment to spatialize sound, and a equipment to provide tactile and force feedback to the users.

**Immersive Process Detailer (IPD)**

The IPD will employ VR technology to immerse the user in a VE to perform activities according to a high-level, abstract process description such as an IDEF3 process model. The user's body motion information such as 3D translation and orientation of the head, limbs, and hand is tracked by sensors attached to a HMD, body suit and data *Dataglove*. The user's body motion data in performing a process activity will be collected into an associated log file which, after correcting measurement distortions, is used to generate body movement control data for popular COTS human models such as JACK© and ERGO©. Figure 3-4 shows how the IPD user specifies detailed process information.

![Figure 3-4 Immersive Process Detailing](image_url)
3.4.2 Hybrid architecture of coordination

As mentioned before, coordination between different simulation entities in the VE and coordination between different levels is very important to make the VE system alive. There are two levels of coordination, one is high-level global interrelations between individual entities. The other is hierarchical coordination between OBM, simulation engine and behavior knowledge. The high-level coordination can be achieved by using IDEF3. IDEF3 captures the description of what a system actually does by modeling scenarios. High-level coordination is described as process flow in terms of its temporal, casual, and logical relations. Precedence, relationship and object flow are described by different types of link in IDEF3.

(IDEF3 Process Description Capture Method, 1992)

Coordination between object models, simulation engine and knowledge engine will most probably be carried out in source code level. The framework will not provide the detailed coordination information. Intelligent agent will be a kind of solution, but to be more reliable, the VE developer (programmer) can perform the coordination and integration between different levels and different entities. The coordination (communication) can be described as Figure 3-5.

Collaborative communication occurs in the form of request and reply messages. An object entity is not required to response to any message it receives. This leaves each entity the freedom to decide when and to whom to react. Any request always contains the entity's identifier, the entity's contact information (for replies), the ontology used in the request, and a request identifier generated by the agent issuing the request. The identifier is necessary since an agent may send out multiple requests simultaneously whose replies may arrive out of order.
3.5 Benefits of Integrated Virtual Environments for Complex Systems

3.5.1 Immediate and long term significance

This framework provides a structured, standard method to combine the techniques and tools from SE and VR. The idea of Integrated VEs for Complex System (IVECS) will produce a profound impact on three areas: 1) Systems Design Synthesis, 2) Systems Evaluation and Assessment, and 3) Systems Reengineering and Reuse.
Benefits to the Systems Engineering/design process

First, IVECS will result in reducing the time and cost to developing system designs. These benefits accrue because of the time and cost reductions owing to the integration of SE tools with VP/VR tools. (For example, the use of standard behavior and property libraries will significantly reduce the time and effort needed to design and develop a system). Second, it will allow for the exploration of a larger number of design alternatives. This is due to the tight integration of system behavior models with the VR simulation models: it is possible to explore many more operational scenarios through rapid generation of executable simulations from system process models and repositories. The integration between the immersive process analysis and the process model information will also help in increasing the productivity of the immersive virtual design exploration task.

Benefits arising from the delivered higher quality system designs

First, the IVECS will deliver higher quality first-time designs. This will significantly reduce the life cycle system cost and risk, and increase life cycle system performance. These benefits accrue because of designs that are more maintainable, have fewer operational problems (because the maintenance and operations processes (including failure modes and exceptions) were successfully "debugged" during the design phase). Second, by making integrated VR-enhanced models available to operations and maintenance personnel, the system can be operated more effectively (for example, by using the VR-models of the maintenance process for doing "augmented reality" diagnosis and repair).
Benefits on better user performance

First, the IVECS facilitates the sharing of a common understanding. Second, the system provides visualization and communication support to different user. Decision-making, in particular those relying on depth, height and geographic data, can be aided by 3D environment.

IDEF3 is a powerful process-modeling tool, VR is a strong human-machine interface, a combination of these two creates a good opportunity for addressing technique challenges for Complex System (CS) engineering. This framework is general enough for applications ranging from CS engineering to mechanical engineering. Users of IVECS can be system engineers, business managers, product designers, product customers, trainees of training applications. Different users of this framework can benefit from different levels: from business (process) managers point of view, this system allows decision making regarding development alternatives and tradeoffs; from manufacturing engineers point of view, this system enables users to assess the impacts design changes and all possible scenarios will have on products as they would in the real world; from customers (manufacturing field) point of view, this system makes it easy to configure their needs for a specific product; from end users' (VE for training) point of view, this system provides a realistic, convenient (or safe) environment for navigation.

3.5.2 Advantages resulting from the framework

An integration of qualitative, quantitative, and immersive process analysis has been completed. The framework has developed an innovative method and implementation for synergistically combining qualitative, quantitative, and immersive process analysis techniques. Use IDEF3 to model and simulate the process first, will help to point out
design problems at the early stage. If the system designer is satisfied with the simulation result, then the developer will continue to refine the system design and assemble the Object Behavior Model. In another words, a 2D simulation has been conducted beforehand.

The basic behavior information such as actors, objects involved, Unit of Behavior (mapping from an activity) is obtained from process level. It should simplify VE authoring by allowing behaviors to be specified at a higher level of abstraction. Standard library helps to reduce redundant programming for some basic behaviors and properties by different developers, and thus increase the reusability. In addition, the behavior would be an integral part of the object, just as important as its geometry or appearance. The graphic, property and behavior of an object entity are tightly coupled together. Actually, this idea tends to be widely accepted. e.g., Martin Baker proposed a VRML editor that allows nodes to have physical properties in addition to appearance and geometry. Properties such as mass, velocity, etc. would allow the node to move and interact with other nodes in the world without any specific code to define this behavior. (http://easyweb.easynet.co.uk/m.baker/) Platinum's VRCreator is capable of including standard library of geometry, behavior in the authoring tool already. We need also note that it is decided by the VE developer whether or not to add property or behavior to a specific object. For some objects such as the wall, it is unnecessary to model the gravity react on it unless you are constructing an virtual room to validate the structure.

3.6 Summary

A framework for IVECS is developed, which combines the top down and bottom up approaches for system design and behavior modeling. We use IDEF3 to capture the
system scenarios, which consists of an *ordered sequence* of activities; For the low-level, physically based behavior modeling, an O-O hierarchical OBM approach which will be developed in the following chapter is used. This framework enables developers to build a VE that integrates system and enterprise level abstract qualitative processes with fine-grained quantitative process information required for high fidelity engineering analysis and evaluation. IVECS performs as a communication platform for system designers and developers, domain experts, and users. The framework allows complex SE to utilize advanced VR technology, and will produce comprehensive impacts on both VR and SE field.
4. Dynamic Behavior Modeling

4.1 Introduction-system dynamic simulation

When we develop a VE for engineering design and analysis, we are facing a system or a process. System simulation plays an indispensable role in using VE as a Human Computer Interface.

Creating simulation models necessitates in-depth knowledge of how a system operates. To be able to predict how a system will react to changes, we need to gain knowledge about its dynamic behavior - how it organizes itself over time in response to imposed conditions.

The dynamics of a system/environment are the rules of interaction among its components describing their behavior as they exchange energy or information. Typical examples of interaction rules can be: differential equations of Newtonian dynamics describing the motion and interaction between different objects; grammatical rules or even of look-up tables for pattern-match-triggered action rules.

Following the previous chapter which develops a system oriented framework for effective construction of VEs, this chapter will focus on the Object Behavior Modeling.

4.1.1 Entity characteristics

Entities are the objects in the simulation, which move around, change state, affected and being affected by other entities and the states in the system. In a VE, there are static and dynamic entities. Instead of simply discussing static versus dynamic
entities, it is useful to divide entities according to behaviors that are "deterministic" and "non-deterministic".

In this context, describing an entity's behavior as "deterministic" simply means that its state is a function only of time; in other words, we can determine the complete state of the entity at any given simulation step. Entities with "non-deterministic" behavior, on the other hand, are by nature unpredictable. Human beings fall into this category; we do things for our own (internal) reasons, and it is impossible to predict what a human being will do at some arbitrary time in the future. It is also not practical to "rewind" human behavior to some point in the past.

Not only human beings exhibit non-deterministic behavior, but also any entity that has a glimmering of simulated intelligence will be similarly unpredictable. If we create a virtual animal, and we attempt to realistically model the behavior of the real animals, then the virtual animal's behavior will be non-deterministic.

Moreover, any entity that responds to interaction (direct or indirect) with a non-deterministic entity is non-deterministic (since its state is unpredictable, being the result of unpredictable actions).

Each basic type (deterministic or non-deterministic) can be further subdivided. Entities with deterministic behavior can be "static" or "animated", and entities with non-deterministic behavior can be "Newtonian" or "intelligent".

**Static entities**

A static entity is deterministic. Its state never changes, and therefore its state at any given time is known. Mountains, buildings and any kind of permanent structure fall into this category. We call an entity static when it will not change in a simulation.
Animated entities

Unlike a static entity, an animated entity changes state over time. However, the changes are easily predictable, and are a function only of time and possibly a set of pre-defined behavior parameters. Examples of animated entity are the hands of a clock, the moon in its orbit around the earth, an animated texture map for a waterfall, or the ambient light level of a world as it goes through a virtual day-night cycle.

Newtonian entities

Unlike static or animated entities, Newtonian entities respond to changes in their environment. However, they do so in a very straightforward, generic way. You can pick them up, put them down, drop them, throw them or bump them against each other and they will react according to the laws of whatever virtual physics the system developer implemented. They have no "volition", no goals, no appearance of intelligence. They simply respond to stimuli. Chairs, flashlights and chess pieces are Newtonian entities.

Intelligent entities

Intelligent entities appear to have a free will. They have specific goals, complex behavior, and are inherently unpredictable. Obviously, human beings fall into this category; so do birds in flight, spiders crawling on a wall, or any virtual creatures.

4.1.2 Levels of behavior

In the context of VR, behavior includes the change of object location, orientation, attribute and its relationship to other objects. Behavior also encompasses the responsiveness of objects to react to each other and to the user. Generally speaking,
dynamic behavior consists of the way each object in a system interacts, functions, responds or performs. We need to develop a general approach for behavior modeling.

As described earlier, every entity has a set of "attributes" (e.g. location, orientation, shape, etc), and behaviors are things that alter those attributes over time. Just as it was informative to divide entities into different categories, it is useful to divide behaviors into different "levels". This division is directly related with the hierarchic approach for object behavior modeling in the next section.

For example, consider a virtual animal. It is capable of several "high level" behaviors, such as foraging for food, evading a predator, or finding a mate.

Just as we had a taxonomy of entities, we now have a hierarchy of behaviors. It has three levels, roughly corresponding to the types of entities we identified earlier.

The three levels of behavior are as follows:

Level 1: change an entity's attributes over time, e.g. set_velocity U= (30, 20, 14)

Level 2: combination of above levels of behaviors to perform some task, e.g. open the door

Level 3: top-level decision making, e.g. "decide whether to forage, flee or find a mate"

The level 3 behavior has the job of selecting level 2 behaviors; it sets priorities and does "executive level" decision-making. The level 2 behaviors decompose a task into simpler level 1 actions, which in turn update the actual entity state.

At the very highest level of "virtual intelligence" lies the process of selecting one of those behaviors. This selection may be based on any number of factors. For example, the decision of whether to look for food or find a mate is based on the current "hunger" value of the animal. The decision to fight or flee when encountering a predator is based on the animal's speed, strength, distance to the predator, and so on.
There are a number of ways of modeling the "behavior-selection" level. For example, we can implement each of the high-level behavior as a single state in a state machine; the entity would be in the "feed" state until the presence of a predator triggers a transition to the "flee" state. This works, but is a little difficult to scale; every time we add a new state, we (potentially) have to add a new transition to that state from each of the existing states.

Another approach suggested by Eben Gay, of ERG Engineering is to treat each behavior as a task in a multi-tasking system, and the behavior-selection mechanism as the scheduler. The scheduling of each task would be based on entity-specific parameters such as hunger, fear, and so on. The animator also needs the ability to control the objects at different Levels of Detail. No matter which approaches to adopt, without a systematically organization scheme, modeling high-level behaviors might be difficult. From the standpoint of system simulation, modeling entities behavior without consideration of the system dynamics will cause chaos. Bottom up approaches for entity behavior modeling alone will not meet the requirements of system dynamic behavior modeling. A top down approach is also in need, as we have discussed in the previous chapter.

4.2 A Hierarchical Approach for Dynamic Behavior Modeling

The behavior of the system and system components (entities) are very important for the realism of VE. The real world is dynamic. Therefore dynamical behavior modeling is the main approach to enhance the realism of VE. In this thesis, we concentrate on physically based modeling of the behavior of a rigid body. From the Ping-Pong Game, we could outline the key issues, the problems and corresponding
solution on physically based modeling in a VE. Based on the research we have done, we propose a hierarchical approach for generic dynamic behavior modeling.

4.2.1 Related work

Dynamic behavior modeling is a very important research field in VR applications. The use of dynamics and other types of physical simulations is rapidly becoming a standard technique in computer animation [David Baraff, 1993, Jessoca K. H. et al., 1995, Brain Mirtich, 1996, D. Terzopoulos, 1994]. This can be seen from the number of papers presented at recent SIGGRAPH conferences related to this topic. Prof. Demetri Terzopoulos published some papers on dynamic computer graphics [P. Faloutsos, 1997, D. Terzopoulos, 1996, 1994], which mainly investigated deformation of graphics. Much of the dynamical modeling in VR is actually dynamical behavior animation, such as artificial fishes [D. Terzopoulos, 1994], or humanlike characters [Jessoca, 1995]. Hanqiu Sun [Hanqiu Sun, 1997] presented a relation-based model for dealing with important issues of scene animation. Several approaches have been proposed for behavior animation:

The sensor-effector approach (Braitenberg, 1984; Wilhelms and Skinner, 1989) is among the first approaches to behavior specification. An object's motion in a VE is based on how the environment is sensed and how the information is processed through the neural network. This approach simulates the way humans and animals normally perform in the real world.

The rule-based approach is another solution to the problem of behavior specification. A set of behavioral rules is used to control and describe the object action, such as when and what actions should be performed, and how the action should be performed. Rules selection can be represented by a decision tree, where each branch contributes
one control alternative. Alternative branches rank the order of importance for selecting a particular motion, depending on the weights and thresholds that are used by the behavior rules.

Another approach is based on the use of predefined environment (Ridsdale, 1988). This approach is mainly used in applications with static environments. A typical application is to select one optimal path in the environment.

Most of the work is ad hoc, little attention has been paid on reusability and extension due to the complexity and difficulty. There is no standard or general approach for Object Behavior Modeling. On the other side, outside of VR, there are many behavior models available for different applications, such as OBM which is Object-Oriented (O-O) and some other non O-O behavior models. There are a number of different system behavior models:

Operational models describe behavior in terms of the system operation;

Structured analysis models use process-interaction (data flow diagrams) to model behavior;

Stimulus-oriented models describe behavior by stimuli (single stimulus or sequences of stimuli);

Finite state machines denote all possible states of an object and transitions between these states to model behavior;

Decision-based models use decision trees and tables;

Petri-nets.

Such behavior models are more or less used in VR applications.

The following table presents some differences and similarities between Object Behavior Modeling (OBM) and Petri nets which is a widely used method for behavior presentation.
Table 4-1. Comparison of behavior models

<table>
<thead>
<tr>
<th>Feature:</th>
<th>OBM</th>
<th>Petri nets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
<td>Can take time to</td>
<td>Time not supported</td>
</tr>
<tr>
<td></td>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>Triggers</td>
<td>Condition/Event based</td>
<td>Event based</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Interobject &amp; intraobject</td>
<td>Interobject</td>
</tr>
<tr>
<td>Exceptions</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Real-time constraints</td>
<td>Yes, in triggers and actions too</td>
<td>No</td>
</tr>
<tr>
<td>Generalization /Specialization</td>
<td>State net / Method</td>
<td>Method</td>
</tr>
</tbody>
</table>


From the comparison we can observe that OBM is more suitable for the physical simulation since it takes time into consideration. OBM can be used to describe the behavior of objects in a wide variety of projects ranging from simple input systems to complex real-time, concurrent, distributed systems. Therefore, it will be extremely helpful in complex VR system design.

4.2.2 Object-Oriented dynamic behavior modeling

In order to develop a framework for dynamic modeling in VR applications, we present a hierarchical approach, which introduces O-O concept to VR system. Concepts of the O-O approach are used to define the general organization of a system.
(classification and dynamic instantiation, encapsulation, distinction between the specification and the implementation, inheritance, dynamic linking and polymorphism) and also to define the data processing dimension of a system (instance variables and methods).

Traditional object modeling techniques use three kinds of models to describe a system: The object model, describing the objects in the system and their relationships; the dynamic model, describing the interactions among objects in the system; and the functional model, describing the data transformations of the system. The overall approach is more function-oriented. In function-oriented methodologies, primary emphasis is placed on specifying and decomposing system functionality. Such an approach might seem the most direct way of implementing a desired goal, but the resulting system can be fragile. If the requirements change, a system based on decomposing functionality may require massive restructuring. In the contrary, the O-O approach focuses first on identifying objects from the application domain, then fitting procedures around them.

O-O approach has following advantages such as: data abstraction, compatibility (combine software components), flexibility (task allocation in software development), reusability, extensibility and ease of maintenance. Each object of the system can be considered as an autonomous object living in the virtual world. The behavior of the object depends on the input message and on the transition conditions owned by the object itself.

Communication between objects is in abstract: when we specify a unit of behavior, we do not need to know which internal functions will be used, and the functions are commonly unavailable in a high level description. In this way, we can manipulate the
object as an entity, rather than having to handle the collection of separate parameter values that make up the states.

In Figure 4-1, a hierarchic approach for dynamic behavior modeling is proposed. We combine object graphic model and behavior model together. Therefore Object Behavior Modeling can be decomposed into graphic modeling and behavior modeling, the later can be supported by behavior knowledge and physical engine. Behavior models can be represented by states, rule based transition conditions and actions. The definitions of such terms are:

- **state**: state represents an object's status, phase, situation. An object can be transformed from one state into the other in its lifetime. State means changeable information.

- **transition conditions**: transition conditions describe under which condition(s) does the state of the object change. The process of changing a state of an object is defined as transition. A transition consists of a trigger description and an action description, and optional identifier.

- **Actions**: actions can take place whenever a trigger is activated. An action may create or destroy objects, cause events, observe objects and relationships, and send or receive messages. Actions associated with transitions are non-interruptible while actions associated with states are interruptible.

Following this structure, we can model object behavior from high-level to numerical detail, no matter which lower level modeling tool is used such as Petri net or PBM. In most VEs for engineering, the behavior is required to change according to physical law and other rules. We put those rules (or knowledge of behavior) in the behavior knowledge base, detailed discussion on PBM will be given in the following section.
4.2.3 Physically Based Behavior Modeling

Physically Based Modeling is the technique of modeling dynamic scenes for animation based on the simulation of physical behaviors. Though physics and computer graphics has a long history, PBM is a relatively new research area in VR [Badler et al., 1991; Witkin and Kass 1991]. Applying physical simulation to a VE is a challenging work, because of the functional requirements of VEs as photo-realism, animation speed and real-time interaction.

A VE can consist of a set of objects. For each individual object, in addition to the geometrical and graphical attributes, physical properties are also to be specified. Different rules and methods are needed for simulating different kinds of objects' behavior.
Applying physics in the dynamic simulation is straightforward. This technique can produce realistic motion and requires a modest amount of control input. We need only to find the appropriate physics equations and program them. Programming the equations can be quite easy (given that the appropriate set of equations can be found, which is not always the case), but there are two main problems associated with the use of physics in computer animation. The first problem is controlling the motion produced by the dynamics simulations. The dynamics simulations are driven by forces and torques, which are not a natural way of specifying motion for most animators. The average animator and even the average physicist, has no intuitive feeling for the appropriate forces and torques required to produce a given motion. In order to make physical equations generally useful, control techniques that can automatically produce these forces and torques are required. The other main problem is the development of numerical techniques for solving the equations of motion. Numerical techniques should be efficient and stable, so that interactive control of the motion is possible.

**Physical simulation**

Computer animation places two major restrictions on the simulation techniques that can be used. First, animation lasts for a significant length of time. Thus the solution interval for the differential equation will be quite long, and we need a numerical technique that is quite stable and not strongly influenced by round-off error. Second, we would like to have interactive control over the animation. If these two problems were independent, they could be tackled individually, unfortunately, there seems to be a strong relationship between the two problems. A number of the control schemas that we would like to use tend to introduce nonlinearity and stiffness into the differential equations. This greatly complicates their solution and forces us to use more sophisticated techniques. In addition, available solution techniques affect control
techniques that can be effectively used. If the equations can be solved in close to real-time, then interactive control techniques are possible. If the solution requires overnight batch runs, we are forced to look at automatic techniques that rely on feedback or learning.

Deep investigation of physical simulation in PBM of VEs is beyond the scope of this thesis. We will consider some simple problems (actually it is not simple in Computer Graphics) utilizing Newtonian physics.

Milenkovic [Victor J. Milenkovic 1996] categorized physical simulation techniques as acceleration-based [M. Moore and J. Wilhelms, 1988], velocity-based [David Baraff, 1993], or position-based [Z. Li., 1994, Z. Li. and V.J. Milenkovic, 1995]. Acceleration-based methods come closest to simulating true physics, and they are the most expensive to be carried out in computation. Velocity-based methods are less close to true physics than acceleration-based methods but faster. Position-based methods gain speed but scarify the physics reality. We can choose a technique according to practical implementation.

\[
P_i = [x_i, \ y_i, \ z_i]^T
\]
\[
v_i = [\dot{x}_i, \ \dot{y}_i, \ \dot{z}_i]^T
\]

Physical simulation provides state of objects over time. For a rigid body, the states can be represented as position and velocity matrixes, which are consisting of six variables. The position and velocity matrixes of the object are expressed as below:

In a simulation loop, a typical time step is based on computation of forces or velocity, which can be obtained through solution of Ordinary Differential Equations (ODE) that consist of energy or momentum conservation. Figure 4-2 is the simulation control flow and the following is the pseudo code.
Pseudo code

float time, delta;
float state[6]; force[3]
state=get-initial_state();
for (time=t0; time<time_final; time+=delta)
{
    force=force-function(state, time);
    state=ode(force, delta);
}

Take a Ping-Pong ball game as an example: If we need to consider friction and rotation, the situation is much more complex than the rigid body. We need to detect the contact points and decide whether a deformation will occur, and finally compute the contact force. Collision detection might be a problem for the real-time physical simulation. Therefore, we propose a solution for this problem by dividing the paddle (or hand) into grids, maximal n points:

We assume that the deformation mainly occurred on the ball, a pseudo code is developed to address the PBM problem.
Do $i=1,2,\ldots$

$$t_i = i \cdot \Delta t$$
$$t_{i+1} = (i+1) \cdot \Delta t$$

$$r^b_o(t_{i+1}) = r^b_o(t_i) + v^b_o(t_i) \Delta t + \frac{1}{2} a^b_o(t_i) \Delta t^2$$

Do $j=1,2\ldots\text{Nummax}$

Pair-wised ball-paddle collision detection:

Distance between point $P$ on the paddle to center of the ball:

$$d_p = \left| r^b_o(t_{i+1}) - r^b_p(t_{i+1}) \right|$$

if $d_p > R_b$, $R_b$ is ball radius

then {$
\begin{align*}
\text{there is no contact between } P \text{ and ball at time } t_{i+1};
\end{align*}$

$j=j+1$; $
$

else {$

\begin{align*}
P \text{ contact with ball at } t_{i+1}
\quad \text{update contact points}
\end{align*}$

end do

case 1: No contact points at $t_{i+1}$;

$$a^b_o(t_{i+1}) = 0 \quad v^b_o(t_{i+1}) = 0$$

return;

case 2: No contact at $t_i$, but contact at time $t_{i+1}$

$$r^b_b(t_{i+1}) = r^b_b(t_{i+1})$$

case 3: contact at $A$ at both $t_i$ and $t_{i+1}$, then there is a deformation

$$v^b_A(t_i) = v^b_o(t_i) + \omega^b(t_i) \times (r^b_A(t_i) - r^b_o(t_i))$$
\[ \mathbf{a}^b(t_i) = \mathbf{a}_0^b(t_i) + \mathbf{e}^b(t_i) \times (\mathbf{r}^b_{A}(t_i) - \mathbf{r}^b_{O}(t_i)) \]

\[ \mathbf{r}^b_{A}(t_{i+1}) = \mathbf{r}^b_{A}(t_i) + \mathbf{v}^b_{A}(t_i) \Delta t + \frac{1}{2} \mathbf{a}^b_{A}(t_i) \Delta t^2 \]

\[ \mathbf{F}^b_{A}(t_{i+1}) = K \cdot (\mathbf{r}^b_{A}(t_{i+1}) - \mathbf{r}^b_{A}(t_{i+1})) \]

\[ \mathbf{F}^b(t_{i+1}) = \sum_{A} \mathbf{F}^b_{A}(t_{i+1}) + \mathbf{G} \]

\[ \mathbf{a}_0^b(t_{i+1}) = \mathbf{F}^b(t_{i+1}) / m_b \]

\[ \mathbf{M}^b(t_{i+1}) = \sum_{A} \mathbf{F}^b_{A}(t_{i+1}) \times (\mathbf{r}^b_{A}(t_i) - \mathbf{r}^b_{O}(t_i)) \]

\[ \mathbf{e}^b(t_{i+1}) = \mathbf{M}^b(t_{i+1}) / I \]

\[ I = \frac{2}{3} m_b R_b^2 \]

EndDo

From the above example we can have an idea of how the underlying physical simulation is going in the dynamic behavior simulation.

### 4.3 Summary

Dynamical behavior modeling is of critical importance in enhancing the sense of realism in VEs. A hierarchical structure for dynamical behavior modeling is proposed based on characteristic study of simulation entities and behavior levels in a VE.

By comparing existing behavior modeling tools or methodologies, an O-O approach is adopted. The flexibility, reusability problems of VE development are then partially solved.

Physically Based Behavior Modeling as an instance of dynamical behavior modeling is specially discussed in this chapter.
5. Implementation and Feasibility Findings

5.1 Introduction

In order to test the feasibility and implement the framework, VPPG is developed, which is a pilot study supported by Ford Motor Company [Mitchell Tseng et al, 1998]. In VPPG, there is a virtual hall with four walls, one ceiling and one floor. There is one to several balls (decided by the user) bumping inside. A user can hit the ball with a paddle. The ball will bounce back after it hits the wall, the trajectory of the ball is determined by the force and impulse reacting on it. In order to simplify the problem, the Ping-Pong ball is considered as a rigid object, no deformation occurring. The features of the scenario are: the motion status of the ball can be autonomous in the sense of being determined by physical law instead of being predefined by the developer, but also can be changed by unpredictable user interaction. There are multiple moving objects (Ping-Pong balls) which are not isolated (can collide with each other) and constrained by the environment (can not pass through walls or floor of the hall). Different from common PC-based games, the response of the ball after being hit by the paddle is based on the previous state of the ball and user input. The user input is in a more natural and intuitive way than predefined input pattern such as using key board or mouse button.
5.2 VPPG Construction

5.2.1 System configuration

The application software platform is World Toolkit® (WTK) release 7 on IRIX 5.3 or 6.4. We chose WTK because it is a software development system, which has a C library providing high-level functions for VE construction. It also provides an object-oriented modeling system allowing the display and animation of hierarchical objects. The single loop simulation model used in WorldToolKit is a standard approach which sequentially reads sensors, updates the world, and generates output graphics. A virtual hall and balls are created by using graphic models provided in WTK. A paddle is imported from 3D Studio Max. Lighting and color are specified. The world coordinate is determined by the right-hand method, the positive z-axis points to the paper, and y-axis is in the direction of gravity. Figure 5-1 describes the VPPG system configuration.

Figure 5-1 Virtual Environment for VPPG
5.2.2 User interface

It is critical to select suitable input devices since it will determine the style of human computer interaction. Overall performance of the VE is affected by the integration of user with a VE through input devices. Traditionally, keyboard and mouse are used to conduct user interaction such as moving forward and backward or navigating around in the VE, selecting or picking up objects. If we choose this interaction paradigm, depth information of paddle motion and quantitative information of motion is unavailable since motion in the VE is in fact change of viewpoints. We improved the traditional interaction paradigm by adding an interaction mode in which the mouse movement is linked to the paddle instead of the viewpoints. This interaction paradigm is more intuitive and demands less mental load. We can easily get the quantitative information of the paddle movement. The most obvious strong point is that the control of input is easy and unambiguous, and the weak point is the limited Degree of Freedom (DOF), which causes lacking of depth information to control the movement. In order to perform six Degree of Freedom manipulation, we need to choose other input devices. Dataglove is widely accepted as a user input device. It generally has 18 sensors, use gesture to specify the input instruction such as move forward, backward, grasp and release. It needs calibration according to the individual user, and gesture recognition is not an easy job in practice. Dataglove keeps tracking each joint of fingers, which is unnecessary for moving a paddle and is a computing resource waste. Furthermore, we need to write some code to get the motion information from Dataglove since it is not provided in WTK. As a result, we choose Polhemus FASTRACK as the input sensor. We attach the sensor on a real paddle to enable an easier user manipulation in the real world. The position/orientation of the paddle in the user's hand is monitored by the sensor whose motion is linked with the
paddle in the VE, and the virtual paddle is interacting with the ball. The user's action in the real world is thus mapped into the VE. Consistent mapping is the prerequisite for successful user interaction. In addition, in a VE where has no force feedback, there is no output to the user's hand no matter if the paddle have hit the ball or not. The data flow between the real hand and the virtual object is basically one way only. As a result, the movement of the hand can not be restricted, the position of hand (paddle) could easily become either different from that of the virtual paddle or out of the mapping space (as a result, the paddle will be out of the hall). The manipulation in VE is then difficult because users must do it without the haptic contact with the object they rely on in the real world to orient themselves and their manipulation. Though there is considerable ongoing research in the area of active haptic feedback [Durlach 1995], general purpose haptic feedback devices that do not restrict the mobility of the user are not yet practical or available.

5.2.3 Dynamic behavior modeling

According to the IVECS framework, VPPG can be considered as a simple process, which is consisting of a single activity (UOB). From the UOB we can get information about the object entity involved such as the Ping-Pong ball, the avatar (the paddle), and the activity (hit the ball). The VE is comparatively simple. The main consideration of VPPG is modeling the behavior (motion) of the Ping-Pong ball (the paddle is controlled by the user). Traditional method of modeling behavior (shape variation and motion) in computer graphics is pre-specified such as keyframe animation of bodies and control points of the object. The disadvantages of such method are either un-realism or a large burden for the modeler. In our case, the scenario of multiple moving objects and unpredictable user interaction is more
dynamic and complex than a single isolated moving object. Therefore, the traditional animation method can not solve the problem.

In order to model the dynamic behavior in VE instead of artificially developing some animation, we make some assumptions to simplify the underlying physics of the Ping-Pong game and to reduce the difficulty of the problem.

Assumptions:
The ball is rigid, no deformation during contact;
There is no friction, and therefore no rotation;
There is no energy loss during collision.

**Object Oriented Ping-Pong behavior modeling**

Figure 5-2 shows how we can encapsulate common PDE or ODE solver or other numerical techniques as "black boxes".

![Diagram of Ping-Pong behavior modeling](image)

**Figure 5-2 Paradigm of Ping-Pong behavior modeling**

We can follow the hierarchic approach for dynamic behavior modeling described in Chapter 3 and in Chapter 4. Since the standard behavior lib and physical lib are not
available yet for WTK, we still need to model the motion of the ball from scratch. We use the O-O concepts in the Ping-Pong game and encapsulate physical law into behavior models.

Communication with environment

The Ping-Pong ball can be considered as an object entity. The original state can be the input message of the entity, data from input devices such as mouse or Dataglove is another kind of message between VE and individual object entity.

5.2.4 Technical Challenges

Among key issues mentioned at the beginning of this thesis, we concentrate our efforts on Object Behavior Modeling, which is still representing a technical challenge in VR applications, moreover, it is one of the core parts of our proposed framework. In this section we will introduce the technical challenges of dynamic behavior modeling and our approaches to those challenges.

Collision detection

In VPPG, the ball moves in response to gravity (ignoring the friction, wind, or air resistance, we add a coefficient to indicate energy loss) and the impulse (infinite forces applied instantaneously) given by the wall and the paddle. The impulse is caused by collision between objects, therefore, precise and fast collision detection is the foundation of PBM. The computation of collision detection is time consuming. In order to achieve real-time display and simulation, fast and precise collision detection is the prerequisite. However, there is an inherent conflict between discrete sampling systems and fast collision detection requirements.
Control of behavior

Mathematically, physically-based models are often translated into differential equations, which are typically posed as initial-value problems; the user has little control other than setting up the initial configuration. In a VE, object behavior is determined not only by physical law, but also by other higher level rules or unpredictable events. Furthermore, the behavior of objects can be affected by each other. Therefore, the behavior can be deterministic and non-deterministic at the same time. Control of such kind behavior still remains a research issue.

In VPPG, the motion of the ball is both continuous in the interval of each bouncing and discrete when it collides with other objects such as the paddle, other balls and walls. Therefore, only one initial condition is not enough. First of all, it is necessary to detect collision as well as the contact point of collision and program the appropriate response. When the paddle hits the ball, i.e., collision is detected, the solution of governing equation of momentum conservation should be triggered to give the velocity of the ball. For each collision between the paddle and the ball, the “initial condition” should be re-calculated by the displacement of the paddle.

Real-time physically based interaction

One of the most important features of a VR system is interaction. However, most of today's VR and visual simulation systems are still rendering a static environmental representation which is prepared by a polygonal modeling package or a dynamic environmental representation by pre-computation. These systems do not allow real-time changes to the virtual environment.

User interaction in this project is in fact 3D direct manipulation. Different from picking or grasping, it is based on user's judgement (perceptual skill) and rapid response (motor skill) to the motion of the ball, a real-time feedback loop needs to be
involved. Many contemporary graphics systems can not provide this kind of throughput.

Since the interaction should follow a physical law, only real-time is not enough, it also should be physically correct in terms of objects behavior. This may be more difficult because of hardware limitation.

Integration of simulation and visualization/data communication

To obtain interactive real-time simulation results, we need to integrate numerical simulations with graphic rendering. How to allocate the computing resource to numerical simulation and graphic processing is another technical challenge. In the Ping-Pong case, the object is a rigid sphere without deformation, no complicated mathematical computing is required so the problem is not that obvious. If we need to model and display a car crash, and conduct crash worthiness analysis in a VE, computing resource must be a problem to enable real-time interaction.

5.2.5 Our approaches to address the challenges (solution)

In this section, we will introduce the technical details to meet the challenges mentioned above. Since the dynamical behavior in the Ping-Pong game is mainly physically based, we will focus on the physical behavior modeling of the ball. The behavior of the ball is mainly consisting of motion.

Motion of a rigid object is separated into transnational motion of the center of mass (linear velocity) and rotation of the body about the center of the mass (angular velocity). At the first stage, we ignore the frictions and only consider the linear velocity. Therefore, the ball can be treated as a point mass.
Collision detection

There are four kinds of collision: Collision between ball and ball, ball and wall, paddle and wall, and ball and paddle. For the first three kinds of collision, we can treat the ball and paddle as point mass (the paddle is much smaller in size compares with the wall, and the collision is only to keep paddle from penetrating the wall), the problem of collision detection is greatly simplified. We simply compute the distance between two objects (can be ball to ball, or paddle to wall, or ball to wall) in each time step (the time interval is determined by the frame rate). If the distance is less than the diameter, then two balls will collide; if it is no larger than the ball radius, then ball and wall will collide. Similarly, if the distance of paddle to the wall is less than the larger half size of the bounding box of the paddle, the paddle is considered to collide with the wall. As for the ball and the paddle, it is no longer suitable to consider it as point mass since the paddle is not a sphere and the size of paddle compared with the ball is not big enough to be considered as a plane. After several trials, we use bounding box algorithm to detect collision between these two objects.

Control of behavior

Control of behavior in the Ping-Pong game is a matter of simulation technique. In this project, when simulating motion of the ball and computing response to collisions, we used the velocity-based method to avoid integrating acceleration to velocity, which needs very small time-steps when the acceleration is very high. Besides, it is not suitable to use the acceleration-based method since we consider the ball and the paddle as rigid objects, they are allowed to contact but not to overlap, contact force is an instantaneous impulse. While typical acceleration-based methods like spring model methods compute force proportional to the amount of overlap. We did not chose position-based methods because the motion is less reasonable for what we are looking
for physically correct behavior. Assuming there is no energy dissipation, it is reasonable to use momentum conservation law and energy conservation law to form the governing equations. Though the velocity-based methods are much more stable and faster, two problems still remain. One causes small time-steps and thus high computational cost; another problem is rattling after collision, we will discuss it in practical issues.

Real-time physically based interaction

User interaction needs behavior modeling and interface design, and these two parts are closely interrelated. To provide a physically based interaction environment, we need to model the collision response following the collision detection. The physically based interaction requires for suitable input and output devices. Therefore, keys to real-time user interaction or more precisely fast 6DOF manipulation are a sufficiently high update rate of graphic displaying and accurate mapping form real world to virtual world. Current available input devices in our lab have some limitations to fulfill the interface requirement as discussed previously in user interface. The movement of the paddle is hard to control without force feedback and consistent position mapping. We selected two types of input devices. At first, we used a 2D mouse as the user-input device. The problem of the 2D mouse is its limited DOF. We fixed the hit direction of the paddle to solve this problem. The paddle is attached to a mouse motion sensor. Velocity of the paddle is directly related to the mouse position variation, through which we can change the state of the ball by moving mouse to hit it. The effects of mouse moving speed are reflected in ball's collision response. In order to have a more intuitive and nature interaction paradigm, we selected Fastrack, a 6DOF sensor which can carry out 6DOF manipulation, but the collision response modeling is computational expensive.
In addition, the user's perception plays an important role in interaction. Newtonian physics assume that we perceive the world on the basis of the detection of simple or lower-order variables such as distance and velocity. Then these variables are combined in the brain to make something more complex such as prediction of time and position of collision. However, the time to collide is readily predicted without specifying either the speed or the distance. The time-to-contact, is given in the expansion gradient on the user's retina. It is given by the speed with which a texture point (at a particular distance from the expansion center of the optic flow) moves away from that center. This variable, plays a part in the Ping Pong game, particularly in deciding when precisely a hit should be initiated, cannot be faithfully simulated on an ordinary computer, where there is no interaction between motion and vision; but it is possible in a VE.

Some human factor issues are unclear to VE system designer, but should be taken into consideration. e.g. Biological systems do not perceive the world on the basis of extrinsic units, such as centimeters or seconds, which are meaningless to the observer, but in terms of intrinsic units which are linked to the body, such as eye-height, or to the action, such as time to contact.

Integration of simulation and visualization /data communication

![Diagram](image)

Figure 5-3 Integration of visual model and physical simulation
Since the graphic model of the ball is very simple and we assume that there is no deformation, we simply used a data structure to associate the graphic with the behavior (state). Integration of visual model and simulation is described as Figure 5-3.

![Image](image.png)

Figure 5-4 Snapshot of the physical simulation
Figure 5-4 is a snapshot in the game, in which the user is hitting toward a ball.

5.2.6 Other practical issues

Theoretically, following approaches mentioned above, we could develop the game now. Yet there are still some practical issues to be solved to get a reasonable and realistic motion of the ball. The usability of the user interface is another important issue to be addressed.

Reasonable and realistic motion simulation

In the developed VE, it happens occasionally that two balls stick together for a while, or the ball rattles on the paddle or the wall. This problem is caused by small time interval between collision detection required by velocity-based method. To solve this problem, we eliminate the ball-ball rattle by adding position tolerance to the ball after collision so that balls can escape from the infinite looping (separate two balls after
collision is once detected). Similarly, we solve the ball-paddle rattle by adjusting the rebounded velocity of the ball to escape the infinite looping. Another problem is that even use the real-time algorithm to detect collision, the paddle will still penetrate the wall if the moving speed is fast. We found that it is caused by excessive collision detection. We detected the paddle with six walls in a loop, while actually the paddle might collide with wall No.6 when the collision detection is still processing for the other walls. By eliminating unnecessary collision detection, the paddle will not penetrate the wall even with a fast moving speed.

**Usability of the user interface**

The user interface is the main issue affecting the performance of the game. At first, motion track is a problem since we used the default local frame to link the sensor with the paddle. Therefore, the mapping of user's action is not consistent after the paddle rotates for a certain angle. It causes a big problem for the user to control the paddle motion. Even by using HMD, the sense of immersion will not help a lot. Since the depth information is not accurate enough to aid a 6DOF dynamic manipulation in the VE. It is a tough job to hit a moving ball in the VE. We have spent much time to improve the interface until we improved the motion track by using world frame for the paddle to get a correct calibration.

Though it has no significant effect on the task performance (this statement is according to our subjective feeling, and needs to be tested by experiments), it is still worth to mention that, in a VE which has no force feedback, there is no output to the user's hand no matter if the paddle hits the ball or not. The data flow between the real hand and the virtual object is basically one way only. As a result, the movement of the hand can not be restricted. The position of hand (paddle) could easily become either different from that of the virtual paddle or out of the mapping space (as a result, the
paddle will be out of the hall). The manipulation in a VE is then difficult because users must interact with the object without the haptic contact, with which they rely on in real world to orient themselves and their manipulation. Though there is a considerable ongoing research in the area of active haptic feedback [Durlach 1995], general-purpose haptic feedback devices that do not restrict the mobility of the user are not yet practical or available. As alternatives to force feedback, we add visual aids and auditory feedback. In detail, the color of the ball will change to red after it collides with the paddle. We also add grids to the wall of the hall to help identifying the 3D position therefore improving the performance of manipulating. We also add sound effects whenever the ball collides with the wall or the paddle. These visual and auditory aids might enhance the sense of realism and improve the task performance. An evaluation of the developed system can be carried out if necessary.

**Integration of technologies and design metaphors**

For the implementation, the developer has to select and integrate appropriate techniques across several areas of computational research include dynamic databases, real-time operating systems, three-dimensional modeling, real-time graphics, multisensory input and display devices, fly-through simulators, video games and so on.

Designing effective VE metaphors could enhance human performance in VEs. It could also be a means of assisting in the integration of multimodal interaction.

VE designers have been guided by the design metaphors available in most 2-D toolkits in the past. Although they might have endless design possibilities, they also have limited guidance in designing effective human-VE interfaces. VE are in need of new design metaphors uniquely suited to their characteristics and requirements. The key to the design of these new metaphors may be to identify expectations developed
through real world experience in common and familiar environments and to use this knowledge to design consistent VE interaction metaphors. This approach is an extension of the anthropomorphic approach that uses models of human-human communication to direct the design of human-computer interaction. Using this approach, the constancy, expectations, and constraints elicited from interactions in the real world should be designed into VE metaphors, thereby potentially engendering a more intuitive interface design.

5.3 Summary

In this chapter, we accomplished some basic physical simulation and successfully integrated visual model and simulation in a straightforward manner. Technique challenges for physically based VE, especially for VPPG are discussed and corresponding solutions are given.

Our work represents that implementation of physically based modeling itself in a VE is expensive but not a major problem. The major difficulty is the integration of software and hardware for a reliable user interface. To enhance the sense of realism, multi-disciplinary efforts from physical, ergonomic domain to psychological, mental domain as well as modeling technology need to be combined together.

The IVECS framework provides an organization of the VE system. If we want to change the scenario of VPPG, e. g. two players will be involved, we can change it from process level, where two actors are specified, and the rest of the original system should be the same.
6. Conclusion and Future work

6.1 Conclusion

Although VR has great potential to be applied to a wide range of applications including complex SE, high cost of VE construction and low fidelity of VE make VR less popular in engineering applications. Little has been done in research level to integrate VE with SE. Improve the sense of realism for a VE and reduce the lead time of VE construction are the main research goals of this thesis. In this research, the followings have been completed:

➢ Integrated VE for Complex System (IVECS) is presented as a framework. This framework uses IDEF3 as process modeling tool, and Object Oriented (O-O) dynamical behavior modeling is integrated in the IVECS. Though future work is still needed, IVECS is hopeful to reduce the high cost of VEs construction and development. IVECS has benefits on SE and VE integration, and it is capable of delivering high quality design and obtaining better user performance and user satisfaction.

➢ A hierarchical structure for dynamical behavior modeling is proposed based on characteristics analysis of simulation entities and behavior levels in a VE. By comparing existing behavior modeling tools or methodologies, an O-O approach is adopted. The flexibility, reusability problems of VE developing are then partially solved by O-O methods. Physically Based Modeling provides a
straightforward approach for dynamic behavior modeling without predefined animation.

➢ In the end, as an implementation example, a Virtual Ping Pong Game is developed following the IVECS framework. Key issues of PBM in a VE are discussed and corresponding solutions have been given. Physical simulation of user interaction is addressed by a careful study of interface.

6.2 Problems and Future Development

Though the efficiency of IVECS is obvious, there is still a long way to go to make the framework fully functional. The mapping from process level to entity level needs some automation capability. Artificial Intelligence is needed for coordination and automatically generated VR models. Standard libraries also have some problems - no matter how complete the list of properties or behaviors for any object has been built inside the VE construction platform, there will still be a new type of object or object behavior that is needed by other applications. Though we want to make the framework general enough for a wide range of applications, there is still a trade-off between generalization and flexibility. Additional work for individual application is still needed, what the framework tries to achieve is to reduce the redundant work and make it reusable, provide a standard approach and communication platform. Task analysis and requirements analysis for the target VE are essential for effective construction of VEs. Take a car crash test as an example, currently, it might not be feasible to create a VE for testing that is the same as the real world test. Modeling the behavior in a physically correct manner is the most critical objective compared with "ease of use" or "photo realistic representation".
The ultimate goal of this research is to enable effective construction of physically based VE for engineering design and analysis without the cost of physical setup and investment. What we did up to now is the first attempt and there are still many technique voids.

In the PBM of VE, collision detection is still a problem for irregular objects. And the response time of physically based user interaction is determined by the numerical technique.

To make the IVECS really useful, more efforts need to be spent on knowledge base and artificial intelligence.

More attention should be paid on human factor study together with research on input and output devices.
Reference


Nadia Magnenat Thalmann, Daniel Thalmann, Interactive computer animation, 1997.


Appendix

A. Notation

\( r_A^b(t) \) position of contact point A (ball surface) at time t;

\( v_A^b(t) \) velocity of contact point A (ball surface) at time t;

\( a_A^b(t) \) acceleration of contact point A (ball surface) at time t;

\( r_O^b(t) \) position of the ball center O at time t;

\( v_O^b(t) \) linear velocity of the ball center at time t;

\( a_O^b(t) \) linear acceleration of ball center O at time t;

\( r_A^p(t) \) position of contact point A on the paddle at time t;

\( \omega^p(t) \) angular velocity of the ball at time t;

\( \epsilon^b(t) \) angular acceleration of the ball at time t;

K material coefficient;
B. Demo description

The Virtual Ping-Pong Game includes: One executable file; One source .c file; One .3DS file; Three .wav files.
The operation guide is provided when the file is executed.
In the demo, users can add or eliminate gravity, open or close sound.
There is a list of instructions for a flexible game by simply pushing a key:
<b>: Reset the original position of ball
<ss>: Disable the sound
<ff>: Flip between to use mouse change viewpoint and hit balls
<gg>: Add gravity
<hh>: Reset the original position of paddle
<vv>: Reset viewpoint
<tt>: Show the simulation time, framerate & no. of frames
<?: Display help menu

System requirements and setting

Hardware:
1) SGI Infinity Reality Onyx2
2) Acoustetron II (Crystal River Engineering) sound server
3) Polhemus Fastrack

Software:
1) SGI OS IRIX 6.4
2) WorldToolkit Release 7.0

On SGI platform:
   Environment variables setting for sound server:
   TRONCOM=x@384
   TRONDEV=/dev/ttyd3
On sound server platform:
   Boot the sound server at 38400 baud rate
On Polhemus Fastrack:
   Set the Polhemus Fastrack at 38400 baud rate

Cable Connection:
1) Connect Acoustetron II sound server to SGI Onyx2, port 3 (tty_3)
2) Connect Polhemus Fastrack to SGI Onyx2 port 2 (tty_2)

A video clip can be downloaded from IEEM server. For detail, please contact icvzxk@ust.hk.
C. Feedback from Ford Motor Company

The followings comments are cited from Ford's feedback about our VPPG project.

"...some of the pioneering work you are doing at the University may be directly applicable to the automotive industry and the VCL (Visual Computing Laboratory). … Through collaboration and future research efforts we will be able to bring to fruition many industry firsts in simulation environments and modeling."

"I am very interested in working together and hope that in the near future we can discuss how we can work together in the coming year."